A PRINCIPAL COMPONENT APPROACH
TO ANALYZING SIMULATION OUTPUT

A Dissertation

by

THOMAS FREEMAN, JR.

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 1992

Major Subject: Industrial Engineering

A PRINCIPAL COMPONENT APPROACH TO ANALYZING SIMULATION OUTPUT

A Dissertation

by

THOMAS FREEMAN, JR.

Approved as to style and content by:

R. E. Shannon

Robert E. Shannon
(Chair of Committee)

M. A. Wort

Martin A. Wortman
(Member)

Gary L. Hogg
(Member)

R. R. Hocking

Ronald R. Hocking
(Member)

Gary L. Hogg
(Head of Department)

August 1992

ABSTRACT

A Principal Component Approach
to Analyzing Simulation Output. (August 1992)

Thomas Freeman, Jr., B.S., United States Military Academy;
M.S., University of Southern California
Chair of Advisory Committee: Dr. Robert E. Shannon

A new procedure, called the principal component method, is developed to handle the problem of data correlation in simulation output analysis. The method is derived from matrix diagonalization theorems, which allow for an orthogonal transformation of data with an estimated covariance structure into a version of the data with uncorrelated structure. Matrix manipulation of this uncorrelated version of the data yields a derivation of an unbiased estimate of the underlying process mean and an estimate of the standard error of the mean. Using the Central Limit Theorem, the confidence interval is constructed. The performance of this confidence interval methodology is empirically tested over several independent replications of M/M/1 queueing models set at various utilization rates and of time series models with known correlation structures. Compared to the batched mean procedure, the principal component method provides good coverage, acceptable half-width information, and excellent bias information.

ACKNOWLEDGEMENT

First and foremost, I thank my Lord and savior, Jesus Christ, for his blessings and his guidance, which keeps everything that I do in this world in the proper perspective.

I thank the members of my graduate committee who persevered through this academic ordeal with me and provided constant support throughout.

I appreciate the efforts of my chairman, Dr. Robert E. Shannon, who took a gamble on my academic potential, staked his professional reputation that I could succeed, and patiently guided me to the finish.

I, also, thank Dr. Martin A. Wortman, who more than anyone else understood what I was going through and offered the encouragement and assistance to continue during those times when I thought all was lost.

I, also, appreciate the efforts of Dr. Gary L. Hogg, who raised my expectations of what I thought I could do.

I owe much to Dr. Ronald R. Hocking, who provided the statistical insight that allowed my research to progress to a successful conclusion.

I thank my wife, Georgia, and my kids, Tom and Janet for their love for me and their independence to carry on with the family "business" without me.

Finally, I thank my parents, Tom and Bertha, who early on instilled in me the drive to be the first of the Freeman clan to earn a professional degree.

DEDICATION

To all who care to know the truth:

"Fear God and keep his commandments:
for this is the whole duty of man.
For God will bring every work into judgement,
with every secret thing, whether it be good,
or whether it be evil."

--Ecclesiastes 12:13-14

To my son, Tom:

"It is only as a man puts off foreign support and
stands alone, do I see him to be strong and to prevail.
He is weaker by every recruit to his banner,
for is not a man better than a town."

--Anonymous

To my father, Tom, my wife, and the rest of my family:

"It's never too late to start,
Never too soon to unbind,
The dreams of your heart,
And the visions of your mind."

--Tom McCauley

To my grandfather, Mr. Otis, and his father, Thomas:

Question, what's in a name?
Answer--Only your character!
Thanks for character.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Much time and effort is spent by the simulation analyst in modeling and generating simulation output. Too often, little effort is directed toward properly analyzing the output. Simulation analysts may overlook the fact that a simulation is really an experiment, and not just another computer program to be run. Therefore, the simulation experiment contains a variety of random elements that interact in many unexpected ways. In fact, the novice simulation analyst too often relies upon the output of a single run as an "answer". More experienced analysts realize that a simulation's output is a realization of a random process and requires a confidence interval estimate.

Often, analysts inappropriately apply classical statistical procedures to the output when the assumptions for such procedures are not met. For example, most classical procedures for analyzing statistical data require independent observations, time invariance, and normality. Typically, a simulation's output fails to satisfy one or more of these assumptions, with failure of the independence assumption being the most critical (Law [1977]).

This dissertation follows the style and format of *Operations Research.*

1.1 Background

Consider a simulation analyst who is asked by a bank's officials to determine whether or not the bank should change from an existing service teller configuration to a proposed teller configuration. A performance measure of the existing setup that the analyst may be interested is the average time that a customer spends waiting and getting serviced in the bank. The analyst develops, verifies, and validates a simulation model of the bank that accurately portrays the daily activities of the bank. After the analyst "warms up" his bank simulation through the transient period, he amasses the steady state data, X_i, of the time that the *ith* customer spends in the bank. At the end of the simulation, the analyst uses this data to calculate the average time a customer spends in the system, $\bar{X}=\frac{1}{n}\sum_{i=1}^{n}X_i$. However, the single number which he determines as the average does not accurately reveal how close it is to the true mean, μ. One realization of the simulation gives an unacceptable estimate of the true mean.

Now, if the analyst runs several independent replications of the experiment, he will get several different batched averages $\bar{X}_i$ as estimates of the true mean. Taking the grand average $\bar{\bar{X}}$ of all of the batched averages for each replication gives a better estimate of the true mean. It is closer to the true mean (has less bias), yet, the analyst cannot tell how close. A measure of the variation about the grand average, $s^2=\frac{1}{n-1}\sum_{i=1}^{n}(\bar{X}_i-\bar{\bar{X}})^2$, assists the analyst in constructing a classical Student

t confidence interval on the true mean, $\bar{X} \pm t_{(n-1,1-\alpha/2)} \sqrt{\frac{s^2}{n}}$. This confidence interval tells how close the grand average is to the true mean. Its common, though erroneous, interpretation is that the analyst has some designated level of confidence that the true mean lies between the two confidence limits.

Yet, there is a troublesome situation inherent to the bank operation process which may cause the analyst to place more confidence in the confidence interval than he should. The inherent situation of the process is that customer waiting times are dependent on prior customer waiting times. The strength of this dependency, or correlation, affects the accuracy of the variation measure and, consequently, the size of the confidence interval. If due to positive correlation, the confidence interval is shorter than it should be, then the analyst, erroneously, interprets the estimate as being closer to the true mean than it actually is.

1.2 Statement of Need

Much of the research work in the analysis of simulation output has been directed toward solving the problem of data correlation. Such methods include replication/deletion methods, batching methods, time-series approaches, spectral approaches, regenerative, and standardized time-series approaches. Most methods are based on the premise that some essentially uncorrelated version of the original data can be obtained so that classical confidence interval procedures then become applicable. However, all of the methods suffer some serious drawbacks to the practical simulation analyst.

For example, the replication/deletion methods are very costly with respect to computer time and they require multiple passes through the transient period. The batch means methods require that the batch size be tediously and arbitrarily determined. The time-series and spectral approaches require great statistical knowledge and expertise on the part of the analyst. The regenerative methods are somewhat dependent on the structure of the underlying process since the process must have identifiable regeneration points. The standardized time-series methods are promising, but require a strong mathematical background in order to understand and use the derived central limit theorems necessary for constructing confidence intervals. Therefore, a method of simulation output analysis which has a simple, yet rigorous, theoretical framework is needed that is easy to implement, and is efficient with respect to computer effort in application.

1.3 Scope

This research is exploratory in nature. It derives a fixed-sample-size method of analyzing simulation output and tests the performance of the method against several widely accepted models and performance measures. It is limited by the storage and computational capacities of a stand-alone personal computer. It does not consider the practical problems of identifying the initial transient period and of determining the batch size required in order to ensure uncorrelated batches.

1.4 Applications

This research is intended for the benefit of the simulation analyst who must take simulation output and construct an appropriate fixed-sample-size confidence interval on the steady state mean response for a single system. The results of this research has applications in operations research, system analysis, manufacturing, and statistics.

1.5 Organization

The remainder of the dissertation is organized as follows. Chapter II is a literature review, which describes the several methods of simulation output analysis and the state of current research in the area. Chapter III describes the problem, the research approach, and experimental methodology. Chapter IV explains the empirical results of the performance of the proposed confidence interval method on some well-known analytical models. Finally, conclusions and recommendations for further research are offered in Chapter V.

CHAPTER II

LITERATURE REVIEW

2.1 Methods of Output Analysis

Methods of analysis to solve the problem of serial correlation are classified into four categories (Law [1983]):

(1) those that seek independent observations;

(2) those that seek to estimate the covariance in output variables;

(3) those that exploit the probabilistic structure of the underlying process;

(4) those that are based on standardized time-series.

Such methods may be a fixed-sample-size procedure or a sequential procedure. Fixed-sample-size procedures are those that are done on a simulation of arbitrarily selected sample size such that the width of the resulting confidence interval is fixed. Sequential procedures are those that allow the sample size of the simulation to be continually increased until a confidence interval of a desired width or precision is obtained. This research does not address sequential procedures.

Excellent surveys of the various output analysis methods are found in Law [1983], Kelton [1983], and Law and Kelton [1984]. Briefly, each method of analyzing simulation output is summarized and discussed.

2.1.1 Replication/Deletion

The method of replication/deletion involves making several independent simulation runs, truncation of some initial portion of each simulation run, and treating the runs as independent and identically distributed normal (NIID) random variables. The independent runs are obtained by using different random numbers for each run. However, there are no readily agreed upon methods for determining how much data to delete and how long to run the simulation. If the truncation point is not appropriately determined, either the sample mean is a biased estimate of the true mean or too much data is wastefully discarded. Several methods to determine the truncation point and the effectiveness of the method are discussed.

Gafarian et al. [1978] formulated a definition for the initial transient problem and proposed a set of evaluative criteria for truncation rules. Five commonly used rules--Conway, modified Conway, crossings-of-the-mean, cumulative mean, and Gordon--performed poorly against their evaluative criteria. By modeling the simulation output as an autoregressive (AR) process, Fishman [1972] determined that ad hoc truncation of initial data reduces bias as intended, but also decreases statistical reliability due to a sometimes significant increase in the variance.

Donnelly and Shannon [1981] developed a method based on minimizing the mean square error to determine the number of replicates to run and the number of observations to retain for estimation. The method works well for AR(1) models with short transient periods, but requires a lot of computation in order to do the mean square error minimization routine. By looking at an AR(1) process, Kelton and Law [1984] quantified the effect of data truncation on the performance of the replication method. The method is viable if there is a small number of replications, say five to ten, and if a small undetermined portion of the data is always deleted. The question of how much data to delete is left open.

Schruben [1982] developed an initialization bias statistical test from a standardized time series transformation of the data into a Brownian bridge stochastic process. The Brownian bridge process is decomposed into a signal and a noise component. Evaluation of the signal component for peaks provides the test for initialization bias. For five analytical models, the test proved to be valid and reasonably powerful in detecting the presence of initialization bias. Heidelberger and Welch [1983] developed a procedure to detect non-stationarity of the initial transient period in order to control the run length. The procedure is likewise based on a Brownian bridge model that detects the initialization bias and performs well if the simulation run length is sufficiently long relative to the initial transient period.

Welch [1983] discussed graphical techniques for determining the end of the transient period. Such techniques consist of:

(1) determining the stopping point in the simulation such that run lengths thereafter yield essentially identical histograms;

(2) plotting the sequence of sample means versus sample size and determining the sample size the sequence converges to;

(3) smoothing the above plot such that short term fluctuations are removed without affecting the long term trend.

Kelton and Law [1983] developed a new approach to estimate when the transient period has ended. They derive the transient expectation function (TEF) from a plot of $E(X_i)$ versus i. This shows the observation at which the mean has settled down. A regression based test for "flatness" is done in order to determine the truncation point. Schruben et al. [1983] developed a family of powerful hypothesis tests for determining if initialization bias is present in truncated output. The test is robust and requires little computation. Pritsker [1986] summarized some truncation rules and identified several difficulties with using them on large-scale models. Schruben's technique is seen to be the solution to these difficulties.

Vassilacopoulous [1989] developed a powerful non-parametric test that detects the initialization bias of M/M/s queues. His test has the advantage of not involving the variance of the output and of being robust and efficient. Whitt [1991] proposed that usually one long run, deleting an initial portion of less than 5 % of the run, is more efficient than many replications.

The problems with some of the more promising truncation methods that are mentioned above are detailed in Law [1983]. First Schruben's procedure is not really an algorithm to determine the truncation point, but rather a test for initialization after truncation is done by some other method. Next, Welch's graphical procedures may require a lot of replications for highly variable stochastic processes.

Finally, Kelton and Law's procedure works well for a variety of processes, but a theoretical limitation of the procedure is that it assumes monotonicity of the TEF. This assumption limits the applicability of the procedure.

2.1.2 Batch Means

The method of batch means involves dividing up the observations of one long simulation run into *n* batches of length *l*. If the batch size *l* is large enough then the means of the observations making up the batches will be essentially uncorrelated and normally distributed. Much research work has been done to determine the batch size necessary to ensure uncorrelated batches.

Brillinger [1973] provided the statistical justification for the use of the batch mean method to construct a confidence interval for a mean response. The justifications is that normally distributed and independent conditions occur as the number of observations get larger and the batch size increases. However, the application to simulation output is effectively described in early editions of the text by Gross and Harris [1985], p.487. The key question in batch means approaches is how to determine the optimal batch size that ensures uncorrelated batches, yet provides an adequate enough number of batches to construct a valid confidence interval.

Law and Carson [1979] gave a sequential procedure for determining the batch size that insures uncorrelated batches. Their procedure is based on a statistical test for independence between batches. Other algorithms which, likewise, determined the batch size based on statistical tests for independence are described in Fishman [1978], Schriber and Andrews [1979], Mechanic and McKay [1966], and Schmidt and Ho [1988]. Adam [1983] developed a sequential procedure that out-performs Law and Carson's method and Fishman's method for high utilization queueing systems. The procedure is also efficient and easy to implement.

Kang and Schmeiser [1987] considered the properties of the batch mean method when the underlying process is an autoregressive moving average (ARMA) time-series. The key conclusion is that the batch means process of an ARMA process is also an ARMA process with easily determined parameters. The variance and covariance structure of the batched ARMA process is shown to be a function of the batch size. This provides enough insight into the structure of the batched process to allow the optimal batch size for uncorrelated batches to be determined without having to use a large number of batches.

Bischak [1988] developed a weighted batch means approach that is an improved method for small sample simulations with strong positive correlations. The weights, however, are obtained by an extensive optimization scheme on an ARMA representation of the simulation output.

Other directions of research in the area of batch means has followed four directions:

(1) Seila [1984] batched discrete observations using equal time intervals;

(2) Schmidt and Ho [1988] deleted observations between batches in order to

decrease the correlation between batches;

(3) Kang [1984] and Chun [1989] modeled the batch mean process as an ARMA process in order to investigate its properties;

(4) overlapping batch mean methods have been developed by:

(i) Ho [1986], who develops an algorithm for testing when the lag correlation *k* between observations has died down, and derives a method that batches observations *k* apart;

(ii) Damerdji [1988], who determines consistent estimators of the time-average variance, which ensure the asymptotic validity of overlapping batch mean methods;

(iii) Schmeiser et al. [1990], who provide four test assumptions that ensure that observations are unbiased and converge meaningfully to the variances.

Law and Kelton [1984] suggest that using batch means with five or fewer batches for a fixed sample size gives coverages as good as any of the above methods, and is the simplest and least expensive to implement. However, Law and Carson [1979] determine that hundreds of batches may be needed in order to find the batch size that ensures uncorrelated batches. Schmeiser [1982] concludes that once an optimal batch size is determined that no more than 30 batches, and usually 10 batches, is sufficient for most simulation purposes to construct a confidence interval with good coverage and half-width properties. Good coverage still results when less than 10 batches are used, but at the expense of a significantly wider confidence interval.

2.1.3 Regenerative

The regenerative method involves identifying regeneration points within the simulation output where the underlying process probabilistically starts over. The observations between any two regeneration points (called a cycle) are considered to be independent of the other observations. A ratio estimator is developed for the mean response based on the ratio, average number of observations to the average cycle length. The regenerative method has the advantage of not having to deal with the problem of the initial transient, however, the simulation must have identifiable regeneration points and it must have enough regeneration points (greater than 5) to provide a valid confidence interval. Law and Kelton [1979] show how the performance of the regenerative method deteriorates when the number of regeneration points is too small.

Both Crane and Iglehart [1974a] and Fishman [1974] are credited with developing the regenerative approach for simulation output analysis. They both examine estimates of queueing performance such as average number of customers in a busy cycle, average length of a busy cycle, idle time, and waiting time. While Fishman [1978] consolidates most of his results in an excellent text, Crane and Iglehart [1974b] [1975a] [1975b] describe in a series of papers applications of the regenerative approach to Markov chains and to simulation models requiring approximations for regeneration points. Crane and Lemoine [1977] provide more details of simulation applications of the regenerative method. Iglehart [1975] [1976] [1978] provide applications of the regenerative method for statistical estimation other than for the mean. The jackknife procedure of estimating the variance is suggested if computational and storage capacities allow it. Finally, Iglehart and Shedler [1980] [1983] provide application of the regenerative method to analyzing queueing networks.

2.1.4 Time Series

Time series approaches seek to model the simulation output as AR processes of order p or as ARMA processes of order p and q. Since the estimates of the variance of time series models account for the correlation problem, confidence intervals are easily constructed. However, the problem of identifying the time series model that best fits a given output data set is a difficult one.

Fishman and Kiviat [1967] were the earliest to develop an AR approach for estimating the variance. Fishman [1971] used AR representations in order to determine the sample size necessary to get a confidence interval of desired precision. Fishman [1973] provides a procedure to determine the order p and the AR model coefficients. Schriber and Andrews [1984] extended Fishman's method to include determining the parameters of ARMA representation of simulation output.

Although the time series approaches provide the distinct advantage of inherently accounting for the correlation, their performance as a confidence interval technique has been dismal in general, especially when applied to models that are not specifically time series models. Law and Kelton [1984] found that the coverages of AR models, in general, were lower than expected.

2.1.5 Spectral

The spectral approaches are based on the idea of harmonic analysis that says that any time-varying output can be decomposed into a sum of independent sinusoids at various frequencies and amplitudes called its spectral density function. Furthermore, the variance of the output is a function of the sum of the squared amplitudes of the sinusoids. Therefore, all spectral approaches seek to determine the value of the spectral density function at zero frequency as an estimate of the output variance.

Duket and Pritsker [1978] were among the first to use the spectral estimate as a variance estimate for confidence intervals. However, empirical studies with M/M/1 queues show the variance estimates to be so large that the confidence coverage is grossly distorted.

Moeller and Welch [1977] develop an alternate approach of estimating the variance by fitting a low order (quadratic) polynomial to the sample spectral density, or periodogram. The method gives valid confidence intervals comparable to the batch mean method and even gives valid confidence intervals for cases where the batch mean method fails to determine the proper batch size to generate a confidence interval.

Heidelberger and Welch [1981a] improved upon the spectral estimation procedure by doing a polynomial regression of the log averaged periodogram. This procedure gives variance estimates that cause the coverage to be less than desired for small sample sizes.

Heidelberger and Welch [1981b] tested other procedures for improving the spectral estimate. Such procedures include polynomial smoothing at higher than quadratic degrees and the use of smoothing splines. These procedures, however, yield poorer confidence interval performance than the original spectral procedure.

2.1.6 Standardized Time Series

The standardized time-series method is based on transforming the simulation output to a Brownian bridge stochastic process. A derived central limit theorem for the Brownian bridge process yields the means by which a confidence interval is constructed (see Schruben [1983]).

Goldsman and Schruben [1984] determined that the standardized time-series confidence intervals strictly dominate the classical confidence intervals. Goldsman and Schruben [1990] developed asymptotically valid confidence interval estimators which are essentially weighted generalizations of Schruben's original method. Goldsman et al. [1990] improved upon the method by using weighted standardized time-series variance estimates. The performance of the confidence intervals result in the least asymptotic bias and excellent coverage.

2.2 State of Current Research

The state of the current research in the area of steady-state simulation output analysis is to develop sequential procedures for many of the aforementioned methods. Law [1983], however, makes the case for developing newer fixed-width confidence interval procedures similar to the replication and the batch means methods, which are both based on independent and identically distributed observations. Because of the simplicity and ease of implementing such procedures, simulation analysts use them more frequently than any of the sophisticated procedures, such as time series, spectral, and standardized time series approaches. Most analysts possess neither the time nor the expertise to use sophisticated procedures.

Therefore, research is ongoing and much needed for addressing the still unanswered problems of fixed-width confidence interval procedures. Such problems that remain to be solved include:

(1) determining the optimal batch size to ensure uncorrelated batches;

(2) developing consistent estimators of the covariance;

(3) deriving more powerful tests for correlation between batches;

(4) developing a small sample confidence interval methodology that yields good performance.

CHAPTER III

APPROACH

In multivariate regression analysis, much literature has been written on the problem of multi-collinearity between variables. Similar to the problems of auto-correlation in the univariate case, multi-collinearity distorts the variance without affecting the bias of estimators. This causes more confidence to be placed in the validity of confidence intervals than should be.

Several methods exist to handle the problem of multi-collinearity. One such method, the principal component method, seeks to obtain a more parsimonious representation of the variance components of the regression model. This representation of the regression model facilitates a more accurate interpretation of the regression estimates and the relationship between the estimates.

The principal component method orthogonally transforms the symmetric covariance structure of the regression model to a structure which contains no cross terms. The variance component of this structure can then be more readily determined. The orthogonal transformation has its basis in certain diagonalization theorems of matrix algebra. One such theorem is as follows:

Diagonalization Theorem: An n x n matrix S is diagonalizable if and only if it has n linearly independent eigenvectors e. S is similar to a diagonal matrix D, with $P^TSP = D$. The diagonal elements of D are the eigenvalues λ of S, while P is a matrix whose columns are the n linearly independent eigenvectors of S (p.261, Kolman [1988]).

Various corollaries give the conditions under which a symmetric matrix S is diagonalizable:

<u>Corollary</u>: If S has real and distinct eigenvalues, it is diagonalizable.

<u>Corollary</u>: If S is symmetric, all of its eigenvalues are real (not necessarily distinct).

<u>Corollary</u>: If S is symmetric, there exists an orthogonal matrix P that diagonalizes it (even if the eigenvalues of S are not distinct).

Ordinarily, a simulation output stream is assumed to be from a normal population with mean μ and variance σ^2, both scalers. However, simulation output is merely a realization of a time series and time series theorems are based on multinormal assumptions. It seems reasonable to consider simulation output under such assumptions and to investigate the potential for the principal component method to handle the problem of collinearity.

