1989

# The statistical optimal design of Shewhart control charts with supplementary stopping rules

Noel Artiles-Leon
*Iowa State University*

Order Number 8920108

# The statistical optimal design of Shewhart control charts with supplementary stopping rules

Artiles-Leon, Noel, Ph.D.

Iowa State University, 1989

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

# The statistical optimal design of Shewhart control charts with supplementary stopping rules

by

Noel Artiles-Leon

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1989

TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# I. INTRODUCTION

In 1931, Walter A. Shewhart originated the control chart for the attainment of statistical stability of a production process. Currently, control charts are used in a wide variety of production, research, and development environments. This chapter presents a summary of the main contributions and current state of the art in the area of control chart design. The literature review chapter covers the most important developments in the area of analysis, design, and optimization of control charts from statistical, semieconomical, and economical criteria in the last three decades. The following chapters address some open questions in the area of design of X-bar charts when "warning lines" are used as part of the control scheme. Mathematical and statistical tools necessary for answering these questions are discussed and numerical examples are presented to illustrate the relevance of this work.

The following paragraphs provide some background information on control charts: their uses, some of the most commonly used control schemes, and the primary considerations involved in the design of control charts.

## A. The Uses of Control Charts

Although the original intention of such control chart was to attain a state of statistical stability for a given process ("process

control"), since its introduction, many modifications have been suggested and other schemes have been introduced. Currently, Shewhart control charts are being used for at least one of the following different purposes.

1. Testing for statistical control

One of the uses of the control charts that was first contemplated is determining whether a process has achieved a state of statistical control. For this purpose, a statistic to be charted is selected depending on the process to be controlled and appropriate data are gathered and checked against trial control limits.

2. Maintaining current control

One of the many problems that arises in the applications of statistics in industry is the detection of changes in parameters specifying the quality of the output from a production process, so that some corrective action can be taken to restore the parameters to satisfactory values.

Many control charts are being used to give an alarm when it is believed that the process has gone out of statistical control. Control limits computed from a given standard are used to detect when a process, which is in control at certain target values of the distribution parameters, departs from those values. Shewhart recommended the use of $3\sigma$ limits as action limits, that is, rectifying

action should be taken if the observed value of the statistic being charted is plus or minus three or more standard deviations from its target value.

Little justification has been given for the selection of these limits and many alternatives of this control method have been introduced. One of these alternatives that is widely used is to call for corrective action when a certain number of points out of a specified number of observations fall outside of a predetermined "warning line."

3. Historical search

The visual record provided by a Shewhart chart is a great help in identifying when changes in the process characteristics occurred so the search for assignable causes is facilitated.

## B. Control Limits and Control Rules

Assume that a given process is in statistical control and that the distribution of the relevant statistic (sample mean, standard deviation, range, fraction defective,...) is known. Several schemes have been suggested to control a parameter of this distribution at a target level. Some of these schemes are:

1. Western Electric Company Statistical Quality Control Handbook (1956)

   Declare the process to be out of control if any of the following situations occurs:

   - o A single point falls outside of the $\pm 3\sigma$ limits.
   - o Two out of three successive points fall above the $+2\sigma$ limit.
   - o Two out of three successive points fall below the $-2\sigma$ limit.
   - o Four out of five successive points fall above the $+1\sigma$ limit.
   - o Four out of five successive points fall below the $-1\sigma$ limit.
   - o Eight successive points fall above the target value.
   - o Eight successive points fall below the target value.

2. Acheson J. Duncan (1974)

   Declare the process to be out of control if any of the following situations occurs:

   - o A single point falls outside of the $\pm 3\sigma$ limits.
   - o One or more points is in the vicinity of a warning limit.
   - o A run of seven or more points fall above or below the central line on the control chart.
   - o A run of 2 or 3 points fall outside of $\pm 2\sigma$ limits.
   - o A run of 4 or 5 points fall outside of $\pm 1\sigma$ limits.
   - o Cycles or other nonrandom patterns in the data.