3.1 Problem Statement

Let a simulation output sample be defined by the sequence $\chi(\omega) = \{X_i(\omega), i=1,2,...,n\}$, where $\chi = \{X_i, i=1,...,n\}$ is the stochastic process represented by the simulation model. In general, the X_i's are seldom mutually independent. Most simulation output is correlated and the classical confidence interval,

$\bar{X} \pm t_{(n-1,1-\alpha/2)}\sqrt{\frac{s^2(n)}{n}}$, is inapplicable as a measure of the precision of the sample mean. For a correlated sample, the sample mean remains an unbiased estimate of the mean of the underlying process; however, the sample variance $s^2(n)$ becomes a biased estimate of the variance σ^2. The bias of the sample variance has the form: $1-2\sum_{j=1}^{n-1}\frac{(1-j/n)}{n-1}\rho_j$ where ρ_j is the correlation [Law and Kelton, 1982]. If there is positive correlation in the sample, the sample variance underestimates σ^2. This causes the confidence interval to appear to be tighter than it really is with resulting lower coverage probability due to the higher likelihood of more independent experiments having the true mean falling outside of the tighter intervals. Alternately, if there is negative correlation in the sample, $s^2(n)$ overestimates σ^2. This causes the confidence interval to be wider than usual with less than the desired precision obtained.

Most approaches to solving the problem of data correlation are based on the premise that some basically uncorrelated version of the data can be obtained so that classical confidence interval procedures then become applicable. The proposed research attempts to do the same by using a different approach with underlying theory taken from the principal component method for handling the multicollinearity problem in multivariate statistical analysis.

3.2 Solution Derivation

The simulation output stream, $X_1,\ldots X_n$, is assumed to be (either individually or batched) from a multi-normal population with mean vector $\bar{\mu}$ and covariance matrix Σ . For an arbitrarily selected maximum J, all X_J observations or batch means are assumed to be estimates of the population mean, i.e.

$$\bar{\mu} = \begin{bmatrix} E[X_1] \\ E[X_2] \\ \cdot \\ \cdot \\ \cdot \\ E[X_J] \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \cdot \\ \cdot \\ \cdot \\ \mu_J \end{bmatrix} \quad \textit{where } \mu_1 = \mu_2 = \ldots \mu_J$$

The covariance matrix Σ is assumed to be a symmetric, positive definite matrix with the form:

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1J} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2J} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \sigma_{J1} & \sigma_{J2} & \cdots & \sigma_{JJ} \end{bmatrix}$$

where $\sigma_{ij} = Var\, X_i$ for $i=j$ and $\sigma_{ij} = Cov(X_i, X_j)$ for $i \neq j$.

It can be shown (see p. 278, Kolman [1980]) that the eigenvalues λ_i , $i = 1, \ldots, J,$ of Σ are the diagonal elements of a diagonal matrix Λ that is formed by the orthogonal matrix V whose columns are the eigenvectors of

Σ , i.e.

$$\Lambda = V^T \Sigma V.$$

Since X is assumed to be distributed $N(\bar{\mu}, \Sigma)$, then $Y = V^T X$ will be distributed $N(V^T \bar{\mu}, \Lambda)$ and is hypothetically an uncorrelated version of the original output (see Chapter 5, Miller [1981]).

Suppose that the transformed data Y is distributed with mean $V^T \bar{\mu}$. Since the elements of $\bar{\mu}$ are equal,

$$\begin{aligned} V^T \bar{\mu} &= V^T (\mu_1, \ldots, \mu_J)^T \\ &= \mu V^T (1, \ldots, 1)^T \\ &= \mu (C_1, \ldots, C_J)^T \end{aligned}$$

where C_i is the *ith* row sum of V^T.

Therefore, Y_i is distributed $N(\mu C_i, \lambda_i)$, $i=1, \ldots, J$, and Y_i/C_i is distributed $N(\mu, \frac{\lambda_i}{C_i^2})$.

Here, Y_i/C_i are the NIID point estimates of μ . For interval estimation of μ , Y_i/C_i should be unbiased estimates and have minimum variance. If weights a_i exist such that $\sum_{i=1}^{J} a_i \frac{Y_i}{C_i}$ is an unbiased estimate and the variance of the unbiased estimate is minimized, then

$$\frac{\sum_{i=1}^{J} a_i \frac{Y_i}{C_i} - \mu}{\sqrt{Var[\sum_{i=1}^{J} a_i \frac{Y_i}{C_i}]}}$$

is distributed *N(0,1)*.

The proposed *100(1-α)%* confidence interval then follows as:

$$\sum_{i=1}^{J} a_i \frac{Y_i}{C_i} \pm Z_{1-\alpha} \sqrt{Var[\sum_{i=1}^{J} a_i \frac{Y_i}{C_i}]} \; .$$

The weights a_i are determined by the Lagrangian technique for nonlinear optimization to be $a_i = \frac{1/V_i}{\sum_{i=1}^{J} 1/V_i}$ where $V_i = \frac{\lambda_i}{C_i^2}$.

Simplifying yields a proposed confidence interval:

$$\sum_{i=1}^{J} a_i \frac{Y_i}{C_i} \pm Z_{1-\alpha} \sqrt{\sum_{i=1}^{J} a_i^2 \frac{\lambda_i}{C_i^2}} \; .$$

See Appendix A for the details of determining the weights a_i.

3.3 Experiment Methodology

The proposed method to test the procedure is to select some known analytical models. The mean of each model is known for any set of model parameters. Tactical issues are addressed such as simulation run length, number of replicates, and number of batches. The simulation of known analytical models is run long enough to ensure that steady state conditions are present. The number of batches of the simulation output is selected arbitrarily without regard for the batch size that would ensure uncorrelated batches.

To ensure that the results are comprehensive and comparable with other previous results in the area of confidence interval methodologies, the proposed confidence interval procedure is tested against a standard set of performance measures under a variety of model parameter, sample size, and batch size factor combinations. No statistical tests of significance are performed on the results since the reporting of the results follow a suggested and well-developed conceptual framework for reporting research in the analysis of simulation output (see Schriber and Andrews [1981]).

3.3.1 Conceptual Framework

The purpose of a conceptual framework for conducting empirical tests on and reporting the results of a proposed confidence interval procedure is to ensure the required uniformity needed to compare the results with the results of other previously conceived and tested confidence interval procedures. In general, the conceptual framework consists of running several replicates of a theoretical output process (TOP) and constructing the proposed confidence interval procedure (CIP) for each replicate. This is repeated for the TOP at various model parameters. Similarly, this is repeated for at least one other TOP. The runs are examined for performance on specific measures of effectiveness (MOE).

TOP's are classified into five major areas:

(1) independent and identically distributed processes, such as normally

distributed random variates;

(2) groupwise independent processes, such as an uncorrelated batch means process;

(3) regenerative processes, such as Markov chains;

(4) time series models, such as ARMA processes;

(5) other output processes with known theoretical properties, such as the M/M/1 queue.

These five classes of TOP's serve many purposes. The primary purpose of a TOP is to provide the output sequence to be used to test the effectiveness of the proposed confidence interval procedure. In this research, three TOP's are used to generate output sequences: the M/M/1 queue, an AR(1) model, and an ARMA(1,1) model.

Alternately, TOP's provide the theoretical framework to suggest newer and improved CIP's. For example, the ARMA process have the Box-Jenkins method of analysis to identify the model parameter estimates needed to construct a confidence interval which is tailor-made for ARMA processes.

Finally, TOP's serve as a basis of comparison for unknown output sequences. Many real world simulation output sequences have eventually been identified as similar in behavior to any one of or a combination of the above five broad classes of TOP's. This identification then suggests to the simulation analyst which CIP is applicable for the unknown output process.

The MOE's provide a complete set of standards to judge and compare the performance of a given CIP on various TOP's. Five specific MOE's proposed under the conceptual framework are:

(1) correct identification of model-specific parameters;

(2) coverage function;

(3) the standardized standard deviation of the standard error;

(4) relative half-width and coverage as a function of sample size;

(5) the extent of gross misinterpretation of model-specific behavior.

The first MOE confirms the credibility of using a proposed CIP for a certain TOP. For example, a CIP is credible for an AR(p) process if it can correctly identify the order p of the AR process. For a batch means process, a proposed CIP should be able to identify the correct batch size to ensure uncorrelated batches.

The second MOE involves a comparison of the actual coverage at various confidence coefficient levels versus a theoretical coverage, which follows a $U(0,1)$ distribution. This comparison is done graphically by noting how far from a 45 degree line the actual coverage empirical cumulative distribution function varies. This comparison is tested statistically by doing a chi-square or an empirical discrete test for uniformity.

The third MOE is a measure of the stability of the CIP. Since the standard error is the primary measure of how wide the confidence intervals are, a small standard deviation of the standard error would indicate fairly invariant confidence interval half-widths. In this research, a measure of the standard deviation of the half-width is used in lieu of the third MOE. The measure is standardized by dividing by the average half-width. Proportionally, the standard error and the half-width are the same and provide the same measure of stability.

The fourth MOE primarily has the purpose of comparing the half-width and the coverage performance of the CIP for a variety of sample sizes. This helps the simulation analyst to determine at which sample size the simulation is to be run in order to achieve the desired level of precision and the expected coverage. The ideal performance on this MOE is to have the half-width decreasing as the sample size increases and the coverage increasing to the nominal confidence level with increasing sample size.

The fifth MOE reports the percentage of runs that the CIP grossly erred, usually with model-specific parameter selection. For example, this research tracks this MOE by recording the number of times the proposed CIP failed to give uncorrelated batches. In other cases, such as with time series, this MOE is tracked by determining the number of times an ARMA CIP incorrectly selected model parameters p and q. Ideally, this measure should be very small.

3.3.2 Models and Parameters

Three TOP's used in this research include an M/M/1 queueing system, an AR(1) time series model and an ARMA(1,1) time series model. All of the models exhibit a variety of correlation behavior from low levels of correlation to oscillatory and to high levels of correlation. Yet, all of the models have a known analytical mean regardless of the correlation behavior. Each model has been studied and used extensively by past simulation practitioners in testing CIP's; therefore, some historical basis for performance comparisons already exists.

First, the M/M/1 queueing system was simulated by using a simply written SIMAN simulation language routine for various loading factors, ρ. Loading factors of .5, .8, and .9, respectively, indicate increasing levels of correlation between observations. The observations that were collected are the customer waiting times in the system, where the theoretical average waiting time is analytically determined from the expression $\frac{\rho}{1-\rho}$. Therefore, the theoretical mean waiting time, which is the mean response that the analyst is interested in estimating, is, respectively, 1, 4, and 9 for loading factors of .5, .8, and .9.

Since this research did not investigate the problem of the initial transient, the simulation model must warm up for a period of time prior to data actually being collected. This warm-up period was determined by pilot runs to detect how long the simulation must run before initial conditions no longer give a biased estimate of the known theoretical mean. For loading factors of .5, .8, and .9, the simulation warm-up period was determined to be 75 thousand, 100 thousand, and 165 thousand simulated time units, respectively.

The AR(1) output was simulated by a FORTRAN-based, time series analysis software called Timeslab (Newton [1988]). Timeslab creates realizations of a first order stationary autoregressive process given by $X_t - \mu - \phi(X_{t-1} - \mu) = \epsilon_t$. The mean and variance of the generated process are both equal to 100. The lag one correlation, ρ_1, is ϕ, and the lag h correlation,ρ_h, is $\phi^{h-1}\rho_1$, for $h = 2,...,J$. The model parameter, ϕ, was selected to be .5 and .9 in order to represent both low and high levels of positive correlation in the data.

The ARMA(1,1) output, similarly, was generated by Timeslab. Timeslab creates realizations of the stationary first order autoregressive moving average process given by: $X_t - \mu - \phi_1(X_{t-1} - \mu) = \epsilon_t - \theta_1\epsilon_{t-1}$. The mean and the variance are both again 100. The lag one correlation is determined as $\rho_1 = \dfrac{(1 - \phi_1\theta_1)(\phi_1 - \theta_1)}{(1 + \theta_1^2 - 2\phi_1\theta_1)}$ and the lag h correlations are $\rho_h = \phi_1^{h-1}\rho_1$. The model parameters are $\phi_1 = -.6$ and $\theta_1 = .6$. The parameters were chosen to satisfy the requirements of an ARMA process with an oscillatory autocorrelation function.

3.3.3 Performance Measures

The major performance measures used in this research to judge the adequacy of the proposed CIP are the nominal coverage, the 90% coverage interval, the standardized standard deviation of the half-width, the average mean square error, and the number of replicates biased high/low. Each performance measure is chosen for a specific reason. The details of how each measure is determined follows.

First, the nominal coverage tells how many replicates have confidence intervals encompassing the known theoretical mean. The nominal coverage should ideally be close to the stated $100(1-\alpha)\%$ confidence level of 90%. The 90% coverage interval is a 90% confidence interval on the coverage and is a means by which the validity of the confidence interval is determined. If the coverage interval contains the confidence level of 90%, then the confidence interval procedure is considered to be valid for the given model set at the given model parameters. For this research, nominal coverages of .84 or higher will have 90% coverage intervals that encompass the desired confidence level. Therefore, a confidence interval will be considered valid if its nominal coverage probability is .84 or higher.

Second, the average half-width is determined from summing the half-widths, $Z_{1-\alpha}\sigma_{\bar{x}}$, for each replicate where $Z_{(1-\alpha)}$ is the $1-\alpha^{th}$ entry of the standard normal table and $\sigma_{\bar{x}}$ is the estimate of the standard error obtained from the proposed CIP. In order to get the standard deviation of the half-width, the squared half-width is, likewise, summed for each replicate. Once all replicates are run, the average half-width and the standard deviation of the half-width are easily determined. The standardized estimate of the standard deviation of the half-width is obtained by dividing the standard deviation of the half-width by the average half-width. The smaller this number, the less variable the confidence interval half-widths and the more stable the CIP.

Third, the mean square error (MSE) is defined as the sum of the variance and the squared bias. The variance is obtained as an estimate from the proposed

CIP. The bias is obtained from the difference between the proposed CIP's estimate of the mean and the known theoretical mean. For each replicate, the MSE is summed. Once the replicates are completed, the average MSE is easily determined. Ideally, the MSE should be as low as possible.

The MSE as a strong statistical measure of estimator consistency is suggested by Fishman [1978] for judging confidence interval adequacy and is used extensively in simulation transient analysis by Donnelly and Shannon [1981]. The MSE clearly shows the contributions to sampling error that can be attributed to "closeness" of the estimator (the bias) and "variability" of the estimator (the variance). Clearly, a desired unbiased estimator for confidence interval construction would have the MSE equal to the variance.

Finally, the number of replicates biased high/low is a sufficient measure suggested by Jennings [1987] to judge whether the confidence interval procedure is "unbiased" or whether it will be equally likely to be above or below the true mean when it fails to encompass the mean. This measure is easily obtained with hardly any extra work than what is required to determine the coverage. If the confidence interval for each replicate has the lower limit higher than the true mean then it is said to be biased high and a counter for the number of confidence intervals biased high is incremented by one. Likewise, if the confidence interval for each replicate has the upper limit lower than the true mean then it is biased low and the appropriate counter is incremented. Ideally, the number of biased high and low should be the same since a symmetric distribution is assumed in forming the confidence interval.

3.3.4 Tactical Issues

Various sample sizes are used for all of the models tested by the proposed CIP. In order to judge the effect that sample size has on the performance measures, small sample sizes are initially run and doubled successively for subsequent runs. For example, the M/M/1 queueing model set at ρ =.5 is run for sample sizes of n = 320, 640, 1280, 2560, 5120, and 10240 customer waiting times. These samples sizes are the common sample sizes used by other practitioners in testing confidence interval procedures and easily allow comparisons of performance with the other procedures.

In order to determine the effect that the batch size has on the performance measures, batch sizes corresponding to 10, 20, and 40 batches from a fixed sample size were taken for each model set at each model parameter. This arbitrary method of batching also allows for showing the effect of correlation on the performance measures since smaller batch sizes will tend to be more highly correlated than larger ones.

In order to obtain the information for the coverage function, the models are run at the largest possible sample size to ensure as close to asymptotic conditions as possible. Then, replicates are run at various *100(1-α)%* confidence levels with the nominal coverage being the only desired performance measure that is needed for the coverage function.

Overall, regardless of sample size, batch size, and model parameter factor combinations, 100 replicates of each factor combination were run with 100 confidence intervals being constructed under the proposed confidence interval procedure. After the replicates were completed, the desired performance measures were calculated.

Now, the adequacy of the proposed confidence interval procedure was compared to one other confidence interval procedure, the generic batch means procedure. The proposed confidence interval procedure uses an arbitrary batch size while a true batch means procedure uses a pre-determined batch size that ensures uncorrelated batches. This pre-determined batch size is obtained by increasing the simulation run-length and conducting statistical tests for correlation on batches of a selected size until an optimal batch size is reached. The difficulty with this method of obtaining the optimal batch size is that it may require more observations than the largest stated sample size that was used for experimentation. To ensure that both CIP's are compared based on the same observations, the batch means procedure is run at the arbitrary batch sizes corresponding to 10, 20, and 40 batches per sample size. This has an effect, however, of having some confidence intervals inappropriately constructed since the batch size will in some cases not be large enough to ensure uncorrelated batches. To account for this discrepancy, the number of such confidence intervals that are inappropriately constructed because of correlated batches is tabulated for both procedures.

3.3.5 Research Plan

The major tasks required by this research approach are as follows:

(1) Generate a simulation output sample of size n, $X_1,\ldots,X_n$, from a given factor combination. Determine the J batch means, $\bar{X}_1,\ldots,\bar{X}_J$. Determine the batch mean confidence interval and calculate the appropriate performance measures.

(2) Form the J-dimensional covariance matrix Σ consisting of $\hat{C}_j$ estimates, where $\hat{C}_j=\sum_{i=1}^{J-j}(\bar{X}_i-\bar{X})(\bar{X}_{i+j}-\bar{X})$.

(3) Determine the eigenvalues and eigenvectors of Σ. Construct the orthogonal matrix T.

(4) Perform the orthogonal transformation, $Y_j=T^T\bar{X}_j$ and the matrix algebra manipulations to get an estimate of the standard error of the mean for the proposed confidence interval procedure. Calculate the appropriate performance measures.

(5) Repeat (1) through (4) for every factor combination.

(6) Tabulate and draw conclusions on the results.

(7) Comment on the computational environment and procedures used to support the research which includes:

(i) describing the computer environment;

(ii) explaining the software used.

CHAPTER IV

EMPIRICAL RESULTS AND DISCUSSION

To compare the performance of the proposed procedure, the principal component (PC) method, an M/M/1 queueing model with utilization rates of .5, .8, and .9 with known theoretical mean customer flow times, an AR(1) time series model with a known mean of 100 and positive lag-1 correlations of .5 and .9 and an ARMA(1,1) time series model with a known mean of 100 and an oscillatory correlation structure were used. For all models, 100 replicates at various sample sizes, *n*, which are each divided into various number of batches, *nb*, were run. The 90% confidence intervals that are constructed by a fixed-width batch means (BM) procedure were compared to those that are constructed by the PC method. The measures of performance that were compared include the nominal coverage probability, the standardized standard deviation of the half-width, the mean square error, and the number of confidence intervals biased high and low. The descriptions of the models, parameters, and performance measures were detailed previously in Chapter III, Sections 3.3.1 - 3.3.3.

The results of the experiments are tabulated in respective tables in this chapter. Since coverage probability is probably the performance measure of most interest to the simulation analyst, it is discussed first and in far greater detail than the other performance measures. Discussions on the half-width, the average mean square error and the bias performance of the methods follow the discussion of coverage for each model set at a particular parameter.

First, a guide to interpreting the tables is needed. The first two tables of this chapter display the results of pilot studies. The remaining tables display the results of the analytical models that were used to test the performance of the principal component (PC) method and the batch means (BM) method. The tables are somewhat self-explanatory, but the following table key is provided to assist in interpreting the tables:

PC	- principal component
BM	- batch means
ρ	- correlation level
n	- fixed sample size
nb	- number of batches
Cov	- nominal coverage
k	- number of replicates that inappropriately used correlated batches
HW	- standardized standard deviation of average half-width
MSE	- average mean square error
Bias	- number of confidence intervals that exceeded/fell short of covering the mean

Table Key

4.1 Pilot Studies

Prior to full experimentation, pilot studies were done in order to determine the large sample performance of the PC procedure. Using the results of Schmeiser [1982], various batches were taken from a sample size of 10240 observations for an M/M/1 queue at each utilization rate. In addition, various batches from a sample size of 2560 observations were taken from an AR(1) model at various correlation levels and an ARMA (1,1) model. A Timeslab macro was written (see Appendix B) in order to perform the analytical steps of covariance estimation, matrix diagonalization, matrix algebra manipulations, interval estimation, and performance measure calculations.

The results of Table 4.1 show that the coverage of the PC method was poor compared to the coverage of the BM procedure. The coverage seemed to peak at about 55%, which was far short of the desired 90% level. As expected, the half-width performance and the bias information of the PC method were comparable to the BM procedure since the PC method is a linear unbiased estimate with minimum variance.

A possible explanation for the undesirable poor coverages of the PC method is the extent to which the minimization scheme of linear unbiased estimation determined a variance estimate that grossly underestimated the true process variance. For example, the half-width of the PC method was about half the size of that of the BM procedure. Therefore, a modification to the procedure was done

Table 4.1

Pilot Study Results
R = 100 Replicates

			PC				BM			
	ρ	nb	Cov (k)	HW	MSE	Bias	Cov (k)	HW	MSE	Bias
MM1	.5	10	.52 (10)	.456	.001	17/31	.88 (9)	.225	.009	3/9
		20	.54 (5)	.451	.001	17/29	.90 (0)	.180	.018	2/8
		40	.55 (0)	.450	.001	16/29	.90 (0)	.165	.035	2/8
	.8	10	.54 (8)	.490	.164	22/24	.89 (6)	.357	1.370	3/8
		20	.52 (2)	.487	.167	23/25	.88 (2)	.310	2.415	3/9
		40	.52 (0)	.488	.162	23/25	.82 (0)	.277	4.181	5/13
	.9	10	.53 (13)	.721	3.278	19/28	.79 (3)	.487	23.15	2/19
		20	.53 (7)	.713	3.251	18/29	.78 (4)	.416	36.34	2/20
		40	.53 (2)	.709	3.244	19/28	.73 (15)	.346	50.47	6/21
AR1	.5	10	.54 (12)	.555	.217	21/25	.86 (10)	.241	1.82	7/7
		20	.55 (4)	.549	.216	21/24	.90 (0)	.162	3.43	5/5
		40	.54 (0)	.549	.216	22/24	.87 (0)	.113	6.51	7/6
	.9	10	.54 (4)	.554	5.544	22/24	.84 (7)	.240	44.04	8/8
		20	.54 (2)	.544	5.502	22/24	.86 (1)	.166	80.28	6/8
		40	.54 (2)	.545	5.439	22/24	.86 (0)	.109	142.1	8/6
ARMA		10	.49 (10)	.570	.943	27/24	.84 (8)	.197	7.049	10/6
		20	.47 (2)	.575	.926	28/25	.84 (1)	.156	12.79	8/8
		40	.48 (1)	.575	.913	28/24	.85 (0)	.115	25.19	7/8

in order to achieve the desired coverage.

Recall from Section 3.2, that the batched simulation output stream, $X_i, i=1,...,J$, is orthogonally transformed into an uncorrelated version, Y_i/C_i, $i=1,...,J$. Because this version of the original data was highly variable, the sample mean of the data was virtually useless as an estimate of the mean. Therefore, the linear unbiased estimate of the mean was determined as a suitable point estimate of the mean. Theoretically, the linear unbiased estimate of the mean has minimum variance determined by the previously mentioned minimization scheme, which is described in detail in Appendix A. This minimum variance estimate severely underestimated the true process variance and, consequently, the coverage. What is needed is another estimate of the process variance that would yield suitable coverage.

Consider estimate a_iY_i/C_i, which is distributed $N(\mu, \lambda_i/C_i^2)$. As determined in Appendix A, the minimum variance of this estimate is $a_i^2 \lambda_i/C_i^2$. This estimate is seen to be too small for adequate confidence interval coverage. One way to increase the variance estimate is to decrease the size of the term in the denominator. This is possible if C_i^2 is replaced by its mean value, which is determined to be 1. This allows the variance estimate to increase to $a_i^2 \lambda_i$ (See Appendix A).

Table 4.2 shows how pilot studies of this adjusted procedure resulted in comparable performance to the BM procedure. The coverage of the adjusted procedure exceeded that of the BM procedure for all model/parameter/number of batches factor combinations. It did so however with somewhat larger half-widths, yet

Table 4.2

Pilot Study Results (Modified)
R = 100 Replicates

			PC				BM			
	ρ	nb	Cov (k)	HW	MSE	Bias	Cov (k)	HW	MSE	Bias
MM1	.5	10	.90 (0)	.427	.003	3/7	.88 (9)	.225	.009	3/9
		20	.96 (0)	.375	.004	1/3	.90 (0)	.180	.018	2/8
		40	.98 (3)	.379	.006	0/2	.90 (0)	.165	.035	2/8
	.8	10	.93 (0)	.465	.402	2/5	.89 (6)	.357	1.370	3/8
		20	.96 (1)	.438	.582	1/3	.88 (2)	.310	2.415	3/9
		40	.99 (4)	.425	.857	0/1	.82 (0)	.277	4.181	5/13
	.9	10	.83 (1)	.675	8.555	3/14	.79 (3)	.487	23.15	2/19
		20	.89 (5)	.627	10.99	2/9	.78 (4)	.416	36.34	2/20
		40	.95 (4)	.669	18.22	1/4	.73 (15)	.346	50.47	6/21
AR1	.5	10	.88 (1)	.494	.528	9/3	.86 (10)	.241	1.82	7/7
		20	.94 (5)	.465	.766	4/2	.90 (0)	.162	3.43	5/5
		40	.97 (4)	.451	1.291	3/0	.87 (0)	.113	6.51	7/6
	.9	10	.88 (2)	.486	13.09	9/3	.84 (7)	.240	44.04	8/8
		20	.91 (0)	.449	19.06	7/2	.86 (1)	.166	80.28	6/8
		40	.97 (3)	.459	29.97	3/0	.86 (0)	.109	142.1	8/6
ARMA		10	.86 (0)	.498	2.158	10/4	.84 (8)	.197	7.049	10/6
		20	.91 (6)	.505	3.120	6/3	.84 (1)	.156	12.79	8/8
		40	.98 (2)	.468	4.851	2/0	.85 (0)	.115	25.19	7/8

with surprisingly smaller average mean square error. Amazingly, the larger half-width measures that were provided by the PC method are not due to larger variance estimates than the variance estimates that are produced by the BM method. In fact, in all cases, the average variance that is determined by the BM method exceeded that determined by the PC method by more than two-fold. However, the standard error estimate from the BM method is determined by dividing the variance estimate by the number of batches, *nb*. The standard error for the PC method is not likewise reduced. The remainder of this chapter will discuss the results of more detailed experimentation of the procedure.

4.2 M/M/1 Queue

The empirical results of the proposed procedure on an M/M/1 queueing system are shown in Tables 4.3 - 4.5. In general, the coverage of the PC method exceeded the coverage of the BM method in all cases. Surprisingly, the coverage was extremely good for small sample sizes of lightly utilized queues. For the smallest sample size, $n=320$, Table 4.3 shows the coverages of the PC method as .87, .90, and .95, respectively, for $nb=10$, 20, and 40 batches as compared to BM coverages of .83, .81, and .71. If the sample size is too small for more highly utilized queues, the coverage is poor. Tables 4.4 and 4.5 show that only for sample sizes greater than 2560 observations was adequate coverage achieved by the PC method whereas the BM method failed to provide adequate coverage even at the largest sample size that was used.

Table 4.3

M/M/1 Queue with $\rho = .5$
R = 100 Replications

		PC				BM			
n	nb	Cov (k)	HW	MSE	Bias	Cov (k)	HW	MSE	Bias
320	10	.87 (2)	.7238	.0978	4/9	.83 (4)	.5669	.3183	3/14
	20	.90 (3)	.6586	.1292	4/6	.81 (3)	.4244	.4146	2/17
	40	.95 (6)	.6913	.2050	2/3	.71 (11)	.3434	.5845	7/22
640	10	.85 (1)	.5437	.0401	2/13	.79 (4)	.4405	.1376	3/18
	20	.91 (0)	.6448	.0670	0/9	.81 (0)	.4245	.2669	2/17
	40	.94 (1)	.5231	.0860	0/6	.78 (1)	.3583	.4139	4/18
1280	10	.85 (0)	.5182	.0240	3/12	.85 (4)	.3483	.0791	2/13
	20	.94 (4)	.5425	.0331	0/6	.85 (1)	.3638	.1358	0/15
	40	.97 (4)	.5557	.0508	0/3	.84 (0)	.3046	.2363	0/16
2560	10	.92 (2)	.4791	.0121	2/6	.84 (4)	.2917	.0380	5/11
	20	.93 (1)	.4870	.0167	2/5	.84 (1)	.3083	.0714	5/11
	40	.98 (2)	.4698	.0257	0/2	.90 (0)	.2689	.1359	1/9
5120	10	.88 (0)	.4223	.0054	3/9	.87 (6)	.2878	.0195	3/10
	20	.97 (0)	.4135	.0082	0/3	.89 (1)	.2403	.0364	2/9
	40	.98 (3)	.3980	.0123	0/2	.88 (0)	.2027	.0676	3/9
10240	10	.90 (0)	.4272	.0026	3/7	.88 (9)	.2246	.0094	3/9
	20	.96 (0)	.3750	.0038	1/3	.90 (0)	.1801	.0178	2/8
	40	.98 (3)	.3792	.0062	0/2	.90 (0)	.1649	.0348	2/8

Table 4.4

M/M/1 Queue with ρ = .8
R = 100 Replications

		PC				BM			
n	nb	Cov (k)	HW	MSE	Bias	Cov (k)	HW	MSE	Bias
320	10	.74 (1)	.8808	6.832	5/21	.59 (14)	.6232	12.63	8/33
	20	.76 (1)	.9278	8.739	3/21	.52 (42)	.5393	14.03	12/36
	40	.82 (9)	.8140	8.184	0/18	.43 (73)	.4179	10.98	7/50
640	10	.77 (2)	.8428	5.167	3/20	.60 (10)	.6690	10.93	6/34
	20	.83 (1)	.8522	6.498	1/16	.49 (22)	.5668	12.96	14/37
	40	.91 (6)	.8630	6.684	1/8	.46 (49)	.3836	11.05	14/40
1280	10	.82 (1)	.7091	2.721	3/15	.75 (1)	.5714	6.80	4/21
	20	.90 (3)	.6725	3.493	0/10	.73 (12)	.4685	9.13	6/21
	40	.89 (2)	.9856	6.308	0/11	.58 (29)	.4523	11.00	8/34
2560	10	.92 (0)	.6189	1.643	0/8	.83 (3)	.4784	4.44	1/16
	20	.97 (2)	.5625	2.076	0/3	.81 (5)	.3915	6.67	5/14
	40	.97 (5)	.5850	3.139	0/3	.72 (15)	.3296	9.13	6/22
5120	10	.85 (0)	.5760	.786	0/15	.78 (2)	.4639	2.52	2/20
	20	.88 (3)	.5734	1.129	0/12	.78 (2)	.4093	4.20	5/17
	40	.95 (3)	.5223	1.629	0/5	.77 (2)	.3584	6.95	5/18
10240	10	.93 (0)	.4646	.402	2/5	.89 (6)	.3572	1.37	3/8
	20	.96 (1)	.4375	.582	1/3	.88 (2)	.3099	2.41	3/9
	40	.99 (4)	.4253	.857	0/1	.82 (0)	.2773	4.18	5/13

Table 4.5

M/M/1 Queue with $\rho = .9$
R = 100 Replications

		PC				BM			
n	nb	Cov (k)	HW	MSE	Bias	Cov (k)	HW	MSE	Bias
320	10	.48 (0)	.8596	67.62	12/40	.35 (29)	.6222	86.60	19/46
	20	.61 (1)	.7367	51.23	7/32	.28 (71)	.4387	61.96	15/57
	40	.61 (6)	.7962	106.9	8/31	.25 (94)	.4106	63.14	17/58
640	10	.54 (0)	.9631	54.78	7/39	.33 (19)	.6745	77.16	17/50
	20	.67 (1)	.8959	56.79	3/30	.35 (51)	.5010	71.90	12/53
	40	.72 (4)	.9170	74.15	3/25	.23 (87)	.4594	74.15	15/62
1280	10	.65 (5)	.9569	58.58	6/29	.51 (16)	.6953	87.57	13/36
	20	.63 (0)	.9260	47.99	2/35	.40 (37)	.5658	69.60	12/48
	40	.71 (6)	.9591	66.88	1/28	.35 (74)	.5087	73.68	14/51
2560	10	.74 (0)	.8026	32.24	3/23	.59 (8)	.6240	65.30	11/30
	20	.84 (2)	.9296	35.39	0/16	.53 (23)	.5375	63.72	12/35
	40	.87 (7)	.9607	55.79	0/13	.40 (57)	.4739	70.89	18/42
5120	10	.80 (5)	.7520	15.62	1/19	.68 (5)	.5708	39.13	3/29
	20	.88 (0)	.7059	20.06	0/12	.65 (11)	.4851	52.27	6/29
	40	.92 (3)	.7705	30.75	0/8	.58 (35)	.4103	63.35	11/31
10240	10	.83 (1)	.6751	8.55	3/14	.79 (3)	.4865	23.15	2/19
	20	.89 (5)	.6267	11.00	2/9	.78 (4)	.4159	36.34	2/20
	40	.95 (4)	.6687	18.22	1/4	.73 (15)	.3457	50.47	6/21

The PC method achieved such excellent coverage because the half-widths of the confidence intervals that are constructed by the PC method were larger than those constructed by the BM method, especially at small sizes. The half-width measures of the PC method were about twice as large as those of the BM method. For example, Table 4.3 shows for $n = 320$ observations the half-width measures for the PC method as .7238,.6586, and .6913 as compared to half-width measures of the BM method of .5669, .4244, and .3434. For highly utilized queues, the difference between the half-width measures of the two methods was less significant.

In all cases, the average MSE for the PC method was smaller than that of the BM method. The difference became less marked as the utilization level of the queue increased. For lightly utilized queues, the average MSE of the BM method was from two to three times larger than that of the PC method. For highly utilized queues, it was usually less than twice the size of the MSE measure of the PC method.

Finally, the bias distribution of failed confidence intervals for both procedures was asymmetrical. Both methods were biased low, which means that confidence intervals that failed to cover the mean were more likely to fall short of rather than exceed the mean.

4.3 AR(1) Time Series Model

The empirical results for the AR(1) time series model are shown in Tables 4.6 and 4.7. The coverage of the PC method greatly exceeded the coverage of the

Table 4.6

AR(1) Time Series with ρ = .5
R = 100 Replications

		PC				BM			
n	nb	Cov (k)	HW	MSE	Bias	Cov (k)	HW	MSE	Bias
320	10	.89 (1)	.4176	4.008	6/5	.81 (3)	.2259	12.69	11/8
	20	.93 (2)	.4260	5.605	4/3	.79 (1)	.1518	23.91	12/9
	40	.97 (4)	.4051	8.612	2/1	.79 (0)	.1105	41.77	13/8
640	10	.92 (1)	.3941	2.276	3/5	.88 (11)	.2098	7.16	8/4
	20	.97 (1)	.3808	3.166	1/2	.85 (1)	.1553	13.15	10/5
	40	.98 (2)	.3766	4.403	1/1	.84 (0)	.1146	23.53	10/6
1280	10	.92 (1)	.4735	1.088	5/3	.88 (8)	.2427	3.58	9/3
	20	.96 (0)	.4344	1.537	2/2	.89 (3)	.1696	6.57	9/2
	40	.97 (3)	.4510	2.559	2/1	.94 (0)	.1266	12.30	6/0
2560	10	.88 (1)	.4938	.528	9/3	.86 (10)	.2411	1.82	7/7
	20	.94 (5)	.4647	.766	4/2	.90 (0)	.1624	3.43	5/5
	40	.97 (4)	.4505	1.291	3/0	.87 (0)	.1128	6.51	7/6

Table 4.7

AR(1) Time Series with ρ = .9
R = 100 Replications

		PC				BM			
n	nb	Cov (k)	HW	MSE	Bias	Cov (k)	HW	MSE	Bias
320	10	.87 (3)	.4462	91.9	7/6	.73 (4)	.2521	232.5	15/12
	20	.91 (0)	.4456	123.3	5/4	.70 (7)	.1778	339.1	16/14
	40	.96 (6)	.4880	176.6	2/2	.57 (60)	.1498	412.2	23/20
640	10	.90 (0)	.4130	53.7	4/6	.83 (8)	.2108	154.4	11/6
	20	.94 (3)	.4035	73.3	2/4	.78 (2)	.1637	238.2	13/9
	40	.97 (4)	.4016	104.4	1/2	.68 (11)	.1305	334.9	18/14
1280	10	.91 (0)	.4686	26.4	6/3	.88 (8)	.2445	85.2	8/4
	20	.95 (4)	.4625	38.0	3/2	.87 (3)	.1769	143.7	9/4
	40	.98 (4)	.4547	57.9	1/1	.85 (0)	.1320	234.3	11/4
2560	10	.88 (2)	.4862	13.1	9/3	.84 (7)	.2404	44.0	8/8
	20	.91 (0)	.4487	19.1	7/2	.86 (1)	.1664	80.3	6/8
	40	.97 (3)	.4593	30.0	3/0	.86 (0)	.1093	142.1	8/6

BM method at all sample sizes and at all batch sizes. In fact, the coverage of the PC method exceeded the *desired* confidence level of .90 for virtually all cases. By comparison, the BM method failed to achieve the desired nominal coverage if the sample size was too small and rarely attained the desired confidence level of .90 even for the largest sample sizes.

The PC method attained such excellent coverage due to the tendency of the procedure to produce confidence intervals with larger half-widths. Table 4.6 shows that for low correlation, the half-width measure of the PC method was about twice as large as that of the BM method. For high correlation, Table 4.7 shows that the difference was less significant with the half-width measure of the PC method being less than twice as large as that of the BM method.

The average MSE for the AR(1) model shows similar results as with the M/M/1 model. The average MSE of the PC method was smaller than the average MSE of the BM method in all cases, about one-third the size of the BM method. However, the bias distribution of failed confidence intervals was asymmetrical for the PC method and symmetrical for the BM method, which is different from the M/M/1 result. Failed confidence intervals of the PC method tended to exceed the mean rather than fall short of it.

4.4 ARMA (1,1) Time Series Model

The empirical results of the ARMA(1,1) model are shown in Table 4.8. The coverage of the PC method greatly exceeded the coverage of the BM method. The

Table 4.8

ARMA(1,1) Time Series
R = 100 Replications

		PC				BM			
n	nb	Cov (k)	HW	MSE	Bias	Cov (k)	HW	MSE	Bias
320	10	.87 (1)	.4872	16.26	7/6	.81 (9)	.2625	50.30	11/8
	20	.95 (1)	.4612	22.85	3/2	.82 (1)	.1616	90.49	11/7
	40	.98 (3)	.4459	34.96	1/1	.82 (0)	.1219	154.0	11/7
640	10	.90 (2)	.3746	7.55	7/3	.83 (9)	.2381	27.14	13/4
	20	.96 (4)	.3588	10.42	2/2	.87 (1)	.1688	50.34	10/3
	40	1.00 (2)	.3378	15.52	0/0	.88 (0)	.1135	91.43	9/3
1280	10	.90 (3)	.4258	4.11	7/3	.86 (10)	.2476	13.76	10/4
	20	.96 (5)	.3669	5.54	3/1	.86 (2)	.1705	25.77	10/4
	40	.98 (2)	.3939	9.02	1/1	.87 (0)	.1180	49.09	10/3
2560	10	.86 (0)	.4982	2.16	10/4	.84 (8)	.1973	7.05	10/6
	20	.91 (6)	.5053	3.12	6/3	.84 (1)	.1558	12.79	8/8
	40	.98 (2)	.4679	4.85	2/0	.85 (0)	.1151	25.19	7/8

coverages of the PC method ranged from .86 to .98; therefore, the desired nominal coverage was attained at all sample sizes. Conversely, the coverages of the BM method ranged from .81 to .88. The desired nominal coverage was obtained only after sample sizes of 640 observations although the 90% confidence level was never attained.

Similar to the results of previous models, the excellent coverage of the PC method was obtained because of the larger half-widths of constructed confidence intervals. The half-width measure of the PC method was about twice as large as that of the BM method. The half-width measures of the PC method ranged from .3378 to .5053 compared to those of the BM method, which ranged from .1135 to .2625.

The average MSE for the PC method was significantly smaller than that of the BM method, especially for smaller batch sizes corresponding to 40 batches from a fixed sample size. This is due to a significant difference between the variance estimates of the two methods. The variance estimate that was obtained by the BM method is about six times as large as the variance estimate that was obtained by the PC method.

The bias distributions of both methods tended to be slightly asymmetrical. At large sample sizes, the confidence interval constructed by the PC method was more likely to exceed covering the true mean when it failed to cover the mean. However, at smaller sample sizes, the confidence interval constructed by the BM method was more likely to exceed covering the mean when it failed to cover the true mean.

4.5 Factor Levels

This section discusses the effect that different sample sizes and batch sizes has on coverage and half-width. It is well documented for the BM procedure that as the sample size increases, a more accurate and less variable estimate of the true mean is obtained. This results in less biased and less variable estimates, which improves the coverage performance of constructed confidence intervals at a smaller half-width measure and at a smaller average MSE.

As the batch size increases, the effect of correlated batches lessens. This causes the variance estimate to be a more accurate estimate for confidence interval construction. Therefore, the larger batch sizes (or smaller number of batches if sample size is fixed) should result in higher coverages, but not necessarily at lower half-width measure or lower average MSE. Higher half-width measures may result with the large batch sizes depending on the nature of the correlation between the batches and the number of batches used to estimate the variance.

The nature of the correlation level affects the variance estimate of constructed confidence intervals. As positive correlation increases, the more the variance is under-estimated. Alternately, as negative correlation increases, the more the variance estimate is over-estimated. For confidence interval construction, positive correlation has a more detrimental effect on confidence interval performance than negative correlation. Under-estimation of the variance caused by positive correlation results in confidence intervals that are not wide enough to cover the true mean at the desired confidence level. Therefore, too much confidence will be placed in the confidence interval than there should be. There is little problem, however, with the effect of negative correlation. Placing less confidence in a wider confidence interval that will more likely cover the true mean more often than expected is wasteful, yet not harmful.

From the empirical results previously discussed, it is apparent that as one factor level was changed while holding the other factor levels constant, the performance measures of confidence intervals constructed by the PC procedure were affected in dramatic ways. Increasing the sample size while holding the batch size constant, in general, increased the coverage and enhanced the half-width, average MSE, and bias performance measures of confidence intervals that were constructed by the BM method. However, somewhat different results occurred with the PC method and the results were model-dependent.

Increasing the samples size for the M/M/1 model caused better coverage and lower half-width measures, as anticipated, for confidence intervals that were constructed by the PC method. However, for both the AR(1) model and the ARMA(1,1) model, comparable coverages occurred regardless of sample size for confidence intervals that were constructed by the PC method. There was no noticeable trend in increased coverage as the sample size was successively doubled for PC confidence intervals as was evident with BM confidence intervals. Also, the half-width measure did not change drastically with increasing sample size, which suggests that constructed confidence intervals may be indeed wider at smaller sample sizes, yet still are rather stable. Finally, the average MSE of the PC confidence intervals for the time series models tended to decrease as the sample size increased, which is similar to the results of the BM confidence intervals. However, the PC confidence intervals tended to become biased high as the sample size increased, which is quite different from the results of the BM confidence intervals.

Increasing the batch size while holding the sample size constant, generally, increased the coverage, increased the half-width measure, decreased the average MSE, and decreased the bias of failed confidence intervals for BM confidence intervals. However, increasing the batch size while holding the sample size constant for PC confidence intervals decreased the coverage and had little or no effect on the half-width and the bias measures. This result is typical of all models and at all sample sizes.

4.6 Coverage Function

The coverage function is a graphical measure developed by Schruben [1980] to test the robustness of confidence interval procedures. Typically, the coverage of a particular confidence interval procedure is estimated by the frequency of constructed confidence intervals that cover the true mean at a particular confidence level. Since many researchers use different confidence levels to test their confidence interval procedure, there is no means by which comparisons of coverage performance can be made. Therefore, the coverage function is suggested as a means to compare the coverage of different confidence interval procedures.

The coverage function is basically a graph of the desired coverage to the actual coverage at various confidence levels. A well-behaved confidence interval procedure would yield a uniformly distributed coverage function that could be graphically seen and statistically tested. Several coverage functions of analytical models are shown in the succeeding pages of this section. The data for each coverage function was obtained by constructing confidence intervals from 100 replicates of an analytical model, which was run at the largest possible sample size and at each batch size corresponding to the specified number of batches. Confidence intervals for each replicate were constructed at various confidence levels ranging from .01 to .99. Both the BM procedure and the PC procedure have their coverage functions for each model set at different model parameters graphically represented together on the same figure. The basis for comparison between the two procedures is to note how close the coverage functions are to an ideal 45 degree line, which corresponds to an ideal uniformly distributed coverage function.