3. Grant and Leavenworth (1988)

   Declare the process to be out of control if any of the following situations occurs:

- o A run of seven or more points fall above or below the central line on the control chart.
- o Ten of eleven consecutive points fall on the same side of the central line on the control chart.
- o Twelve of fourteen consecutive points fall on the same side of the central line on the control chart.
- o Fourteen of seventeen consecutive points fall on the same side of the central line on the control chart.
- o Sixteen of twenty consecutive points fall on the same side of the central line on the control chart.

4. Wetherill (1977)

Declare the process to be out of control if any of the following situations occurs:

- o A single point falls outside the $\pm 3\sigma$ limits.
- o Two points fall in a row outside of the $+2\sigma$ limit.
- o Two points fall in a row outside of the $-2\sigma$ limit.

## C. Selection of Control Limits and Control Rules

1. The Type-I Type-II error approach

Consider a control chart with the simple control rule: "Declare the process to be out of control if a single point falls above $x\sigma$ limit or below $-x\sigma$ limit." For a given x, there are two possible types of errors when control charts like this are used. The first occurs when

the process involved is in control but a point falls outside the control limits due to intrinsic process variation and randomness. Consequently, it is incorrectly concluded that the process is not in control, the process is stopped, an attempt to locate the cause of a nonexistent problem is made, and then a cost due to production lost and wasted time is incurred. This type of error is referred to as Type-I error.

The second error, referred to as Type-II error, occurs when the process involved is out of control but the sampled point falls within the control limits due to chance. As a result, it is incorrectly concluded that the process is in control and costs associated with any resulting increase in nonconforming output are incurred.

The size of the risk of a Type-I error, $\alpha$, depends only on the choice of the control limits; wider limits reduce the risk of this error and, as a result, this risk encourages the adoption of wide control limits. On the other hand, the risk of a Type-II error, $\beta$, is a function of the control limits and the degree to which the process is out of control; this risk encourages the adoption of narrow control limits.

The optimum control limits may be defined as those that minimize the total cost of making an error (the cost of a Type-I error plus the cost of a Type-II error). Consequently, if the cost of examining a process to identify the cause of a presumable out-of-control situation is high, wider limits should be adopted; conversely, if that cost is low, narrower limits should be selected. If the cost of nonconforming

output produced by a process is considerable, narrower control limits should be favored; otherwise, wider limits should be selected.

If the cost of a Type-I error and the cost of a Type-II error are about the same, wide control limits should be chosen, and attention should be given to decreasing the resulting risk of a Type-II error by increasing the sample size; in addition, to reduce the duration of any out-of-control situation which might occur, more frequent samples should be taken. Finally, if past experience with a process reveals that an out-of-control condition happens quite frequently, narrower control limits should be preferred because the large number of possibilities to make a Type-II error; on the other hand, if the rate at which the process goes out of control is low, wider limits will be favored.

Unfortunately, it seems that most organizations adopt $\pm 3\sigma$ limits as a matter of course and try to minimize the total cost by determining the optimal sample size and the best sample frequency.

2. The average run length (A.R.L.) approach

When complex control rules, such as those listed in Section I.B are used to control a process, risks of the first and second type are not the proper quantities to consider as criteria of how good a given control rule is. It is obvious that any control rule will give a signal some time even though the process is operating under control and it is also certain to give a signal some time after a deterioration in the

process has occurred. The relevant questions here are: "How often?" in the first case and "How quickly?" in the second one.

A quantity of fundamental interest is the average run length (A.R.L.) of a given set of control rules (or process inspection scheme). The average run length is the expected number of samples taken before action is taken when the quality of the process remains constant (not necessarily at the target value). A large number of samples is desired before receiving an out-of-control signal when the process is in control and a small number of samples ("fast response time") is desired when the process has departed from target. These are conflicting goals and in practice some kind of compromise between these two requirements has to be accepted.

If the cost of examining a process to identify the cause of a presumable out-of-control situation is high, a process inspection scheme with a very long A.R.L.'s should be selected. On the other hand, if the cost of nonconforming output produced by a process is considerable, a process inspection scheme with a very short off-target A.R.L. should be preferred.