4.6.1 Coverage Function, M/M/1

Figures 4.1 through 4.9 show the coverage functions for 100 replicates of confidence intervals that were constructed by the BM and the PC methods on an M/M/1 queueing model with ρ=.5, .8, and .9, and at batch sizes corresponding to

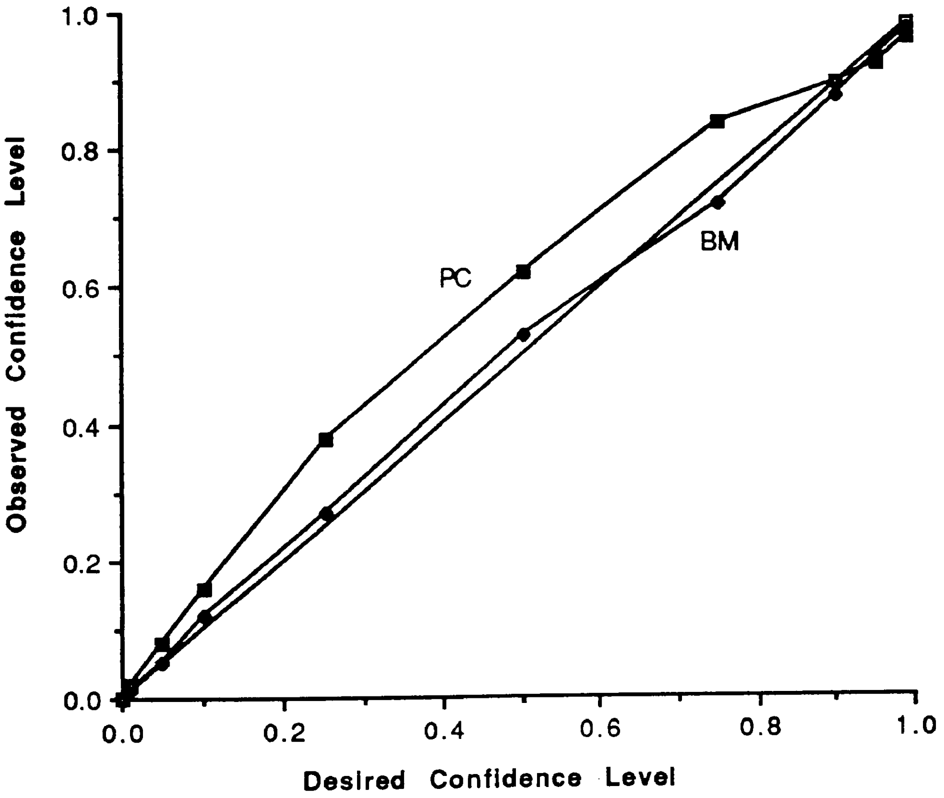

Figure 4.1 Coverage Function
M/M/1, p = .5, nb =10

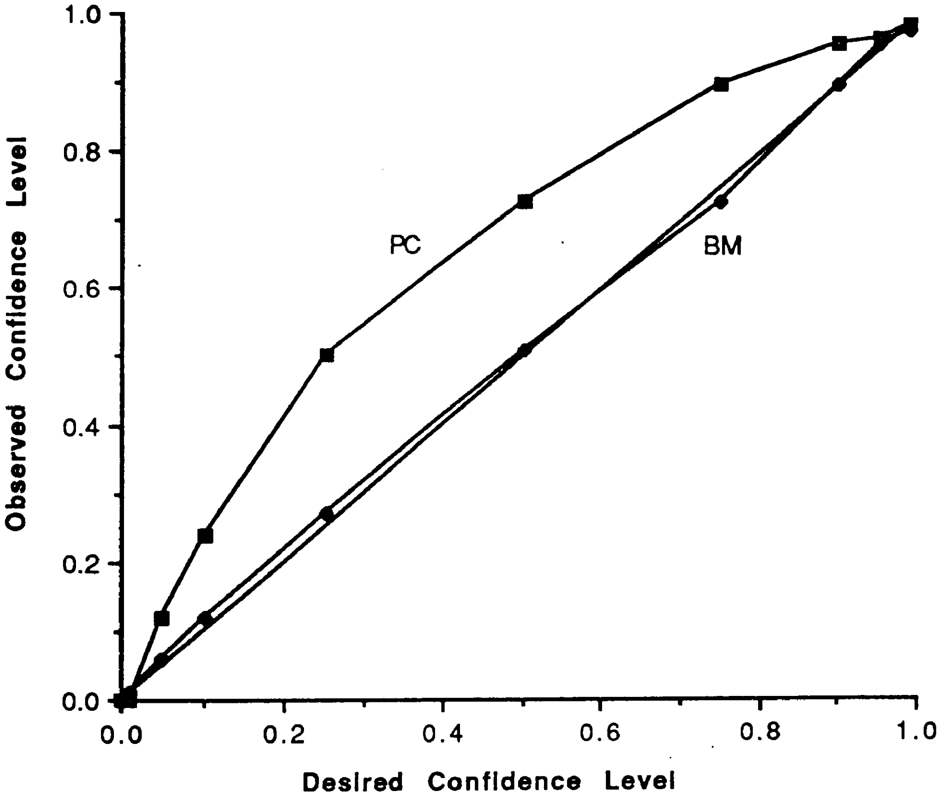

Figure 4.2 Coverage Function
M/M/1, *p* = .5, nb = 20

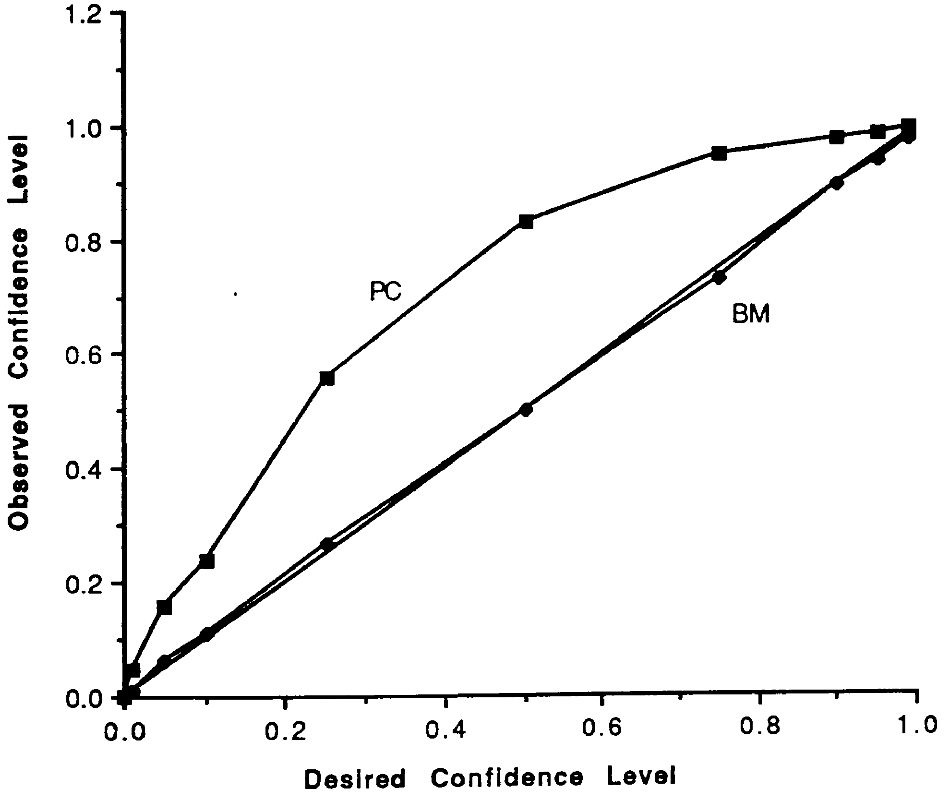

Figure 4.3 Coverage Function
M/M/1, *p* = .5, nb = 40

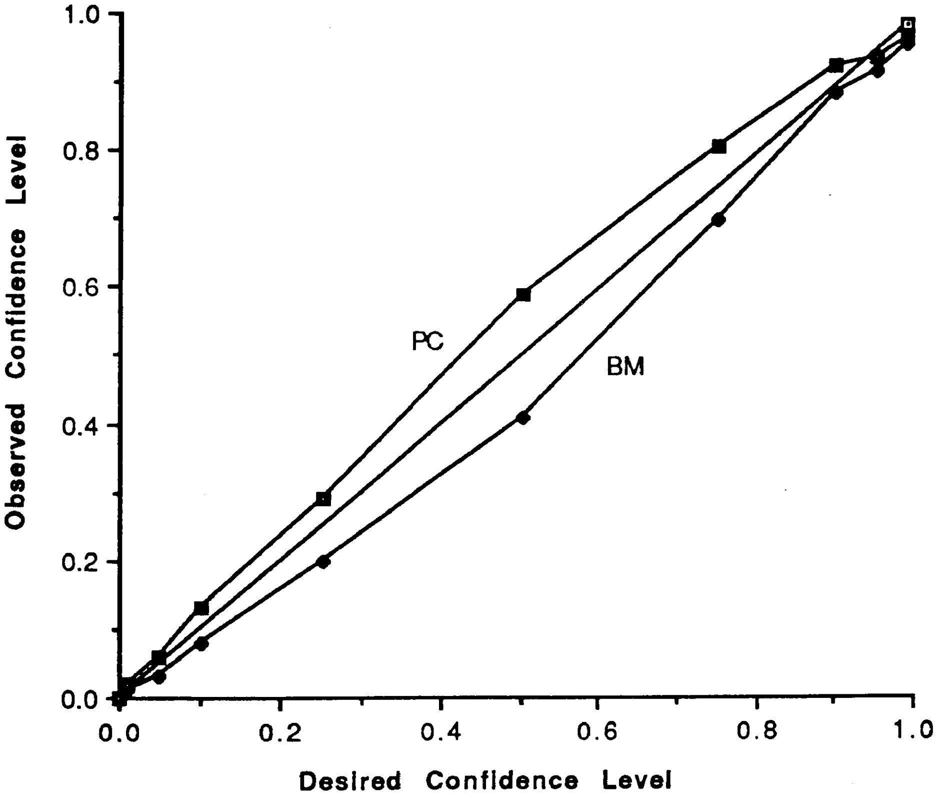

Figure 4.4 Coverage Function
M/M/1, *p* = .8, nb = 10

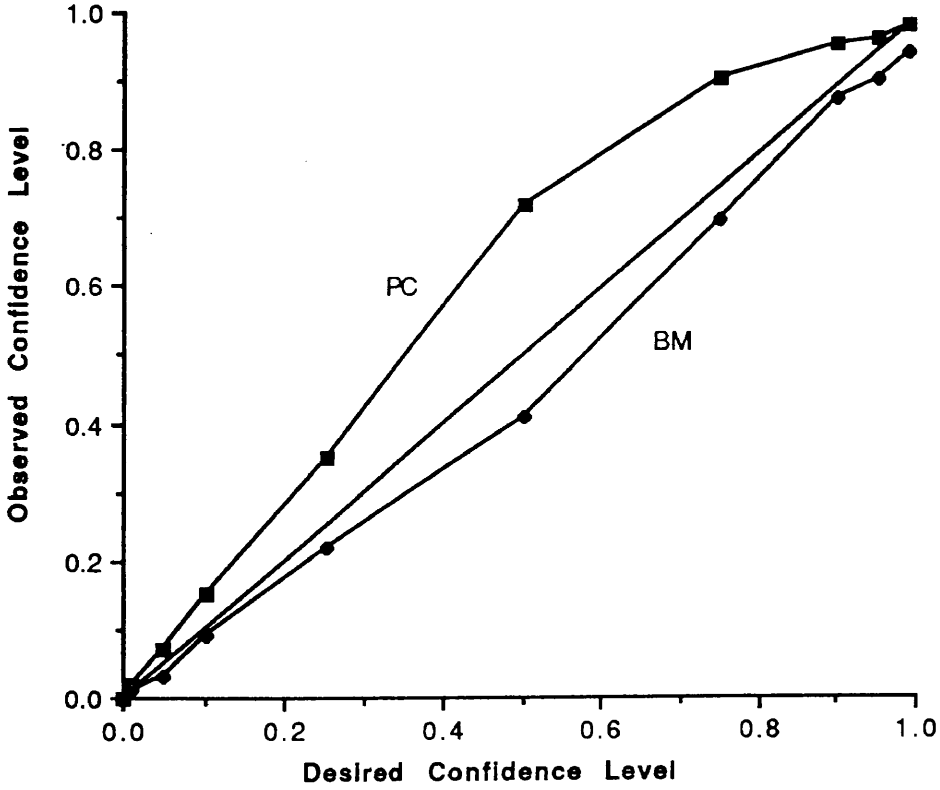

Figure 4.5 Coverage Function
M/M/1, *p* = .8, nb = 20

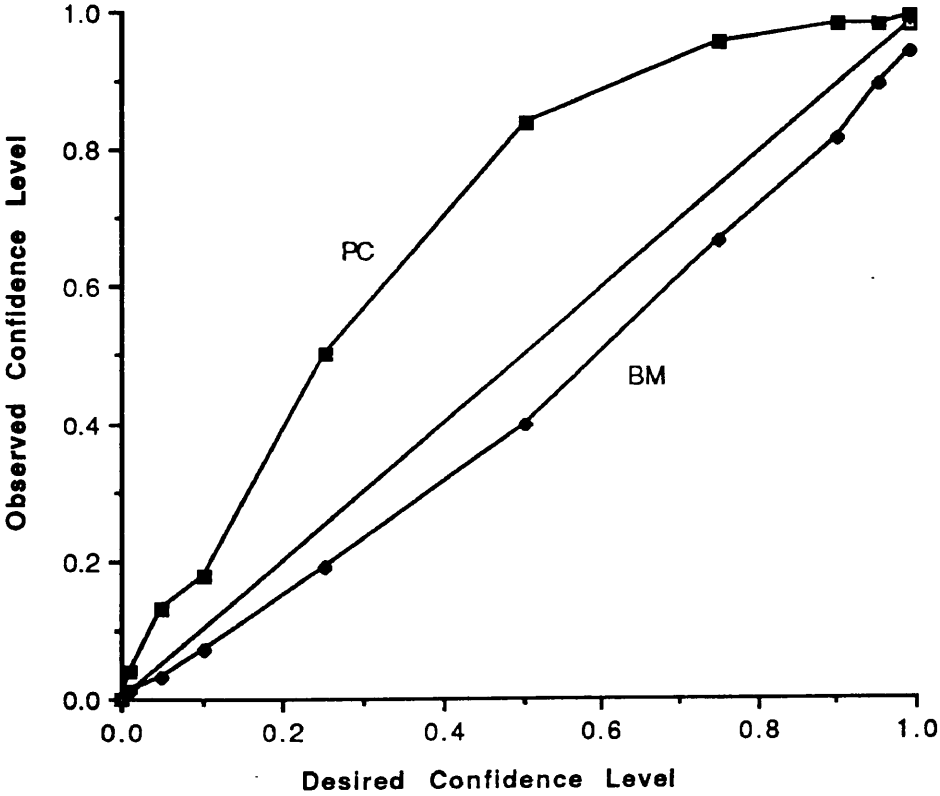

Figure 4.6 Coverage Function
M/M/1, p =.8, nb = 40

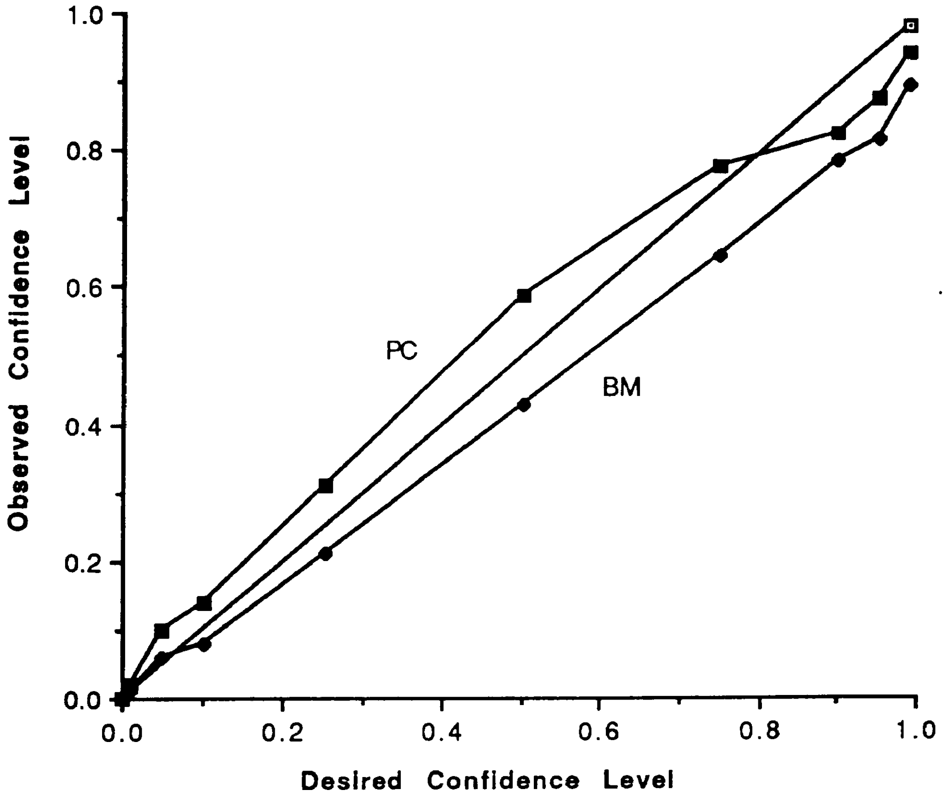

Figure 4.7 Coverage Function
M/M/1, *p* = .9, nb = 10

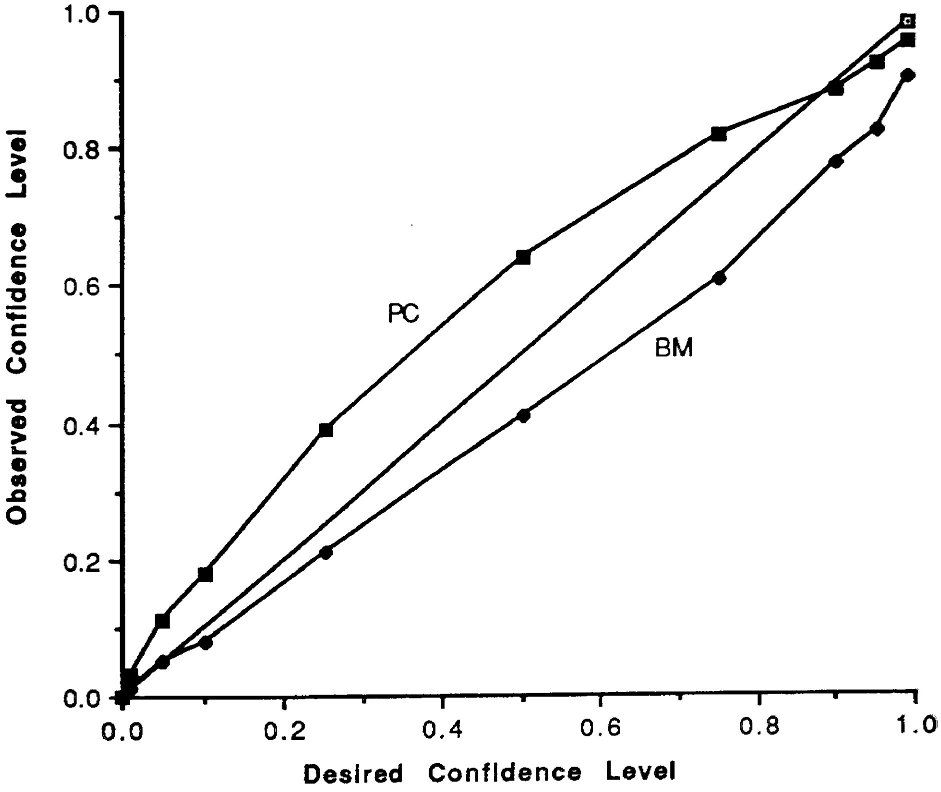

Figure 4.8 Coverage Function
M/M/1, *p* = .9, nb = 20

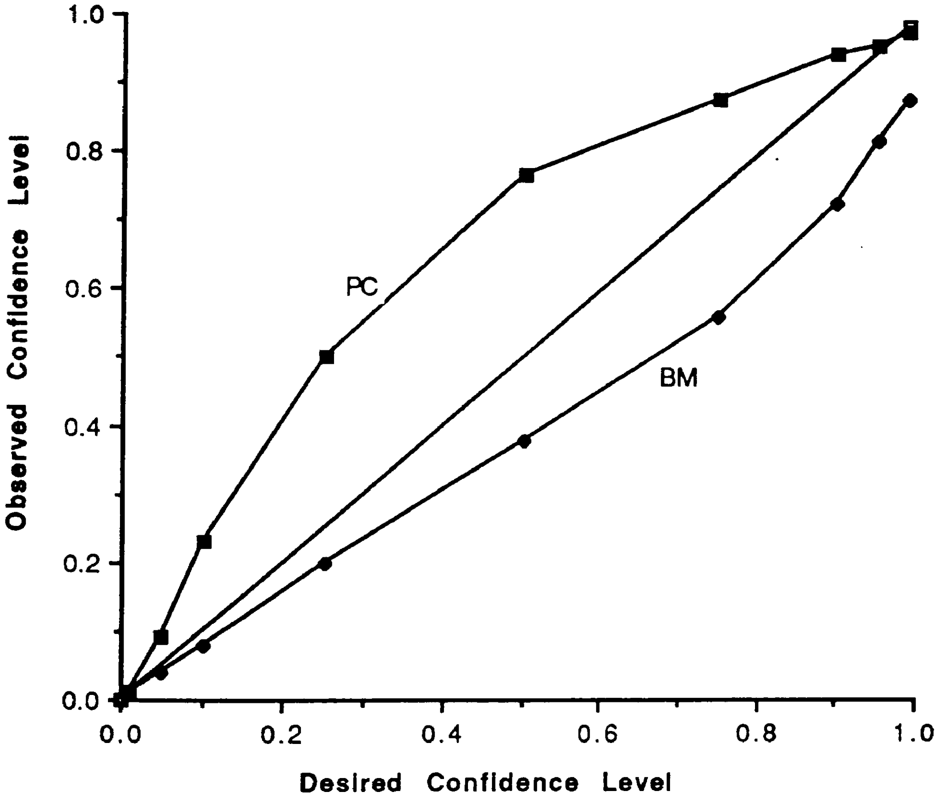

Figure 4.9 Coverage Function
M/M/1, *p* = .9, nb = 40

10, 20, and 40 batches, respectively. For $\rho = .5$, the BM procedure is clearly more robust than the PC procedure. For all batch sizes, the coverage function of the BM procedure is closer to the ideal 45 degree line than the coverage function of the PC procedure. As the batch size decreases, the coverage function of the PC method departs further from the ideal, especially in the low to medium range of confidence levels. The coverage function of the PC method suggests that the method provides somewhat inflated confidence intervals that will exceed the desired coverage for confidence levels of interest, such as the 75 percent and above confidence levels. While this result is not necessarily disadvantageous, it is somewhat wasteful. There are not many analysts who would be very disappointed if their constructed confidence intervals tended to exceed the stated coverage. However, the analyst may likely prefer to select a lower confidence level if it will result in a tighter confidence interval that would yield the desired coverage.

For $\rho = .8$, the PC method inflates the coverage for all confidence levels while the BM method under-estimates the coverage for all confidence levels. For the higher confidence levels of interest, the coverage function of the PC method under-estimates the desired coverage, yet it does not under-estimate by as much as the BM method. Therefore, it is more suitable than the BM method for attaining confidence intervals with good coverage.

For $\rho = .9$, the coverage function of the BM method severely under-estimates the true coverage at all confidence levels. As the batch size decreases, the under-estimation becomes more severe. For $nb=40$ batches, the coverage is so severely under-estimated that the BM method becomes unsuitable as a confidence interval procedure. The coverage function of the PC method exceeds the true coverage for low to medium range confidence levels. However, at higher confidence levels of interest, the coverage function of the PC method slightly under-estimates the true coverage, but is still closer to the ideal than the BM method.

4.6.2 Coverage Function AR(1)

Figures 4.10 through 4.15 show the coverage functions for 100 replicates of confidence intervals that were constructed by the BM and the PC method on an AR(1) model with $\rho=.5$ and .9, at a fixed large-sample size and at various batch sizes. For $\rho=.5$, the coverage function of the BM method over-estimates the coverage for very low confidence levels and under-estimates the coverage for medium to high confidence levels. The PC method over-estimates for low to medium confidence levels and under-estimates for high confidence levels. However, at higher confidence levels of interest, the PC method's coverage function is still closer to the ideal than the coverage function of the BM method.

For $\rho=.9$, the coverage function of the BM method for nb = 10 and 20 batches over-estimates the coverage at low confidence levels. The coverage function of the PC method over-estimates the coverage for low to medium confidence levels and under-estimates the coverage at high confidence levels. At high confidence levels, however, the coverage function of the PC method is closer to the ideal than

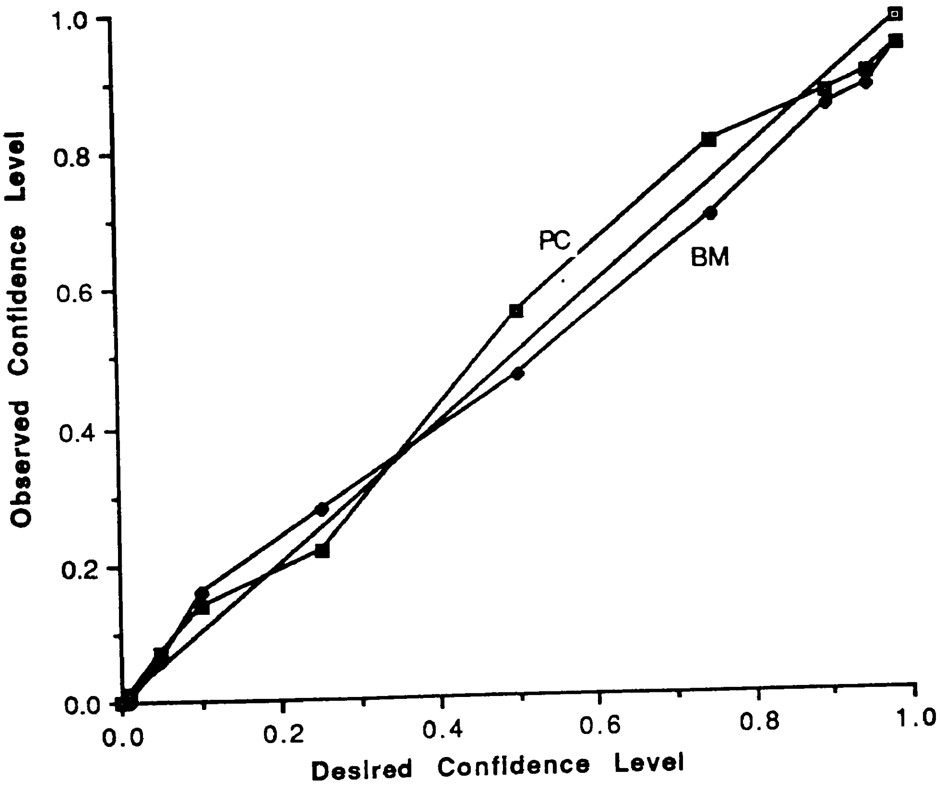

Figure 4.10 Coverage Function
AR(1), p = .5, nb = 10

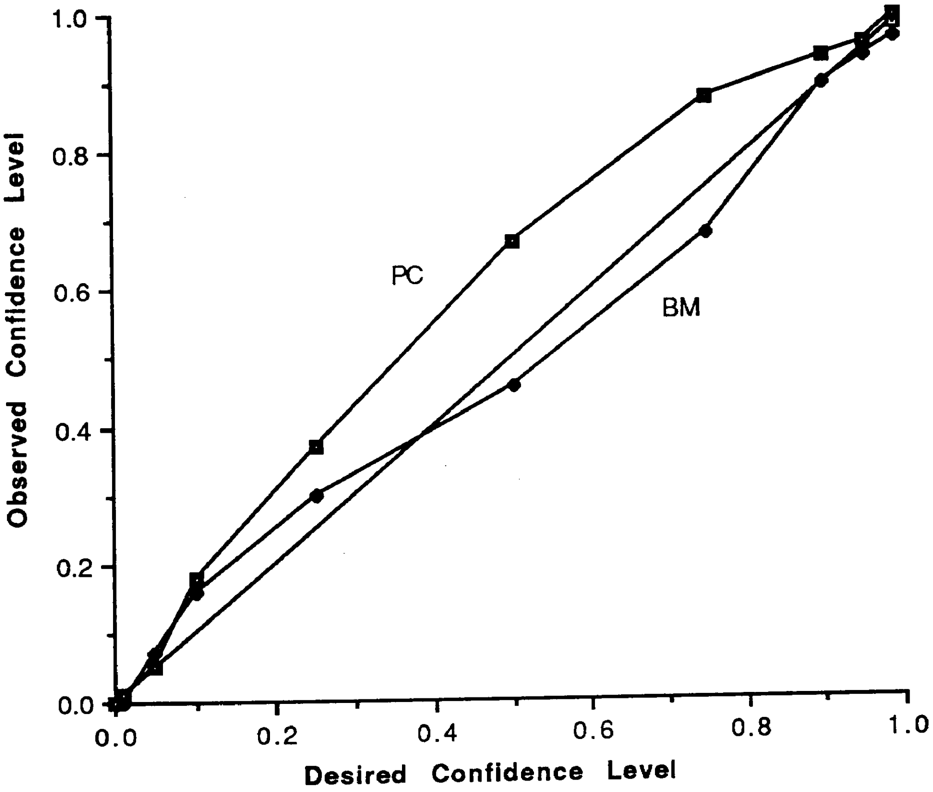

Figure 4.11 Coverage Function
AR(1), p = .5, nb = 20

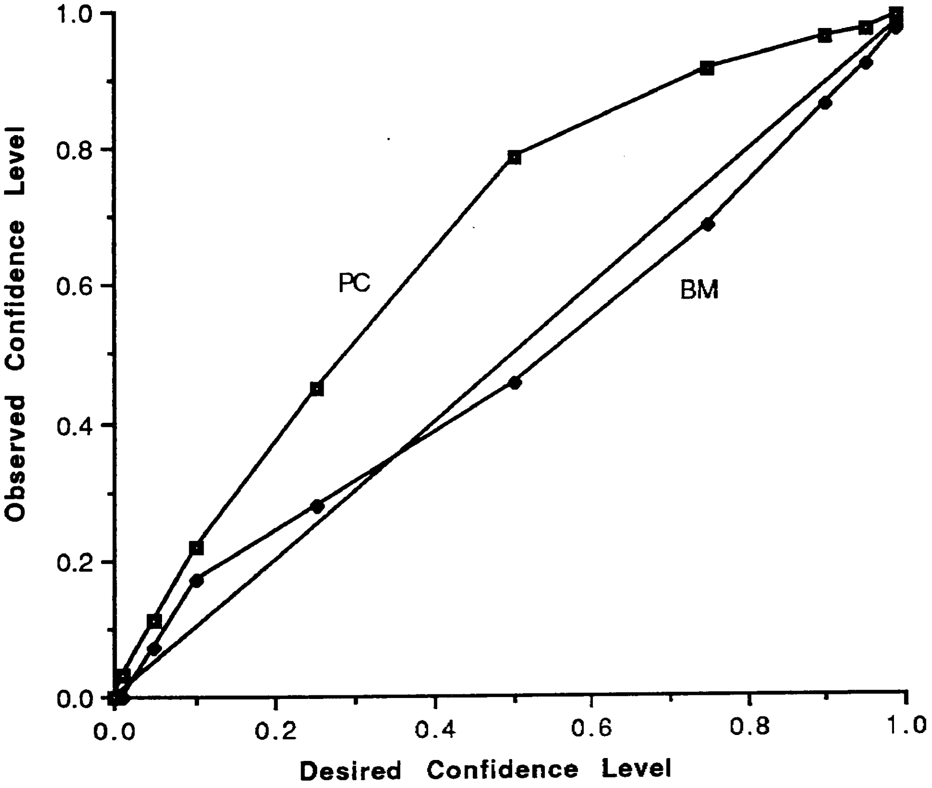

Figure 4.12 Coverage Function
AR(1), p = .5, nb = 40

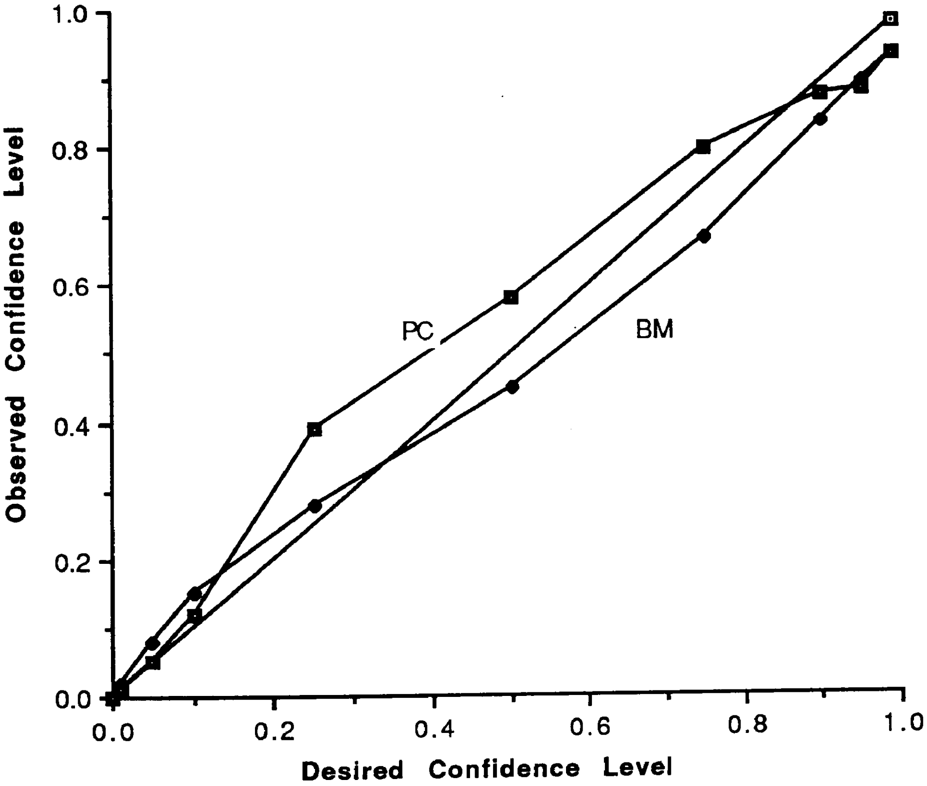

Figure 4.13 Coverage Function
AR(1), *p* = .9, nb = 10

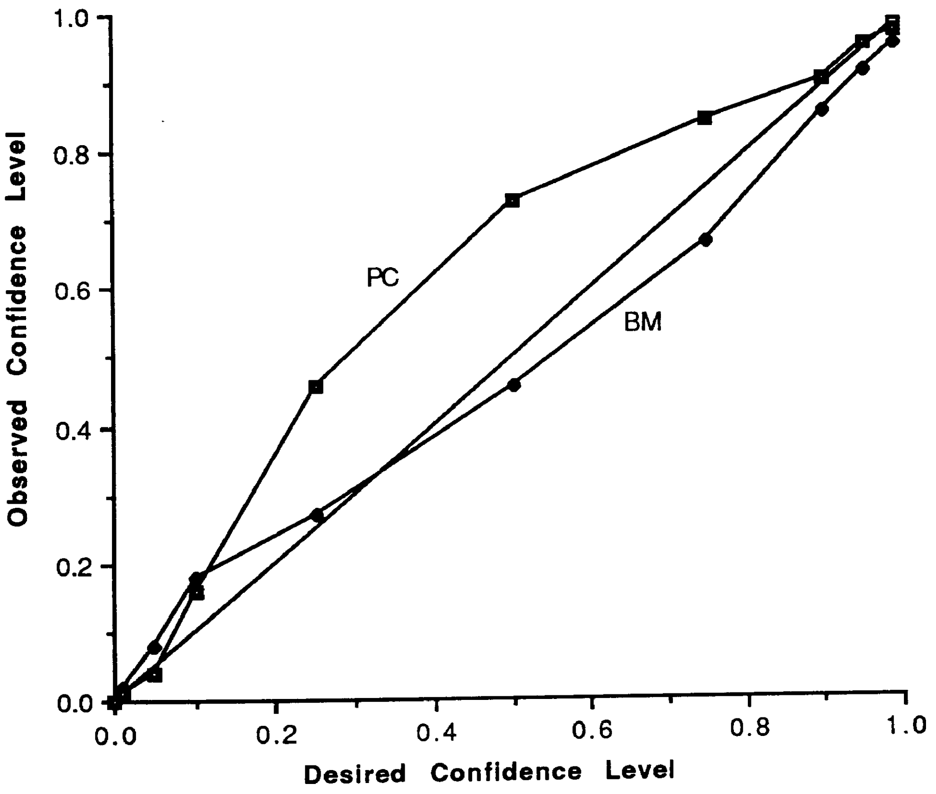

Figure 4.14 Coverage Function
AR(1), *p* = .9, nb = 20

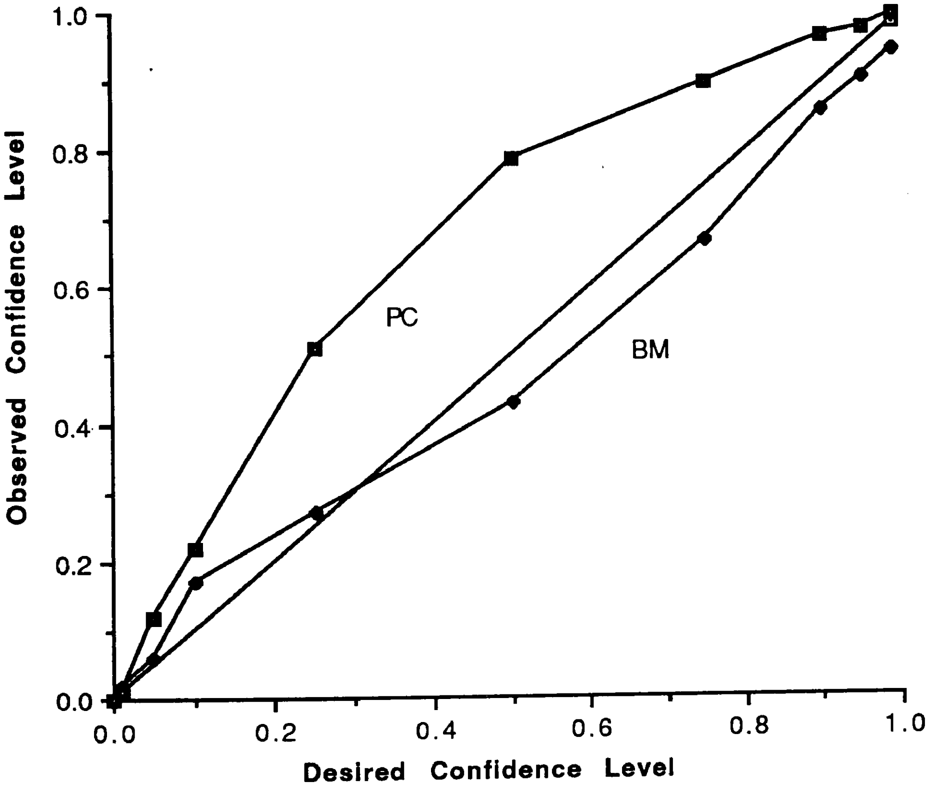

Figure 4.15 Coverage Function
AR(1), p = .9, nb = 40

the coverage function of the BM method. For batch sizes corresponding to 40 batches, the coverage function of the BM method behaves the same as it does for 10 and 20 batches; however, the coverage function of the PC method, now, exceeds the desired coverage at all confidence levels.

4.6.3 Coverage Function ARMA(1,1)

Figures 4.16 through 4.18 show the coverage functions for 100 replicates of confidence intervals that were constructed by the BM and the PC methods on an ARMA(1,1) model for a fixed large-sample size and at various batch sizes. The coverage function of the BM method under-estimates the true coverage at all batch sizes and at all confidence levels For batch sizes corresponding to 10 and 20 batches, the PC method, however, over-estimates the true coverage at low to medium confidence levels and under-estimates the coverage at high confidence levels. For 40 batches, the coverage function of the PC method exceeds the desired confidence level at all confidence levels.

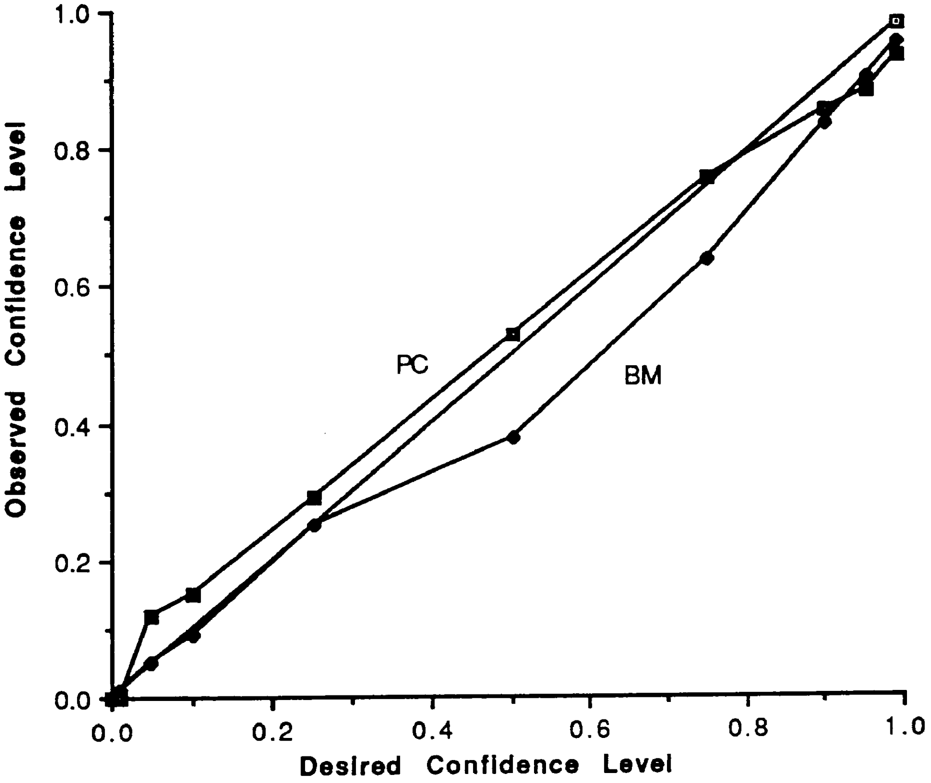

Figure 4.16 Coverage Function
ARMA(1,1), nb =10

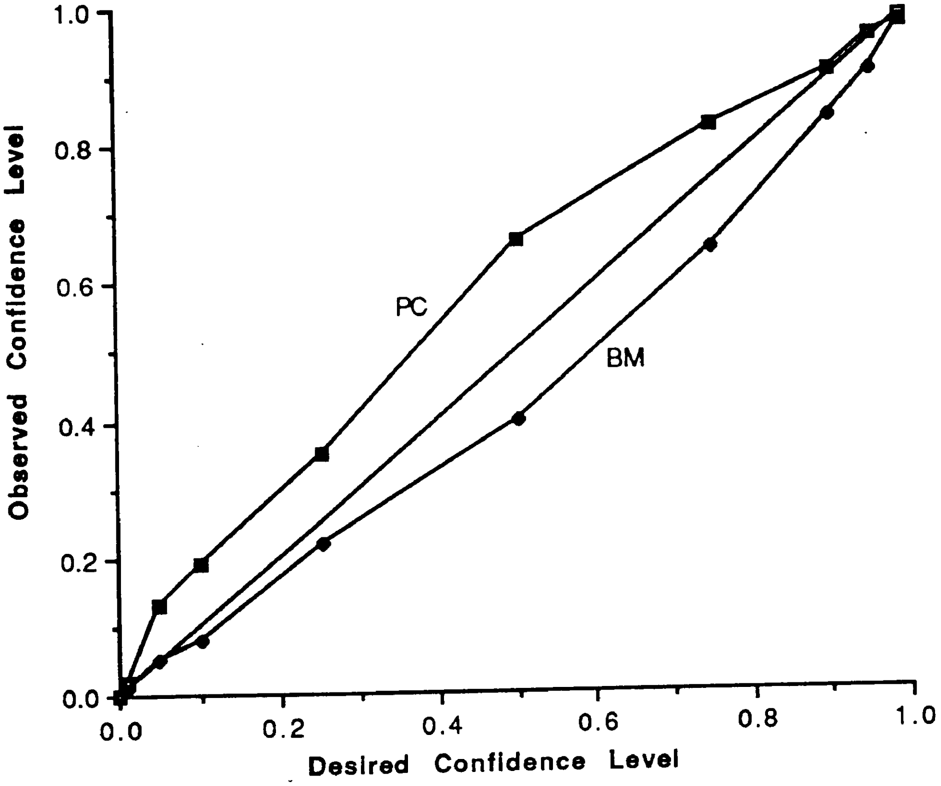

Figure 4.17 Coverage Function
ARMA(1,1), nb = 20

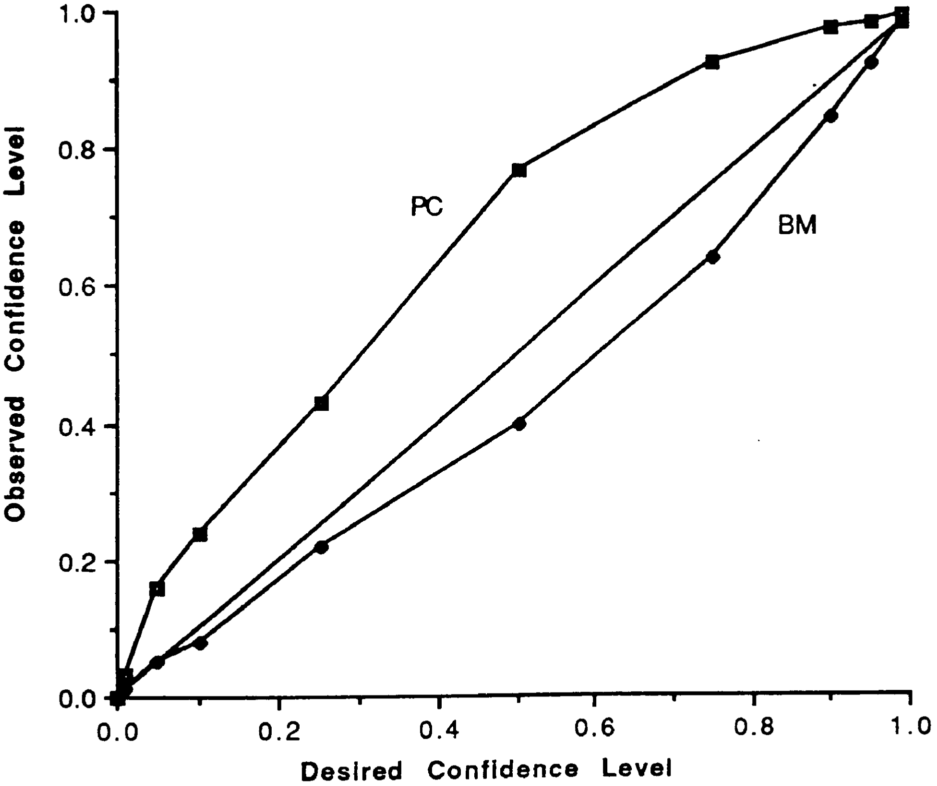

Figure 4.18 Coverage Function
ARMA(1,1), nb = 40

CHAPTER V

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

This research has developed a new procedure to handle the problem of data correlation in simulation output analysis. The principal component method is derived from matrix diagonalization theorems, which allow for an orthogonal transformation of the original output data with an estimated covariance structure into data with an uncorrelated structure. The performance of the PC method was empirically tested according to an acceptable conceptual framework for analyzing confidence interval methodologies. This framework consists of testing selected performance measures of confidence intervals that are constructed by the proposed confidence interval procedure and comparing them to the performance measures of one other well-known confidence interval procedure, the batch mean procedure. Three different theoretical output processes with known analytically determined means were used for testing. An M/M/1 queueing model at various utilization rates, an AR(1) time series model at various correlation levels, and an ARMA(1,1) time series model with oscillatory correlation structure were used. For all models, 100 replicates of sample sizes ranging from as small as 320 observations to as large as 10240 observations were done. In addition, each fixed sample size was run at batch sizes corresponding to number of batches equal to 10, 20 and 40 batches. The 90% confidence intervals of the proposed procedure were compared to the 90%

confidence intervals of the batch mean procedure by calculating selected performance measures. The performance measures used for comparing the two methods include the nominal coverage probability, the standardized standard deviation of the half-width, the average mean square error, the number of confidence intervals that failed to cover the true mean by exceeding it, and the number of confidence intervals that failed to cover the true mean by falling short of it.