Little research has been done on the determination of the A.R.L. of the inspection schemes described in Section 1.B. and their relative merits. However, J. I. Weindling (1967) has shown that for any control chart with warning and action limits, with a fixed sample size and process variance, the average run length is greatest when the process is in control and decreases strictly with an increase in the absolute value of a shift in the process mean. Consequently, an A.R.L.

curve, as a function of the process mean, has a shape similar to that of the operating characteristic of a two-tailed test, and may be used to compare the effectiveness of two control procedures.

3. The economic decision model approach

The design of control charts with respect to economic criteria has been the subject of considerable study during the last thirty years. Several different process models have been developed and applied to most of the major types of control charts.

In order to formulate an economic model for the design of a control chart, certain assumptions about the behavior of the production process are required. Most of the economic models that have been developed incorporate, explicitly or implicitly, the following assumptions to some degree.

a. The production process is assumed to be characterized by a single in-control state but it may have several out-of-control states (usually each of them associated with a particular type of assignable cause).

b. The residence (or waiting) time of the production process in each state (in control or out of control) is assumed to be exponentially distributed and the transitions between states are assumed to be instantaneous.

c. It is assumed that once a transition to an out-of-control state has occurred, the process can only be returned to the

in-control condition by an operator intervention after an action signal on the control chart.

Most of the economic models consider three categories of costs: the cost of sampling and testing, the costs associated with the production of nonconforming items, and the costs associated with the investigation of an action signal and with the correction of any assignable causes found. The cost of sampling and testing is assumed to be a linear function of the sample size; because the difficulty associated with the estimation of these costs a more complex relationship is probably inadequate. The cost associated with producing nonconforming items consists of the cost of repairing or replacing units covered by warranties or guarantees, losses resulting from product liability claims against the company, and market share reduction because of customer's dissatisfaction. The costs of searching for an assignable cause and possible correcting the process following an out-of-control signal have been modeled in two different ways. Some researchers suggest that the costs of investigating for false alarms are different from the costs of correcting assignable causes and, consequently, these two situations must be represented in the model by two different cost coefficients; in addition, some authors suggest using a different cost figure for each type of assignable cause. Other researchers argue that, since in most cases small shifts are difficult to find but easy to fix, while large shift are easy to find but difficult to correct, accuracy is not lost if only a single cost coefficient is used to represent this cost.

Economic models are generally formulated as a total cost per unit time function. The production, monitoring, and adjustment process may be thought of as a series of cycles. Each cycle begins with the process being in the in-control state; at some point during the cycle an assignable cause occurs and eventually an action signal is generated which leads to the discovery of the assignable cause; the cycle ends when the process is returned to the in-control state. If E(CT) denotes the expected duration of the cycle and E(CC) denotes the expected total cost incurred during the cycle, the expected cost per unit time is $E(C) = E(CC)/E(CT)$. Optimization techniques are then applied to this equation to find the economically optimum control chart design.

# II. LITERATURE REVIEW

In the last four decades, many volumes of journals have been filled with the exposition, application, modification, and economic design of control charts. Many papers have been published on innovative concepts on control chart techniques. It is impossible to include this vast amount of knowledge in just a few pages; consequently, this chapter will not include and deal with all of the many modifications and refinements of the techniques of control charts. However, we attempt to highlight the most important developments in this field, specially those related to Shewhart control charts and their ramifications, to the determination of average run lengths for control charts with "warning limits", and to the economic and semieconomic design of X-bar control charts.

## A. Semieconomic Design of Control Charts

Early work on the design of conventional Shewhart control charts was carried out by several researchers. One of the first papers addressing the effectiveness of Shewhart control charts was written by L. A. Aroian and H. Levene (1950). In this paper the authors assume a sampling scheme only with control lines in which there is a constant probability $\alpha$, at each decision point, of saying that is out of control (when the process is in control) and that, when the process suddenly goes out of control, there is a constant probability $\beta$ of

taking action at each decision point until remedial action has been taken. This framework is used to derive (for production at a constant level and for erratic production) how often the samples should be taken and the run-length distributions of Shewhart control charts used alone.