5.1 Summary

A summary of the results of this research follows. Performance of the PC procedure as compared to the performance of the BM procedure for each theoretical model is discussed, as measured by the coverage, the coverage function, the half-width performance, the average MSE and the bias distribution of confidence intervals that failed to cover the true mean.

M/M/1 Queueing Model

(1) For both methods, coverage eventually approached the desired confidence level if the sample size was large enough. The coverage of the PC method exceeded the coverage of the BM method for every batch size/sample size combination. Unlike the BM method, the PC method provided confidence intervals with excellent coverage at the smallest sample sizes.

(2) The coverage functions of both methods showed markedly different behavior, which became more pronounced as the correlation level increased either

as a result of increasing the model's utilization factor or of reducing the batch size for a particular fixed sample size. In general, the PC method tended to provide nominal coverages that exceeded the desired confidence level while the BM method tended to provide nominal coverages less than the desired confidence level. As the correlation level increased, the departure from the desired confidence level grew stronger for both methods, with the effect being more pronounced with the PC method than with the BM method. As previously mentioned, coverages that exceed the stated confidence level may be wasteful, but are not harmful as when coverages fall short of the stated confidence level. Therefore, the coverage function behavior of the PC method suggests it to be a more robust and reliable method of providing confidence intervals with excellent coverage than the BM method.

(3) For both methods, the half-widths of confidence intervals that were constructed became smaller and more stable as the sample size increased. Unlike the BM method, increasing the batch size for a fixed sample size did not significantly change the half-width (stability) measure. The confidence intervals that were constructed by the PC method at a particular fixed sample size remained at about the same level of half-width stability, regardless of how many batches were used to construct them. As expected, as the utilization rates of the queueing model increased, which indicated increased correlation level, the half-width stability decreased. Wider and more variable confidence intervals were required in order to get the desired coverage. A surprising result, however, was that the half-width

stability measure of the PC method was significantly larger than that of the BM method. This was due to how the two methods determine the estimate of the standard error, which is the primary factor that affects half-width. This idea is discussed later in more detail.

(4) Similar to the BM method, the average MSE of confidence intervals that were constructed by the PC method decreased as the sample size increased. Increasing sample size causes a better point estimate of the mean (thus, reduces the bias) and reduces the variance. Also, the average MSE of PC confidence intervals decreased as the batch size increased. This is primarily due to the decreasing variance estimate that is obtained due to increased correlation as the batch size decreases, since the average bias is relatively unaffected by batch size, regardless of the utilization rate of the queueing model.

In all cases, the average MSE of the PC method was less than that of the BM method, which suggests that confidence intervals constructed by the PC method consistently provide better point and interval estimates of the true mean than the BM method. An astonishing result, however, is the relationship that existed between the variance estimates and the estimates of the standard error that both methods determined. The standard error that is determined by the BM method results from dividing the variance estimate by the number of batches that are used to construct the confidence interval. The standard error that is determined by the PC method, however, is not found in similar fashion. Therefore, the standard error of the PC method was somewhat larger than that of the BM method, which alone accounted for the significant difference in their half-width measures.

(5) The bias distribution of confidence intervals that failed to cover the true mean show that both methods tended to be biased low. This means that of the number of confidence intervals that failed to cover the true mean, more of them were likely to fall short of covering the true mean than to exceed covering it. However, the bias distribution of the PC method was closer to an ideally symmetrical distribution than that of the BM method, which is another indication of the better confidence interval adequacy of the PC method over the BM method.

AR(1) Time Series Model

(1) Unlike the BM method, the coverages of confidence intervals that are constructed by the PC method did not increase as the sample size increased, but remained relatively stable for all sample sizes. The PC method provided excellent coverage at all sample sizes, even for the smallest sample size of $n=320$ observations. Regardless of correlation level, the PC method provided excellent coverage at all sample size/batch size combinations. Unlike the BM method, decreasing the batch size by increasing the number of batches that are taken from a fixed sample size caused the coverage to increase. This suggests that the PC method adequately handles the problem of correlated batches in the original simulation output data.

(2) The coverage functions of both methods exhibited a variety of behaviors.

In all cases of the AR(1) model, the coverage function of the BM method exceeded the desired coverage for low confidence levels (i.e. .25 and less) and fell short of the desired coverage for medium to high confidence levels (i.e. .50 and greater). The coverage function of the PC method, however, exceeded the desired coverage for low to medium confidence levels and slightly fell short of the desired coverage at high confidence levels (i.e. .90 and greater). The PC method is preferable to the BM method for high confidence levels, which are usually the confidence levels of interest. The coverage function of the PC method either exceeds the stated confidence level or falls short of the stated confidence level by less of an amount than the BM method.

(3) Unlike the M/M/1 case, the half-width stability of both methods did not increase as the sample size increased. However, decreasing the batch size did cause an increase in the half-width stability with the effect being more pronounced in the BM method than in the PC method. Half-width stability remained relatively level although higher for the PC method when compared to that of the BM method. The reason for the higher half-width measure of the PC method is again the difference in the way the two methods determine the estimate of the standard error, which was explained previously for the M/M/1 model.

(4) For both methods, the average MSE decreased with increasing sample size and with increasing batch size. In all cases, the average MSE of the PC method was smaller than that of the BM method, which again is the result of the smaller

variance estimate that the PC method provides.

(5) The bias distribution of failed confidence intervals shows that both methods were generally biased high with the PC method showing a more significantly higher bias than the BM method. At large sample sizes, the BM method tended to be symmetrically biased while the PC method still showed a significantly high bias.

ARMA (1,1) Time Series Model

(1) Similar to the results of the previously discussed model, the coverage of the PC method significantly exceeded the coverage of the BM method in all cases and attained the desired confidence level at very small sample sizes. The coverages of both methods tended to remain fairly consistent as the sample size increased. As the batch size increased, the coverage of the BM method remained level, but that of the PC method noticeably decreased.

(2) The coverage function of the two methods shows that the BM method gave coverages that always fell short of the desired confidence level while the PC method gave coverages that usually exceeded the desired confidence level. As the batch size decreased, the coverage of the PC method departed more from the desired confidence level while the coverage of the BM method remained relatively the same. For the higher confidence levels of interest, the PC method provided coverages that were closer to the stated confidence level than the BM method.

(3) Similar to the results of the AR(1) model, the half-width stability of both methods did not increase significantly as the sample size increased. However,

decreasing the batch size did increase the half-width stability for both methods. As stated previously, the half-width measure of the PC method was significantly greater than that of the BM method because of the way that the two methods determine the standard error.

(4) Similar to the results of the previous models, the average MSE of the PC method was smaller than that of the BM method. As the sample size increased or as the batch size increased, the average MSE's of both methods increased.

(5) The bias distribution of failed confidence intervals show that both methods tended to be biased high. Most of the confidence intervals that failed to cover the true mean tended to exceed the true mean rather than fall short of the true mean.

5.2 Conclusions

Based on the results presented previously, certain conclusions may be drawn. Although the batch mean will still likely be one of the more popular confidence interval procedures, the method that was developed in this research has some distinct advantages. Some of the major conclusions are as follows:

(1) The PC method does not require an excessively large number of simulated observations in order to determine an appropriate batch size to construct the confidence intervals as is required by the batch means method. It uses any arbitrary number of batches from a fixed sample size instead and requires no statistical test for correlation to ensure that the batch size is sufficiently large enough

to ensure uncorrelated batches.

(2) The PC method provides higher coverage than the BM method. More noteworthy, the PC method will usually provide a confidence interval with good coverage for small sample sizes and for the most correlated data.

(3) The PC method considers and directly adjusts for, rather than ignores, the effect of correlation on estimating the variance. This differs from the BM method, which tries to obtain a batch size that eliminates correlation. This also differs from other methods, such as time series and spectral methods, which try to estimate the standard error of the mean by modeling the original data as some parametric model.

(4) The PC method does not require significantly more computational effort to be implemented since it deals directly with the given original data. It does not try to fit an analytical model to it. It does not need to use extra data to determine the appropriate batch size that ensures uncorrelated batches.

(5) The PC method is grounded in theoretical concepts of matrix algebra and statistical estimation; therefore, unlike the procedures that are done by the batch means method to determine the appropriate batch size that ensures uncorrelated batches, implementation of the new method is more science than art.

(6) The best performance of the new method seems to occur when the batch size is not too large, say, a batch size corresponding to forty batches from a fixed sample size. The best coverages result from 40 batches of the largest possible fixed

sample size and the half-width stability is adequate although higher than with other batch sizes.

(7) For a fixed sample size, the new method adequately provides an uncorrelated version of the data since very few replicates that were used to construct confidence intervals failed the test for correlated batches. The largest number of replicates with correlated batches that was ever used inappropriately to construct confidence intervals by the PC method was six out of a possible 100 replicates. The BM method, however, typically had more than one-half of the total 100 possible replicates failing the test for correlated batches and being inappropriately used to construct confidence intervals. Usually, the BM method had this problem if the batch size and the sample size were too small.

(8) The PC method provides confidence intervals that are fairly stable even at small sample sizes. In fact, the half-width stability remains rather constant regardless of sample size. This suggests that even for small sample sizes, confidence intervals with good coverage are obtained without having to be significantly wider than large-sample-size confidence intervals.

(9) . The PC method provides a framework for further investigation of simulation output methods based on multi-normal assumptions. In most experimentation with simulation output analysis, the emphasis has been on univariate normal assumptions and developing models or tools to estimate the standard error. Future experimentation may now be developed based on the framework of this

research to develop other approaches to estimating the standard error.

(10) The PC method is, however, limited by the size of matrices that can be successfully diagonalized by using current programming routines. The maximum size of the covariance matrix that was possible for this research was *50 x 50*. Most other subroutines, in general, suffer like limitations in the size of the covariance matrix.

5.3 Recommendations

Several areas for further research are suggested in this section. Most work in the area of simulation fixed-width confidence interval methodologies is usually extended to developing sequential approaches. A sequential procedure is one that iteratively increases the sample size until the resulting confidence interval half-width is of a desired precision. The principal component method could very easily be extended to a sequential procedure. However, some arbitrary number of batches must be selected *a priori* since the size of the covariance matrix which is to be diagonalized is extremely limited.

The estimate of the covariance structure is the key element of the new method. Most estimates of the covariance require large sample sizes since the estimates themselves are highly variable and, in fact, correlated with each other if the sample size is too small. Therefore, another recommendation for future research is an investigation of the effect that correlation and sample size has on estimating the covariance matrix.

Another research topic is to develop some means other than linear unbiased estimation to determine estimates of the mean of transformed data derived from a diagonalized covariance matrix. Linear unbiased estimation gives an excellent point estimate of the mean, but the estimate of the variance is too small to be useful or reliable for constructing confidence intervals. Until the principal component method was modified in its estimate of the variance, it yielded confidence intervals with unacceptable coverage.

Another research topic is to develop a procedure based on the principal component method which will provide confidence intervals for steady-state measures other than the mean. Iglehart [1976] and Seila [1982] provide methods based on the regenerative approach that give quantile estimation. This is useful because in some cases, such as in highly skewed data, the estimate of the median may be a better measure of central tendency than the mean.

REFERENCES

Adam, N.R. 1983. Achieving a Confidence Interval for Parameters Estimated by Simulation. *Mgmt. Sci.* **29**, 856-866.

Bischak, D.P. 1988. Weighted Batch Means for Improved Confidence Intervals for Steady State Processes. Ph.D. Dissertation, Dept. of Industrial and Operations Engineering, The University of Michigan, Ann Arbor.

Brillinger, D.R. 1973. Estimation of the Mean of a Stationary Time Series by Sampling. *J. Appl. Prob.* **10**, 419-431.

Chun, Y. 1989. Time-Series Models of Batch-Means Processes in Simulation Analysis. Ph.D. Dissertation, Dept. of Business Administration, University of Minnesota, Minneapolis.

Crane, M.A. and D.L. Iglehart. 1974a. Simulating Stable Stochastic Systems: I, General Multi-Server Queues. *J. ACM* **21**, 103-113.

Crane, M.A. and D.L. Iglehart. 1974b. Simulating Stable Stochastic Systems: II, Markov Chains. *J. ACM* **21**, 114-123.

Crane, M.A. and D.L. Iglehart. 1975a. Simulating Stable Stochastic Systems: III, Regenerative Processes and Discrete Simulations. *Opns. Res.* **23**, 33-45.

Crane, M.A. and D.L. Iglehart. 1975b. Simulating Stable Stochastic Systems: IV, Approximation Techniques. *Mgmt. Sci.* **21**, 1215-1224.

Crane, M.A. and A.J. Lemoine. 1977. *An Introduction to the Regenerative Method for Simulation Analysis*. Lecture Notes in Control and Information Sciences. Springer-Verlag, New York.

Damerdji, H. 1988. Topics in Discrete-Event Stochastic Systems. Ph.D. Dissertation, Dept. of Operations Research, The University of Wisconsin-Madison.

Donnelly, J.H. and R.E. Shannon. 1981. Minimum Mean-Squared Error Estimators for Simulation Experiments. *Comm. ACM* **24**, 253-259.

Duket, S.D. and A.A.B. Pritsker. 1978. Examination of Simulation Output Using Spectral Methods. *Math. Comp. Simulation XX* **1**, 53-60.

Fishman, G.S. and P.J. Kiviat. 1967. The Analysis of Simulation-Generated Time Series. *Mgmt. Sci.* **13**, 525-557.

Fishman, G.S. 1971. Estimating Sample Size in Computing Simulation Experiments. *Mgmt. Sci.* **18**, 21-38.

Fishman, G.S. 1972. Bias Considerations in Simulation Experiments. *Opns. Res.* **20**, 785-790.

Fishman, G.S. 1973. Statistical Analysis for Queueing Simulations. *Opns. Res.* **20**, 363-369.

Fishman, G.S. 1974. Estimation in Multi-Server Queueing Simulations. *Opns. Res.* **22**, 72-78.

Fishman, G.S. 1978a. Grouping Observations in Digital Simulations. *Mgmt. Sci.* **24**, 510-521.

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

Gafarian et al. 1978. Evaluations of Rules for Detecting Steady State in Simulations. *Naval Res. Logist. Quart.* **25**, 511-529.

Goldsman, D. and L. Schruben. 1984. Asymptotic Properties of Some Confidence Interval Estimators for Simulation Output. *Mgmt. Sci.* **30**, 1217-1225.

Goldsman, D. and L. Schruben. 1990. New Confidence Interval Estimators Using Standardized Time Series. *Mgmt. Sci.* **36**, 393-397.

Goldsman, D. et al. 1990. Properties of Standardized Time Series Weighted Area Variance Estimators. *Mgmt. Sci.* **36**, 602-612.

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

Heidelberger, P. and P.D. Welch. 1981. A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations. *Comm. ACM* **24**, 233-245.

Heidelberger, P. and P.D. Welch 1981a. Regression-Adjusted Estimates for Regenerative Simulations, with Graphics. *Comm. ACM* **24**, 260-273.

Heidelberger, P. and P.D. Welch. 1981b. Adaptive Spectral Methods for Simulation Output Analysis. *IBM J. Res. Dev.* 25, 860-876.

Heidelberger, P. and P.D. Welch. 1983. Simulation Run Length Control in the Presence of an Initial Transient. *Opns. Res.* **31**, 1109-1144.

Ho, C. 1986. The Method of Sequential Systematic Sampling in Digital Simulation. Ph.D. Dissertation, Dept. of Operations Research, Virginia Polytechnic Institute and State University, Blacksburg.

Iglehart, D.L. 1975. Simulating Stable Stochastic Systems: V, Comparison of Ratio Estimators. *Naval Res. Logist. Quart.* **22**, 553-565.

Iglehart, D.L. 1976. Simulating Stable Stochastic Systems: VI, Quantile Estimation. *J. ACM* **23**, 347-360.

Iglehart, D.L. 1978. The Regenerative Method for Simulation Analysis. In *Current Trends in Programming Methodology: Vol III. Software Engineering,* K.M. Chandy and R.T. Yeh (eds.). Prentice-Hall, New Jersey, 52-71.

Iglehart, D.L. and G.S. Shedler. 1980. *Regenerative Simulation of Response in Networks of Queues*. Lecture Notes in Control and Information Sciences. Springer-Verlag, New York.

Iglehart, D.L. and G.S. Shedler. 1983. Statistical Efficiency of Regenerative Simulation Methods for Networks of Queues. *Adv. Appl. Prob.* **15**, 183-197.

Jennings, D.E. 1987. How Do We Judge Confidence Interval Adequacy? *Amer. Stat.* **41**, 335-337.

Kang, K. 1984. Confidence Interval Estimation via Batch Means and Time Series Modeling. Ph.D. Dissertation, Dept. of Industrial Engineering, Purdue University, W. Lafayette, Indiana.

Kang, K. and B. Schmeiser. 1987. Properties of Batch Means from Stationary ARMA Time Series. *Opns. Res. Letts.* **6**, 19-24.

Kelton, W.D. 1983. Simulation Analysis. In *Proceedings of the 1983 Winter Simulation Conference*, The Society for Computer Simulation, Arlington, Virginia, 159-168.

Kelton, W.D. and A.M. Law. 1983. A New Approach for Dealing with Start-Up Problem in Simulation. *Naval Res. Logist. Quart.* **30**, 641-658.

Kelton, W.D. 1984. Steady-State Confidence Interval Methodology: A Forum on Theory, Practice and Prospects. In *Proceedings of the 1984 Winter Simulation Conference*, The Society for Computer Simulation, Dallas, Texas, 243-249.

Kelton, W.D. and A.M. Law. 1984. An Analytical Evaluation of Alternative Strategies in Simulation. *Opns. Res.* **32**, 169-184.

Kolman, B. 1980. *Introductory Linear Algebra with Applications*. McMillan, New York.

Law, A.M. 1977. Confidence Intervals in Discrete Event Simulation: A Comparison of Replication and Batch Means. *Naval Res. Logist. Quart.* **24**, 667-678.

Law, A.M. and J.S. Carson, II. 1979. A Sequential Procedure for Determining the Length of a Steady State Simulation. *Opns. Res.* **27**, 1011-1025.

Law, A.M. and W.D. Kelton. 1982. Confidence Intervals for Steady State Simulations II: A Survey of Sequential Procedures. *Mgmt. Sci.* **28**, 550-562.

Law, A.M. 1983. Statistical Analysis of Simulation Output Data. *Opns. Res.* **31**, 983-1029.

Law, A.M. and W.D. Kelton. 1984. Confidence Intervals for Steady State Simulations I: A Survey of Fixed Sample Size Procedures. *Opns. Res.* **32**, 1221-39.

Mechanic, H. and W. McKay. 1966. Confidence Intervals for Averages of Dependent Data in Simulations II. Technical Report ASDD 17-202, IBM Corp. Yorktown Heights, New York.

Miller, R. 1966. *Simultaneous Statistical Inference*. McGraw-Hill. New York.

Moeller, T.L. and P.D. Welch. 1977. A Spectral Based Technique for Generating Confidence Intervals from Simulation Outputs. In *Proceedings 1977 Winter Simulation Conference*, The Society for Computer Simulation, Gaithersburg, Maryland, 177-184.

Newton, H.J. 1988. *TIMESLAB: A Time Series Analysis Laboratory.* Wadsworth and Brooke/Cole, Belmont, California.

Pritsker, A.A.B. 1984. *Introduction to Simulation and SLAM II.* Systems Publishing, W. Lafayette, Indiana.

Schmeiser, B.W. 1982. Batch Size Effects in the Analysis of Simulation Output. *Opns. Res.* **30**, 556-568.

Schmeiser, B.W. 1990. Overlapping Batch Statistics. In *Proceedings of the 1990 Winter Simulation Conference*, The Society for Computer Simulation, Baltimore, Maryland, 395-398.

Schmidt, J.W. and C. Ho. 1988. An Algorithm for Testing Serial Dependence of Simulation Output Data. In *Proceedings of 1988 Winter Simulation Conference*, The Society for Computer Simulation, San Diego, California, 532-539.

Schriber, T.J. and R.W. Andrews. 1979. Interactive Analysis of Simulation Output by the Method of Batch Means. In *Proceedings of the Winter Simulation Conference*, The Society for Computer Simulation, New York, 513-525.

Schriber, T.J. and R.W. Andrews. 1981. A Conceptual Framework for Research in the Analysis of Simulation Output. *Comm. ACM* **24**, 218-232.

Schriber, T.J. and R.W. Andrews. 1984. ARMA-Based Confidence Intervals for Simulation Output Analysis. *Amer. J. Math. Mgmt. Sci.* **4**, 345-374.

Schruben, L.W. 1980. Coverage Function for Interval Estimators of Simulation Response. *Mgmt. Sci.* **26**, 18-27.

Schruben, L.W. 1982. Detecting Initialization Bias in Simulation Output. *Opns. Res.* **30**, 569-590.

Schruben, L.W. 1983. Confidence Interval Estimation Using Standardized Time Series. *Opns. Res.* **31**, 1090-1108.

Schruben, L.W. et al. 1983. Optimal Tests for Initialization Bias in Simulations. *Opns. Res.* **31**, 1167-1178.

Seila, A.F. 1982. A Batching Approach to Quantile Estimation in Regenerative Simulations. *Mgmt. Sci.* **28**, 573-581.

Seila, A.F. 1984. Batch Ratios in Discrete Event Simulation. Working Paper Series 84-150, College of Business Administration, The University of Georgia, Athens.

Vassilacopoulous, G. 1989. Testing for Initialization Bias in Simulation Output. *Simulation* **52**, 151-153.

Welch, P.D. 1983. The Statistical Analysis of Simulation Results. In *Computer Performance Modeling Handbook*, S. Lavenberg (ed.). Academic Press, New York, 268-328.

Whitt, W. The Efficiency of One Long Run vs Independent Replications in Stead-State Simulations. *Mgmt. Sci.* **37**, 645-666.

APPENDIX A

DERIVATION OF PROPOSED PROCEDURE

In this section, the mathematical manipulations that were used to derive the proposed confidence interval procedure are shown. Assume that a simulation output stream of X data values is distributed multi-normal with known mean vector of equal components and an estimated covariance structure.

If X is distributed $N(\bar{\mu}, \Sigma)$, then $Y=V^T X$ is distributed $N(V^T\ \bar{\mu},\ V^T \Sigma V)$, where the covariance matrix $\Lambda - V^T \Sigma V$, is a symmetric, diagonal matrix with the form:

$$\Lambda - \begin{bmatrix} \lambda_1\ 0 \ldots 0 \\ 0\ \lambda_2 \ldots 0 \\ \cdot \\ \cdot \\ \cdot \\ 0\ 0 \ldots \lambda_J \end{bmatrix}$$

$$\textit{and}\quad \lambda_i\ -\ Var\ Y_i\ .$$

Since the mean vector is assumed to have equal components, Y_i is distributed $N(\mu C_i,\ \lambda_i)$, where C_i is the *ith* row sum of V^T. Therefore, Y_i/C_i is distributed $N(\mu, \frac{\lambda_i}{C_i^2})$, *since* $Var(aY) - a^2 Var Y$.

Here Y_i/C_i are point estimates of the mean, each with different variances. These point estimates are useful for confidence interval construction if some linear combination of them can be determined such that the point estimates are unbiased estimates of the true mean and have minimum variance. So, weights a_i must be chosen such that these useful conditions hold i.e.

We want $E\,[\sum_{i=1}^{J} a_i \frac{Y_i}{C_i}] = \sum_{i=1}^{J} a_i \mu$. This implies $\sum_{i=1}^{J} a_i = 1$. We also want $Var\,[\sum_{i=1}^{J} a_i \frac{Y_i}{C_i}] = \sum_{i=1}^{J} a_i^2 V_i$ to be minimized, where $V_i = Var\,\frac{Y_i}{C_i} = \frac{\lambda_i}{C_i^2}$.

Therefore, the optimization problem then becomes:

$$minimize \sum_{i=1}^{J} a_i^2 V_i \text{ such that } \sum_{i=1}^{J} a_i = 1 \,,$$

which is solved by Lagrangian relaxation.

The Lagrangian problem is solved as follows:

$$\min z = \sum_{i=1}^{J} a_i^2 V_i + \delta(1 - \sum_{i=1}^{J} a_i)$$

Taking the derivatives with respect to a_i and δ yields:

$$\frac{dz}{da_i} = 2a_i - \delta$$

$$\frac{dz}{d\delta} = 1 - \sum_{i=1}^{J} a_i$$

Setting the derivatives equal to zero yields the following:

$$a_i = \frac{\delta}{2V_i}$$

$$\sum_{i=1}^{J} a_i = 1$$

Solving the above two equations yields expressions for a_i and δ in terms of V_i, i.e.

$$\delta = \frac{2}{\sum_{i=1}^{J} 1/V_i}$$

$$a_i = \frac{1/V_i}{\sum_{i=1}^{J} 1/V_i}$$

Taking the second derivatives confirm that the critical points will be minimum since the second partials are strictly greater than 0.

Consider the estimate a_iY_i/C_i, which is distributed $N\,(\mu,\ \lambda_i/C_i^2)$. The minimum variance of this estimate is $a_i^2\lambda_i/C_i^2$. In order to obtain a larger, more useful, estimate of the variance that is suitable for confidence interval construction, the C_i^2 term in the denominator must decrease. If the term is replaced by the average of the C_i^2's then the denominator reduces to 1 for every i. The following illustrates this finding:

Recall that the symmetric covariance matrix Σ is diagonalized into $V^T\Lambda V$, where V is the orthogonal matrix of eigenvectors. Since V is orthogonal, $V^T=V^{-1}$. Therefore, the square of the orthogonal matrix is $V^TV = VV^T = VV^{-1} = I$, where I is the identity matrix. Also, recall that C_i, $i=1,\ldots,J$, is a vector consisting of the *ith* rowsums of V^T, i.e.

$$C_i = (C_1,...,C_J) = V^T E^T = EV$$

where $E = (1,1,...1)$. Therefore,

$$\begin{aligned}\sum_{i=1}^{J} C_i^2 &= (C_1,...,C_J)\ (C_1,...,C_J)^T \\ &= (EV)\ (EV)^T \\ &= (EV)\ (V^T E^T) \\ &= EVV^T E^T \\ &= EIE^T \\ &= EE^T \\ &= J.\end{aligned}$$

Hence, $\frac{1}{J}\sum_{i=1}^{J} C_i^2 = 1$ is the average of the C_i^2 terms. If we replace the C_i^2 term in the variance estimate by its average, then another estimate of the variance is $\sum_{i=1}^{J} a_i^2\lambda_i$, which is not minimum. The proposed confidence interval then becomes:

$$\sum_{i=1}^{J} a_i \frac{Y_i}{C_i} \pm Z_{1-\alpha}\sqrt{\sum_{i=1}^{J} a_i^2\lambda_i}$$

where a_i is determined as before.

APPENDIX B

DESCRIPTION OF TIMESLAB SOFTWARE

Timeslab, a time series analysis laboratory, is an interactive, command-driven graphics software developed by Dr. H. Joseph Newton and detailed in his text, Newton [1988]. The software grew out of a series of lecture notes, computer programs and Fortran library subroutines that were done by Dr. Newton. It consists of about 150 commands that perform various data manipulations as reading, plotting, transforming and forecasting. It even has a built-in text editor which easily allows for interactive commands to be entered at the keyboard or for a batched series of commands to be written to a file called a macro. This appendix provides a guide to the commands that were used in constructing macros needed to conduct the research.

Timeslab requires an IBM-compatible personal computer with 640 kilobytes of random access memory and with either a color graphics adapter or an enhanced graphics adapter. Either a color or a monochrome monitor may be used to display pixel graphics. For this research, an IBM-compatible computer with a Intel 486 microprocessor capable of running at speeds of 25 to 33 kilohertz and a color monitor with graphics capability were used.

The basic rules for data are as follows:

(1) Variable names may be up to 15 characters long.

(2) Data may be any of four types: integer, real scaler, character, real array. There may be a maximum of 60 variables of each type. The total number of elements for all arrays must not be greater than 10,000.

(3) Each array may have at most a 40-character label assigned to it which is used to identify listings and plots of arrays.

(4) Variable names beginning with the letters *i* through *n* are considered to be integers unless such a variable is assigned a real value by including a decimal point.

The arithmetic operations of addition, subtraction, multiplication, division, and exponentiation are evaluated according to the following rules:

(1) Evaluate successively all expressions contained within the innermost curly bracket first, working from left to right.

(2) After all expressions enclosed within the innermost curly brackets have been evaluated, exponentiations, products, and additions are, respectively, done in order from left to right.

(3) Operations on integers result in an integer. Operations on any combination of reals and integers result in a real.

Timeslab commands consist of a command name and a number of internal arguments enclosed within parentheses. The internal arguments are very similar to the arguments that are passed from a Fortran subroutine to a main program. Timeslab has commands for manipulating correlations, for varying the environment of input media, for high resolution graphics, for input/output control, for displaying and transforming data, for conducting math operations, for manipulating matrices, and for simulating analytical models. The general format of a command is:

Y = command name(arg1,...,argj [,argk,...,argn])

where the arguments in square brackets are optional arguments that are to be included in the command's argument listing, as needed, and without the square brackets. Typical Timeslab commands that were used in this research are summarized on the next page.

It is Timeslab's useful capability of allowing a series of commands to be written to a macro file to be executed batch-style at any desired time that makes it a very flexible programming language. It is quite capable of performing many of the complex, repetitive calculations that are normally associated with high level programming languages. For example, looping constructs may be added to macros in order to provide logical branching. Comments may be included in macros to enhance documentation. Any word or statement that is written on a macro line and preceded by a semi-colon is useful as a branching point for a logical branching construct or as a documentary comment that the software will ignore when it executes the macro.

B.1 Summary of Timeslab Commands

This section summarizes the Timeslab commands that were used in the research.

ABS - absolute value of a scaler or an array.

ARDT - simulate an AR process.

ARMADT - simulate an ARMA process.

CLEAN - deletes all or specified variables.

CORR - calculate the variance and correlations of a data set.

CUM - calculate cumulative sums or cumulative averages of an array.

EDIT - edit files using the text editor.

EIG - find the eigenvalues and (optional) eigenvectors of a matrix.

ENDIF - ends a programming logical branch or looping routine in a macro.

EXTRACT - form an array by taking from elements of another array.

GOTO - a programming branching statement.

HELP - invoke the on-line help screen.

IF - starts a programming loop.

INFO - list all variables.

LABEL - provide a label for an array.

LINE - form an array of equally spaced real numbers.

LIST - display the value of variables to an output device.

MACRO - start executing the series of commands written in a data file.

MCHOL - gets the modified Cholesky decomposition of a symmetric matrix.

MINV - invert a matrix.

MMULT - multiply matrices.

PAUSE - stops to await a response from the user.

PLOT - provide graphs of arrays.

PLOTOFF - switch from graphics to text mode.

PLOTON - switch from text to graphics mode.

PRINTSEL - selects a type of printer.

PROMPTOFF - after this command, do not display commands being executed.

PROMPTON - after this command, resume displaying commands being executed.

QUIT - terminate Timeslab and return to DOS.

READ - read a data set from a file.

SAVE - save an array to a file.

SUBMNS - calculate and add in seasonal means of a data set.

TOEPL - form a symmetric (Toeplitz) matrix from an array of real numbers.

TRANS - transpose a matrix.