One of the first and most influential papers in the area of economic modeling of quality control systems is due to M. A. Girshick and H. Rubin (1952). They consider a system, producing items with a quality characteristic, that can be in one of four possible states; state 1 is considered to be the in-control state, state 2 is an out-of-control state, and states 3 and 4 are repair states. The output quality characteristic obviously depends upon the process state and it is described by a probability density function in states 1 and 2. When the system is in state 1 there is a constant probability of shifting into state 2; the systems is assumed to be not self-correcting and, consequently, once the process is in state 2, it has to go through one of the repair states in order to return to the in-control state. The residence time in the repair states are considered to be different but constant and discrete (the time unit used is defined as the time required to produce one item while the process is in state 1). The economic criterion used by Girshick and Rubin is to maximize the expected income from the process. Although this paper is of significant theoretical value, the use of the model in practice is limited because the optimal control rules are difficult to derive (they depend on the solution of integral equations).

Girshick and Rubin were the first authors to propose the expected income per unit time as a criterion for the design of quality control systems. Later researchers investigated generalized formulations of the Girshick-Rubin model, among them were I. R. Savage (1962), J. A. Bather (1963), S. M. Ross (1971), and C. C. White (1974). However, most of their findings are of theoretical interest only because, in general, they do not lead to simple process control rules.

Most of the work of early researchers could be classified as semi-economic design either because they failed to include all the relevant costs or because they did not use a formal optimization procedure to minimize the cost function. For example, G. H. Weiler (1954) suggested that, for an X-bar chart, the optimum sample size should minimize the total amount of inspection required to detect a specified shift. If the shift is from an in-control state, $\mu$, to an out-of-control state, $\mu+\delta\sigma$, the optimal sample size is inversely proportional to the square of $\delta$. Similar approaches were used by Weiler in studies of other control charts. Other semieconomic analyses were used by D. J. Cowden (1957) and N. N. Barish and N. Hauser (1963).

H. M. Taylor (1965) showed that control procedures with fixed sample size at constant time intervals cannot be optimal. However, these kind of rules are widely used in practice because of their administrative simplicity. Taylor suggested that sample size and sampling frequency should be determined based on the posterior probability that the process is in an out-of-control state.

## B. Economic Design of Control Charts

An early attempt to deal with a fully economical model was made by A. J. Duncan (1956). In this paper, Duncan, relying on the earlier work of Girshick and Rubin (1952), establishes a criterion that measures approximately the average net income of a process under surveillance of an X-bar chart when the process is subject to random shifts in the process mean. The quality control rule assumed is that an assignable cause is looked for whenever a point falls outside the control limits. The criterion given is for the case in which it is assumed that the production process is not stopped while the search for the assignable cause is in progress, nor is the cost of adjustment or repair and the cost of bringing the process back into a state of control, after the assignable cause is discovered, charged to the control chart program. Duncan's paper shows how to determine the sample size, the interval between samples, and the control limits that will yield approximately maximum average income. He also discusses numerical examples of optimum design to illustrate how variation in the various risk and cost factors affects the optimum.

Duncan assumes that the assignable cause occurs according to a Poisson process with a rate of $\theta$ occurrences per hour and shows that if samples are taken every h hours, the average time of occurrence within an interval between samples is

$$\tau = [\ 1 - (1+\theta h)\exp(-\theta h)\ ]/[\ \theta(1-\exp(-\theta h))\ ].$$

He also shows that the expected length of a cycle (in control/ detection of an assignable cause/ elimination of the assignable cause/ in control) is

$$E(T) = 1/\theta + h/(1-\beta) - \tau + gn + D ,$$

and that the expected loss per hour incurred by the process is

$$E(L) = \frac{C1 + C2\ n}{h} + \frac{C4\ [E(T)-1/\theta] + C3 + C3'\alpha\ \exp(-\theta h)/(1-\exp(\theta h))}{E(T)}$$

where

C1 + C2 n = Cost of taking a sample of size n.

C3 = Cost of finding an assignable cause.