APPENDIX C

COMPUTER PROGRAMS

```
;;  MM1.MAC: A Timeslab macro to get the batch means of an MM1 process and
;;          determine the proposed 90% confidence interval.
;;
;;  input:      n=number of observations, seed=for random # generator
;;              zval=normal table entry for confidence level
;;              ro=confidence level, nb=number of batches, mu0=true mean
;;              nsamps=number of replicates
;;
PAUSE
;start
;*********************************************************************
;
;                          INPUT VALUES
;
nsamps=100                  ;number of replicates
zval=1.645                  ;for 90% conf int
ro=.9                       ;correlation level
mu0=9                       ;theoretical value for rho
n=10240                     ;sample size per replicate
seed=5                      ;random number seed
nb=10                       ;number of batches per sample size
;
;*********************************************************************
xnb:=nb                     ;real value for integer variable
mb=nb-1                     ;number of lag correlations
PROMPTOFF                   ;turn off commands sent to screen
```

```
;********************************************************************
;
;                         INITIALIZATION OF VARIABLES
;
nskip=0                        ;keeps track of where in SIMAN data file to read
data
hi=0.                          ;summer for # c.i.'s biased hi (BM Method)
hhi=0.                         ;summer for # c.i.'s biased hi (PC Method)
lo=0.                          ;summer for # c.i.'s biased lo (BM)
llo=0.                         ;summer for # c.i.'s biased lo (PC)
sum=0.                         ;summer to calc half-width (BM)
ssum=0.                        ;summer to calc half-width (PC)
sumsq=0.                       ;summer to calc sd half-width (BM)
ssumsq=0.                      ;summer to calc sd half-width (PC)
summse=0.                      ;summer to calc MSE (BM)
ssummse=0.                     ;summer to calc MSE (PC)
sumbias=0.                     ;summer to calc avg bias (BM)
sumvar=0.                      ;summer to calc avg variance (BM)
ssumbias=0.                    ;summer to calc avg bias (PC)
ssumvar=0.                     ;summer to calc avg variance (PC)
in=0.                          ;summer to calc # of corrl'd batches (BM)
im=0.                          ;summer to calc # of corrl'd batches (PC)
;
;********************************************************************
;s1                            ;entry point for branching loop
;
rep=nskip+1                    ;counter for the # of replicates
LIST(rep)                      ;tell the user which sample is being done
;
;********************************************************************
;
;
```

```
;                         BATCH MEANS METHOD
;
read(bat10295.dat,x,nb,nskip)     ;reads from an ASCII Fortran file
xmean=submns(x,nb,1,xbar)         ;batch means of batch size nb
rhox=corr(x,nb,mb,0,1,r0x,per)    ;determines correlation between batches
r0x={xnb/mb}*r0x                  ;unbiased est of variance
covx=r0x*rhox                     ;covariance
rhox=abs(rhox,mb)
if(rhox[1] .gt. .5)               ;Law & Carson's check for uncorrelated batches
in=in+1
endif
sd={r0x/nb}^.5                    ;gets std dev
half=zval*sd                      ;half-width of interval
biassq={xbar-mu0}^2               ;square of bias
sumbias=sumbias+biassq^.5
sumvar=sumvar+r0x
mse=biassq+r0x                    ;calc MSE
summse=summse+mse
sum=sum+half
sumsq=sumsq+half*half
ll=xbar-half                      ;lower conf limit
ul=xbar+half                      ;upper conf limit
if(ll .gt. mu0)                   ;checks to see if limits cover true mean
hi=hi+1
endif
if(ul .lt. mu0)
lo=lo+1
endif
ci=<ll,ul>                        ;puts in array format
data11=<xbar,r0x,half,ci,mse>
label(data11)='Batched, n=#n#, rho=@ro@, rep=#rep#'
;save(data11,6,test.out)
```

```
;*************************************************************************
;s3
;
;*************************************************************************
;
;                    PRINCIPAL COMPONENT METHOD
;
ax=toepl(covx,r0x,nb)               ;makes symmetrical covariance matrix
val=eig(ax,nb,100,ier,vec)          ;calc eigenvalues and eigenvectors
tvec=trans(vec,nb,nb)                ;transpose of eigenvector matrix
y1=mmult(tvec,x,nb,nb,1)            ;transform of original batched data
ones=line(nb,1,0)                   ;an column of ones
rowsum1=mmult(tvec,ones,nb,nb,1);rowsum of eigenvector matrix-transpose
rs1=extract(rowsum1,nb,0,ne,ny)     ;id's nonzero rowsums
y11=extract(y1,rs1,ny)              ;extracts xformed data for nonzero rowsums
rwsum1=extract(rowsum1,rs1,ny)      ;extracts nonzero rowsums
mu1=y11/rwsum1                      ;uncorrelated, though highly variable, est's of mu0
nny=ny-1                            ;# of lag correlations to be calculated
vali=extract(val,rs1,ny)            ;extracts eigenvalues for nonzero rowsums
sqrwsum1=rwsum1*rwsum1              ;rowsum squared
vari=vali/sqrwsum1                  ;theoretical variance of mu0 est's
invari=1/vari                       ;inverse of theoretical variance
sumvari=cum(invari,ny,1)            ;cum sum of inverse
smvari=extract(sumvari,ny,ny)       ;extracts last element of previous array
ai=invari/smvari                    ;linear unbiased weights
muuavg2=mmult(ai,mu1,1,ny,1)        ;unbiased est of mu0
muavg2=extract(muuavg2,1,1)         ;extracts last element of previous array
zz=ai*mu1                           ;less variable est's of mu0 used for variance est
rr=corr(zz,ny,nny,0,1,variante,per) ;calc correlations of mu0 estimates
xny:=ny                             ;real value of integer variable
sqai=ai*ai
variant=mmult(sqai,vali,1,ny,1)
```

```
variance=extract(variant,1,1)        ;unbiased est of variance of mu0
rr=abs(rr,nny)
if(rr[1] .gt. .5)                    ;checks for corrl'd batches
im=im+1
endif
std={variance}^.5                    ;std dev of unbiased est of mean or std error
bbiasq={muavg2-mu0}^2                ;calc bias
ssumbias=ssumbias+bbiasq^.5
ssumvar=ssumvar+variance
mmse=bbiasq+variance                 ;calc MSE
ssummse=ssummse+mmse
hhalf=zval*std                       ;calc half-width
ssum=ssum+hhalf
ssumsq=ssumsq+hhalf*hhalf
lll=muavg2-hhalf                     ;lower conf limit
uuu=muavg2+hhalf                     ;upper conf limit
if(lll .gt. mu0)                     ;check to see if limits cover true mean
hhi=hhi+1
endif
if(uuu .lt. mu0)
llo=llo+1
endif
confi=<lll,uuu>
data21=<muavg2,variance,hhalf,confi,mmse>
label(data21)='Eig mm1, n=#n#, rho=@ro@, rep=#rep#'
;save(data21,6,test.out)
;
;**********************************************************************
;
if(rep .ge. nsamps, theend)          ;check if max replicates
nskip=nskip+1                        ;if not, increase reps by 1
goto(s1)                             ;and go to loop again
```

```
;theend                             ;else, do the following calc's
;
;**********************************************************************
;
;                    CALCULATE PERFORMANCE MEASURES
;
label(ax)='MM1,n=#n#,nb=#nb#,rho=@ro@,reps=#rep#'
list(ax,0,'mm1.out')                ;places label of previous line in output file
label(ax)='Batched Confidence Interval Data'
list(ax,0,'mm1.out')                ;places label of previous line in output file
cov={rep-hi-lo}/rep                 ;calc nominal coverage (BM)
ccov={rep-hhi-llo}/rep              ;calc nominal coverage (PC)
hcov=zval*{cov*{1-cov}/rep}^.5 ;calc half width of coverage int (BM)
hhcov=zval*{ccov*{1-ccov}/rep}^.5 ;calc half width of coverage int (PC)
data1=<cov,hcov>                    ;makes an array of 2 elements
label(data1)='90% Cov Int=@cov@ +- @hcov@'
list(data1,0,'mm1.out')             ;places label of previous line in output file
avghalf=sum/rep                     ;calc avg half-width (BM)
aavghalf=ssum/rep                   ;calc avg half-width (PC)
varhalf={rep*sumsq-sum*sum}/{rep*{rep-1}};calc var of half-width (BM)
vvarhalf={rep*ssumsq-ssum*ssum}/{rep*{rep-1}};calc var of half-width (PC)
sdhalf=varhalf^.5                   ;calc sd of avg half-width(BM)
ssdhalf=vvarhalf^.5                 ;calc sd of avg half-width (PC)
data2=<avghalf,sdhalf>
label(data2)='Half-width is @avghalf@ +- @sdhalf@'
list(data2,0,'mm1.out')
avgbias=sumbias/rep                 ;calc avg bias (BM)
avgvar=sumvar/rep                   ;calc avg var (BM)
data21=<avgbias,avgvar>
label(data21)='Avg Bias=@avgbias@ & Avg Var=@avgvar@'
list(data21,0,'mm1.out')
```

```
data3=<hi,lo>
label(data3)='@hi@ Biased Hi and @lo@ Biased Lo'
list(data3,0,'mm1.out')
avgmse=summse/rep               ;calc avg MSE (BM)
data31=<avgmse>
label(data31)='Avg Mse = @avgmse@ '
list(data31,0,'mm1.out')
inn=<in>
label(inn)='No. Corr. Batches is @in@'
list(inn,0,'mm1.out')
label(rhox)='Eigvalue Conf Interval Data'
list(rhox,0,'mm1.out')
data4=<ccov,hhcov>
label(data4)='90% Cov Int=@ccov@ +- @hhcov@'
list(data4,0,'mm1.out')
data5=<aavghalf,ssdhalf>
aa=aavghalf
ss=ssdhalf
label(data5)='Half-width is @aa@ +- @ss@'
list(data5,0,'mm1.out')
aavgbias=ssumbias/rep           ;calc avg bias (PC)
aavgvar=ssumvar/rep             ;calc avg var (PC)
data51=<aavgbias,aavgvar>
label(data51)='Avg Bias=@aavgbias@ & Avg Var=@aavgvar@'
list(data51,0,'mm1.out')
data6=<hhi,llo>
label(data6)='@hhi@ Biased hi and @llo@ Biased Lo'
list(data6,0,'mm1.out')
aavgmse=ssummse/rep             ;calc avg MSE (PC)
ams=<aavgmse>
label(ams)='Avg Mse = @aavgmse@ '
list(ams,0,'mm1.out')
```

```
imm=<im>
label(imm)='No. Corr Batches is @im@'
list(imm,0,'mm1.out')
it=<0>
label(it)=' '
list(it,0,'mm1.out')
;
;                               THE END
;************************************************************************
```

```
      PROGRAM BATCHING
C
C This is a Fortran program to batch the flow times of customers from an
C M/M/1 queue.  Given a total sample size of n, get NBATCH batches of
C size n/NBATCH each. Do this for NREPS replicates of the experiment by
C reading the flowtimes exported from a SIMAN-generated output
C file.
C
C Variables defined:  MM1.90=the Siman output file of flowtimes in form (X,Y)
C                     BAT90.DAT=the Fortran batched files to be created.
C                     NN=a counter of the run replicates (max: 100)
C                     NBSIZE=batch size
C                     NBATCH=number of batches
C                     NREPS=number of replicates
C                     I=a counter for batches up to NBATCH
C                     N=a counter of observations added up to NBSIZE
C                     XN=real variable for integer N; useful for division
C                     SUM=a variable to sum up flowtimes
C                     X=a Siman variable for event time
C                     Y=a Siman variable for flowtime value
C                     AVG=batched average of NBSIZE observations
C
      DIMENSION AVG(100)
      OPEN (UNIT=4,FILE='MM1.90',STATUS='OLD')
      OPEN (UNIT=7,FILE='E:BAT90.DAT')
      NN=1
      NBSIZE=32
      NBATCH=10
      NREPS=100
      READ (4,8)
    8 FORMAT (/,/,/)
```

```
10 WRITE (7,15) NN
15 FORMAT (1X,'M/M/1, rho=.9, N=320, Repl ',I3,/1X,'10 (10F8.5) ')
  WRITE (*,*) NN,NREPS
  I=1
20 N=0
  SUM=0.
25 IF (N .LT. NBSIZE) THEN
    READ (4,*) X,Y
    IF (X .LT. 0) GO TO 25
    SUM=SUM+Y
    N=N+1
    GO TO 25
28 ENDIF
  AVG(I)=SUM/FLOAT(N)
  IF (I .GE. NBATCH) GO TO 30
  I=I+1
  GO TO 20
30 WRITE(7,35) (AVG(I),I=1,NBATCH)
35 FORMAT(10F8.5)
  IF (NN .GE. NREPS) GO TO 40
  NN=NN+1
  GO TO 10
40 REWIND(4)
  CLOSE(4)
  REWIND(7)
  CLOSE(7)
  STOP
  END
```

```
!       MM1.EXP:  SIMAN experimental frame for M/M/1 customer waiting times
!
begin;
        create:ex(1,1):mark(1);
        queue,1;
        seize:server;
        delay:ex(2,2);
        release:server;
        tally:1,int(1);
        count:1,1:dispose;
end;
```

```
!       MM1.MOD:  SIMAN model frame for M/M/1 customer waiting times
!
begin;
  project, mm1, tomfree, 7/30/91;
  discrete, 1000,1,1;
  resources:1,server,1;
  seeds:1,5,no:2,500,no;
  parameters:1,1:2,.9;
  dstat:1,nq(1),the queue;
  counters:1,customers,162700;
  tallies:1,flow, 90;
  replicate,100,0,162700,YES,YES,160000;
end;
```

```
;;  AR1.MAC: macro to generate nsamps realizations of length
;;           n from an AR(1) process having coefficient alpha,
;;           and find the confidence interval for mu=100 for
;;           each one.
;;
;;  INPUT:   n=# of observations, seed=for random # generator
;;           zval=normal table entry for confidence level
;;           ro=confidence level, nb=number of batches, mu0=true mean
;;           nsamps=number of replicates, alpha=AR coefficient
;;
PAUSE
;start
;*********************************************************************
;
;                         INPUT VALUES
;
alpha=<-.9>
nsamps=100
n=40
nb=40
ro=.9
mu0=100
seed=5
zval=1.645
;
;*********************************************************************
x=wn(seed,100)                  ;warm up the rand # generator
xnb:=nb
mb=nb-1
PROMPTOFF                       ;turn off prompt
;
;*********************************************************************
```

```
;
;                        INITIALIZATION OF VARIABLES
;
nskip=0                       ;keeps track of where to read in the data file
hi=0.                         ;summer for # c.i.'s biased hi (BM)
hhi=0.                        ;summer for # c.i.'s biased hi (PC)
lo=0.                         ;summer for # c.i.'s biased lo (BM)
llo=0.                        ;summer for # c.i.'s biased lo (PC)
sum=0.                        ;summer to calc half-width (BM)
ssum=0.                       ;summer to calc half-width (PC)
sumsq=0.                      ;summer to calc std dev of half-width (BM)
ssumsq=0.                     ;summer to calc std dev of half-width (PC)
summse=0.                     ;summer to calc avg MSE (BM)
ssummse=0.                    ;summer to calc avg MSE (PC)
sumbias=0.                    ;summer to calc avg bias (BM)
sumvar=0.                     ;summer to calc avg variance (PC)
ssumbias=0.                   ;summer to calc avg bias (PC)
ssumvar=0.                    ;summer to calc avg variance (PC)
in=0.                         ;summer to calc # of corrl'd batches (BM)
im=0.                         ;summer to calc # of corrl'd batches (PC)
;
;**********************************************************************
;s1                           ;entry point for branching loop
;
rep=nskip+1                   ;counter for the # of replicates
LIST(rep)                     ;tell the user which sample is being done
;s2
;
;**********************************************************************
;
;
```

```
;                          BATCH MEANS METHOD
;
alpha=<-.9>                      ;re-define AR coefficient for each replicate
x=ARDT(alpha,1,100,0,n,ier,r0)   ;generate AR(1) realization,var=100
x=SUBMNS(x,n,1,mu0,2)            ;add in mean of 100
macro(batch)                     ;call to macro to batch AR data
rhox=corr(xbar,nb,mb,0,1,r0x,per) ;determines correlation between batches
r0x={xnb/mb}*r0x                 ;unbiased est of variance
covx=r0x*rhox                    ;covariance
rhox=abs(rhox,mb)
if(rhox[1] .gt. .5)              ;Law & Carson's check for uncorrelated batches
in=in+1
endif
xmu=cum(xbar,nb,2)               ;cumulative average of batch means
xmubar=extract(xmu,nb,nb)        ;extract last element of previous array
sd={r0x/nb}^.5                   ;std error
half=zval*sd                     ;half-width
biassq={xmubar-mu0}^2            ;bias squared
sumbias=sumbias + biassq^.5
sumvar=sumvar+r0x
mse=biassq+r0x                   ;MSE
summse=summse+mse
sum=sum+half
sumsq=sumsq+half*half
ll=xmubar-half                   ;lower conf limit
ul=xmubar+half                   ;upper conf limit
if(ll .gt. mu0)                  ;checks to see if limits cover true mean
hi=hi+1
endif
if(ul .lt. mu0)
lo=lo+1
endif
```

```
ci=<ll,ul>
data11=<xmubar,r0x,half,ci,mse>
label(data11)='Batched, n=#n#, rho=@ro@, rep=#rep#'
;save(data11,6,test.out)
;
;***********************************************************************
;s3
;
;***********************************************************************
;
;                         PRINCIPAL COMPONENT METHOD
;
ax=toepl(covx,r0x,nb)             ;makes symmetrical covariance matrix
val=eig(ax,nb,100,ier,vec)        ;calc eigenvalues and eigenvectors
tvec=trans(vec,nb,nb)             ;transpose of eigenvector matrix
y1=mmult(tvec,xbar,nb,nb,1)       ;transform of original batched data
ones=line(nb,1,0)                 ;a column of ones
rowsum1=mmult(tvec,ones,nb,nb,1);rowsum of eigenvector matrix-transpose
rs1=extract(rowsum1,nb,0,ne,ny)   ;id's nonzero rowsums
y11=extract(y1,rs1,ny)            ;extracts xformed data for nonzero rowsums
rwsum1=extract(rowsum1,rs1,ny)    ;extracts nonzero rowsums
mu1=y11/rwsum1                    ;uncorrelated, though highly variable, est's of mu0
nny=ny-1                          ;# of lag correlations to be calculated
vali=extract(val,rs1,ny)          ;extracts eigenvalues for nonzero rowsums
sqrwsum1=rwsum1*rwsum1            ;rowsum squared
vari=vali/sqrwsum1                ;theoretical variance of mu0 est's
invari=1/vari                     ;inverse of theoretical variance
sumvari=cum(invari,ny,1)          ;cumulative sum of inverse
smvari=extract(sumvari,ny,ny)     ;extracts last element of previous array
ai=invari/smvari                  ;linear unbiased weights
muuavg2=mmult(ai,mu1,1,ny,1)      ;unbiased est of mu0
muavg2=extract(muuavg2,1,1)       ;extracts last element of previous array
```

```
zz=ai*mu1                          ;less variable est's of mu0 used for variance est
rr=corr(zz,ny,nny,0,1,variante,per) ;calc variance of mu0 estimates
xny:=ny                            ;real value for integer variable
sqai=ai*ai
variant=mmult(sqai,vali,1,ny,1)
variance=extract(variant,1,1)      ;est of variance
rr=abs(rr,nny)
if(rr[1] .gt. .5)                  ;checks for corrl'd batches
im=im+1
endif
std={variance}^.5                  ;std error
bbiasq={muavg2-mu0}^2
ssumbias=ssumbias+bbiasq^.5        ;bias square
ssumvar=ssumvar+variance
mmse=bbiasq+variance               ;MSE
ssummse=ssummse+mmse
hhalf=zval*std                     ;half-width
ssum=ssum+hhalf
ssumsq=ssumsq+hhalf*hhalf
lll=muavg2-hhalf                   ;lower conf limit
uuu=muavg2+hhalf                   ;upper conf limit
if(lll .gt. mu0)                   ;check to see if limits cover true mean
hhi=hhi+1
endif
if(uuu .lt. mu0)
llo=llo+1
endif
confi=<lll,uuu>
data21=<muavg2,variance,hhalf,confi,mmse>
label(data21)='Eig MM1, n=#n#, rho=@ro@, rep=#rep#'
;save(data21,6,test.out)
;
```

```
;**********************************************************************
;
if(rep .ge.nsamps, theend)
nskip=nskip+1
clean(arrays)
goto(s1)
;theend
;
;**********************************************************************
;
;                 CALCULATE PERFORMANCE MEASURES
;
label(ax)='AR1,n=#n#,nb=#nb#,rho=@ro@,reps=#rep#'
list(ax,0,'ar1.out')              ;places label of previous line in output file
label(ax)='Batched Confidence Interval Data'
list(ax,0,'ar1.out')              ;places label of previous line in output file
cov={rep-hi-lo}/rep               ;calc nominal coverage (BM)
ccov={rep-hhi-llo}/rep            ;calc nominal coverage (PC)
hcov=zval*{cov*{1-cov}/rep}^.5 ;calc half-width of coverage interval (BM)
hhcov=zval*{ccov*{1-cov}/rep}^.5 ;calc half-width of coverage interval (PC)
data1=<cov,hcov>                  ;makes an array of 2 elements
label(data1)='90% Cov Int=@cov@ +- @hcov@'
list(data1,0,'ar1.out')           ;places label of previous line in output file
avghalf=sum/rep                   ;calc avg half-width (BM)
aavghalf=ssum/rep                 ;calc avg half-width (PC)
varhalf={rep*sumsq-sum*sum}/{rep*{rep-1}} ;calc var of half-width (BM)
vvarhalf={rep*ssumsq-ssum*ssum}/{rep*{rep-1}} ;calc var of half-width (PC)
sdhalf=varhalf^.5                 ;calc sd of avg half-width (BM)
ssdhalf=vvarhalf^.5               ;calc sd of avg half-width (PC)
data2=<avghalf,sdhalf>
label(data2)='Half-width is @avghalf@ +- @sdhalf@'
list(data2,0,'ar1.out')
```

```
avgbias=sumbias/rep                ;calc avg bias (BM)
avgvar=sumvar/rep                  ;calc avg var (PC)
data21=<avgbias,avgvar>
label(data21)='Avg Bias=@avgbias@ & Avg Var=@avgvar@'
list(data21,0,'ar1.out')
data3=<hi,lo>
label(data3)='@hi@ Biased Hi and @lo@ Biased Lo'
list(data3,0,'ar1.out')
avgmse=summse/rep                  ;calc avg MSE (BM)
data31=<avgmse>
label(data31)='Avg Mse = @avgmse@'
list(data31,0,'ar1.out')
inn=<in>
label(inn)='No. Corr. Batches is @in@'
list(inn,0,'ar1.out')
label(ax)='Eigvalue Conf Interval Data'
list(ax,0,'ar1.out')
data4=<ccov,hhcov>
label(data4)='90% Cov Int=@ccov@ +- @hhcov@'
list(data4,0,'ar1.out')
data5=<aavghalf,ssdhalf>
aa=aavghalf
ss=ssdhalf
label(data5)='Half-width is @aa@ +- @ss@'
list(data5,0,'ar1.out')
aavgbias=ssumbias/rep              ;calc avg bias (PC)
aavgvar=ssumvar/rep                ;calc avg variance (PC)
data51=<aavgbias,aavgvar>
label(data51)='Avg Bias=@aavgbias@ & Avg Var=@aavgvar@'
list(data51,0,'ar1.out')
data6=<hhi,llo>
label(data6)='@hhi@ Biased Hi and @llo@ Biased Lo'
```

```
list(data6,0,'ar1.out')
aavgmse=ssummse/rep                ;calc avg MSE (PC)
ams=<aavgmse>
label(ams)='Avg Mse = @aavgmse@'
list(ams,0,'ar1.out')
imm=<im>
label(imm)='No. Corr. Batches is @im@'
list(imm,0,'ar1.out')
it=<0>
label(it)=' '
list(it,0,'ar1.out')
;
;                                  THE END
;**************************************************************************
```

```
;;
;;  ARMA.MAC: macro to generate nsamps realizations of length
;;          n from an ARMA(1,1) process having coefficient alpha and beta
;;          and find the confidence interval for mu=100 for
;;          each one.
;;
;;  INPUT:  n=number of observations, seed=for random # generator
;;              zval=normal table entry for confidence level, B=confidence level
;;              nb=number of batches, mu0=true mean, nsamps=number of reps
;;              alpha=AR portion of ARMA coefficient, p=AR order
;;              beta=MA portion of ARMA coefficient, q=MA order
;;
PAUSE
;start
;
;**********************************************************************
;
;                               INPUT VALUES
;
alpha=<-.6>
beta=<.6>
p=1
q=1
nsamps=100
n=2560
nb=10
B=99
zval=2.576
mu0=100
seed=5
;
;**********************************************************************
```

```
x=wn(seed,100)                ;warm up the seed
xnb:=nb                       ;real value of integer variable
mb=nb-1                       ;number of lag correlations to be calculated
PROMPTOFF                     ;turn off prompt
;
;**********************************************************************
;
;                   INITIALIZATION OF VARIABLES
;
nskip=0                       ;keeps track of where to read in data file
hi=0.                         ;summer for # c.i.'s biased hi (BM)
hhi=0.                        ;summer for # c.i.'s biased hi (PC)
lo=0.                         ;summer for # c.i.'s biased lo (BM)
llo=0.                        ;summer for # c.i.'s biased lo (PC)
sum=0.                        ;summer to calc avg half-width (BM)
ssum=0.                       ;summer to calc avg half-width (PC)
sumsq=0.                      ;summer to calc sd of avg half-width (BM)
ssumsq=0.                     ;summer to calc sd of avg half-width (PC)
summse=0.                     ;summer to calc avg MSE (BM)
ssummse=0.                    ;summer to calc avg MSE (PC)
sumbias=0.                    ;summer to calc avg bias (BM)
sumvar=0.                     ;summer to calc avg variance (BM)
ssumbias=0.                   ;summer to calc avg bias (PC)
ssumvar=0.                    ;summer to calc avg variance (PC)
in=0.                         ;summer to calc # of corrl'd batches (BM)
im=0.                         ;summer to calc # of corrl'd batches (PC)
;
;**********************************************************************
;
;s1
rep=nskip+1
LIST(rep)     ;tell the user which sample is being done
```

```
;***********************************************************************
;
;                          BATCH MEANS METHOD
alpha=<-.6>
beta=<.6>
x=ARMADT(alpha,beta,p,q,100,0,n,nb,rhod,r0,ier) ;ARMA(1,1) realization, var=100
x=SUBMNS(x,n,1,mu0,2)             ;add in mean of 100
macro(batch)                      ;call to macro to batch ARMA data
rhox=corr(xbar,nb,mb,0,1,r0x,per) ;determines correlation between batches
r0x={xnb/mb}*r0x                  ;unbiased est of variance
covx=r0x*rhox                     ;covariance
rhox=abs(rhox,mb)
if(rhox[1] .gt. .5)               ;Law & Carson's check for uncorrelated batches
in=in+1
endif
xmu=cum(xbar,nb,2)                ;cumulative averages of the batch means
xmubar=extract(xmu,nb,nb)         ;extracts last element of previous array
sd={r0x/nb}^.5                    ;std error of mean
half=zval*sd                      ;half-width
biassq={xmubar-mu0}^2             ;bias-squared
sumbias=sumbias + biassq^.5
sumvar=sumvar+r0x
mse=biassq+r0x                    ;MSE
summse=summse+mse
sum=sum+half
sumsq=sumsq+half*half
ll=xmubar-half                    ;lower conf limit
ul=xmubar+half                    ;upper conf limit
if(ll .gt. mu0)                   ;checks to see if limits cover true mean
hi=hi+1
endif
if(ul .lt. mu0)
```

```
lo=lo+1
endif
ci=<ll,ul>
data11=<xmubar,r0x,half,ci,mse>
label(data11)='Batch ARMA, n=#n#, nb=#nb#,rep=#rep#'
;save(data11,6,test.out)
;
;**********************************************************************
;
;**********************************************************************
;
;                        PRINCIPAL COMPONENT METHOD
;
ax=toepl(covx,r0x,nb)              ;makes symmetrical covariance matrix
val=eig(ax,nb,100,ier,vec)         ;calc eigenvalues and eigenvectors
tvec=trans(vec,nb,nb)              ;transpose of eigenvector matrix
y1=mmult(tvec,xbar,nb,nb,1)        ;transform of original batched data
ones=line(nb,1,0)                  ;a column of ones
rowsum1=mmult(tvec,ones,nb,nb,1) ;rowsum of eigenvector matrix-transpose
rs1=extract(rowsum1,nb,0,ne,ny)    ;id's nonzero rowsums
y11=extract(y1,rs1,ny)             ;extracts xformed data for nonzero rowsums
rwsum1=extract(rowsum1,rs1,ny)     ;extracts nonzero rowsums
mu1=y11/rwsum1                     ;uncorrelated, though highly variable, est's of mu0
nny=ny-1                           ;# of lag correlations to be calculated
vali=extract(val,rs1,ny)           ;extracts eigenvalues for nonzero rowsums
sqrwsum1=rwsum1*rwsum1             ;rowsum squared
vari=vali/sqrwsum1                 ;theoretical variance of mu0 est's
invari=1/vari                      ;inverse of theoretical variance
sumvari=cum(invari,ny,1)           ;cumulative sum of inverse
smvari=extract(sumvari,ny,ny)      ;extracts last element of previous array
ai=invari/smvari                   ;linear unbiased weights
muuavg2=mmult(ai,mu1,1,ny,1)       ;unbiased est of mu0
```

```
muavg2=extract(muuavg2,1,1)       ;extracts last element of previous array
zz=ai*mu1                         ;less variable est's of mu0 used for variance est
rr=corr(zz,ny,nny,0,1,variante,per) ;calc variance of  mu0 estimates
rr=abs(rr,nny)
if(rr[1] .gt. .5)                 ;checks for corrl'd batches
im=im+1
endif
xny:=ny                           ;real value of integer variable
sqai=ai*ai
variant=mmult(sqai,vali,1,ny,1)
variance=extract(variant,1,1)     ;est of variance
std={variance}^.5                 ;std error
bbiasq={muavg2-mu0}^2             ;bias squared
ssumbias=ssumbias+bbiasq^.5
ssumvar=ssumvar+variance
mmse=bbiasq+variance              ;MSE
ssummse=ssummse+mmse
hhalf=zval*std                    ;half-width
ssum=ssum+hhalf
ssumsq=ssumsq+hhalf*hhalf
lll=muavg2-hhalf                  ;lower conf limit
uuu=muavg2+hhalf                  ;upper conf limit
if(lll .gt. mu0)                  ;checks to see if limits cover true mean
hhi=hhi+1
endif
if(uuu .lt. mu0)
llo=llo+1
endif
confi=<lll,uuu>
data21=<muavg2,variance,hhalf,confi,mmse>
label(data21)='Eig ARMA, n=#n#, nb=#nb#, rep=#rep#'
;save(data21,6,test.out)
```

```
;
;**********************************************************************
if(rep .ge.nsamps, theend)          ;check if max replicates
nskip=nskip+1                       ;if not, increase reps by 1
clean(arrays)
goto(s1)                            ;go to loop again
;theend                             ;else, do the following calculations
;
;**********************************************************************
;
;                  CALCULATE PERFORMANCE MEASURES
;
label(ax)='ARMA,n=#n#,nb=#nb#,reps=#rep#'
list(ax,0,'arma.out')               ;places label of previous line in output file
label(ax)='Batched Confidence Interval Data'
list(ax,0,'arma.out')               ;places label of previous line in output file
cov={rep-hi-lo}/rep                 ;calc nominal coverage (BM)
ccov={rep-hhi-llo}/rep              ;calc nominal coverage (PC)
hcov=zval*{cov*{1-cov}/rep}^.5 ;calc half-width of coverage interval (BM)
hhcov=zval*{ccov*{1-ccov}/rep}^.5 ;calc half-width of coverage interval (PC)
data1=<cov,hcov>
label(data1)='@B@% Cov Int=@cov@ +- @hcov@'
list(data1,0,'arma.out')
avghalf=sum/rep                     ;calc avg half-width (BM)
aavghalf=ssum/rep                   ;calc avg half-width (PC)
varhalf={rep*sumsq-sum*sum}/{rep*{rep-1}} ;calc variance of avg half-width (BM)
vvarhalf={rep*ssumsq-ssum*ssum}/{rep*{rep-1}} ;calc variance of avg half-width
(PC)
sdhalf=varhalf^.5                   ;calc std dev of avg half-width (BM)
ssdhalf=vvarhalf^.5                 ;calc std dev of avg half-width (PC)
data2=<avghalf,sdhalf>
label(data2)='Half-width is @avghalf@ +- @sdhalf@'
```

```
list(data2,0,'arma.out')
avgbias=sumbias/rep                ;calc avg bias (BM)
avgvar=sumvar/rep                  ;calc avg variance (BM)
data21=<avgbias,avgvar>
label(data21)='Avg Bias=@avgbias@ & Avg Var=@avgvar@'
list(data21,0,'arma.out')
data3=<hi,lo>
label(data3)='@hi@ Biased Hi and @lo@ Biased Lo'
list(data3,0,'arma.out')
avgmse=summse/rep                  ;calc avg MSE (BM)
data31=<avgmse>
label(data31)='Avg Mse = @avgmse@'
list(data31,0,'arma.out')
inn=<in>
label(inn)='No. Corr. Batches is @in@'
list(inn,0,'arma.out')
label(ax)='Eigvalue Conf Interval Data'
list(ax,0,'arma.out')
data4=<ccov,hhcov>
label(data4)='@B@% Cov Int=@ccov@ +- @hhcov@'
list(data4,0,'arma.out')
data5=<aavghalf,ssdhalf>
aa=aavghalf
ss=ssdhalf
label(data5)='Half-width is @aa@ +- @ss@'
list(data5,0,'arma.out')
aavgbias=ssumbias/rep              ;calc avg bias (PC)
aavgvar=ssumvar/rep                ;calc avg variance (PC)
data51=<aavgbias,aavgvar>
label(data51)='Avg Bias=@aavgbias@ & Avg Var=@aavgvar@'
list(data51,0,'arma.out')
data6=<hhi,llo>
```

```
label(data6)='@hhi@ Biased Hi and @llo@ Biased Lo'
list(data6,0,'arma.out')
aavgmse=ssummse/rep              ;calc avg MSE (PC)
ams=<aavgmse>
label(ams)='Avg Mse = @aavgmse@'
list(ams,0,'arma.out')
imm=<im>
label(imm)='No. Corr. Batches is @im@'
list(imm,0,'arma.out')
it=<0>
label(it)=' '
list(it,0,'arma.out')
;
;                              THE END
;*********************************************************************
```

```
;;   BATCH.MAC:  macro to get batch means of AR1 and ARMA processes
;;
;;     input: n=sample size, nb=number of batches, x=data
;;
;start
nz=n
xbar=line(nb,0,0)
bsize=n/nb
i=1
;loop
avgs=cum(x,bsize,2)
xbar[i]=extract(avgs,bsize,bsize)
clean(avgs)
i=i+1
if(i .gt. nb, endloop)
n=n-bsize
ni=line(n,bsize,1)
x=extract(x,ni,n)
clean(ni)
goto(loop)
;endloop
```