C3'= Cost of investigating a false alarm.

C4 = Difference between the net income per hour of operation in the in-control state and the net income per hour of operation in the out-of-control state.

$1-\beta$ = The power of the control chart, i.e., the probability that an action signal will be generated on a particular sample when the process is really out of control.

g = (Time required to take and interpret a sample of size n)/n.

D = Time required to find an assignable cause following an action signal.

$\alpha$ = Probability that an action signal will be generated on a particular sample when the process is really in control.

Some simplification of this cost function is possible. Duncan notes that the following two approximations can be made

$$\tau = h(\ 1 - \theta h/6)/2,$$

and

$$a \exp(-\theta h)/[1-\exp(-\theta h)] = a/(\theta h)$$

and, consequently, the expected loss per hour can be approximated by

$$E(L) = (C1+C2\ n)/h + [\theta\ B\ C4 + a\ C3'/h + \theta\ C3]/(1+\theta\ B)$$

where

$$B = [1/(1-\beta)-1/2+\theta h/12]h + gn + D.$$

Many researchers have presented minimization algorithms for Duncan's model. For example, A. L. Goel, S. C. Jain, and S. M. Wu (1968) devised an iterative method for optimizing the expected hourly loss that produces the exact optimum solution. Their algorithm consists of solving an implicit equation in the design variables the sample size, n, and the control limit factor, k, and an explicit equation for h, the sampling interval. The use of this procedure not only provides the exact optimum solution but also gives valuable information so that the sensitivity of the optimum cost can be evaluated. In their paper, Goel, Jain and Wu also discuss the nature of the cost surface and the effect of the design variables by using cost contours. In addition, they evaluate the effect of two parameters, the delay factor (the rate at which the time between the taking of the sample and the plotting of a point on the X-bar chart increases with the sample size) and the average time for an assignable cause to occur ($1/\theta$), on the optimum design. A comparison of the results found by Goel, Jain, and Wu with those found by Duncan shows that their algorithm yields designs with smaller cost and in many cases the difference is quite significant. Their procedure is superior to Duncan's approximate

optimization technique specially in situations where either C4 or g are large, or where $\delta$ is small.

W. K. Chiu and G. B. Wetherill (1974) proposed a very simple semi-economic scheme for the design of a control plan using an X-bar chart, that can be applied at the workshop level. Their approximate procedure for optimizing Duncan's model utilizes a constraint on the power of the test; the quality control engineer selects an adequate value for the power, $1-\beta$, to acquire a desired protection against inferior quality (the recommended value is either $1-\beta = 0.90$ or $1-\beta = 0.95$). Then, he determines the values of the control limits coefficients and the sample size from a table provided by the authors and the value of h, the sample interval, is calculated by a simple formula. This procedure usually produces a design close to the true optimum; in fact, despite its simplification of the problem, this method produces, in most of the cases, better solutions than Duncan's more elaborated procedure with the added advantage that the power can now be partly controlled by the engineer.

The expected hourly cost function, E(L), can also be minimized by using an unconstrained optimization or search technique coupled with a digital computer program for repeated evaluations of the cost function. This is the methodology that most recent researchers have taken. Pattern search and various modifications of Fibonacci search have been used effectively. For example, D. C. Montgomery (1982) presented a computer program for the optimization of Duncan's approximated expected loss function. Given fixed and variable sampling costs, the costs of

investigating action signals, the penalty cost of production in the out-of-control state, and other parameters describing process performance, the program finds the sample size, control limit width and interval between samples that minimizes the expected total costs per unit time.

Most of the cost models assume that given that the production process remains in control at a certain point in time, the probability of its deterioration by some future time is independent of the past history of the process. K. R. Baker (1971) suggests that this assumption is one of mathematical convenience only and that the robustness of the Poisson model is open to some question, and he proposes two process models that allow this assumption to be investigated. His models consider a discrete time process in which the output quality characteristic of interest is continuous. The process starts in control, with a quality-characteristic mean at a level $\mu$ and its standard deviation, $\sigma$; the occurrence of an assignable cause results in shift in the process mean to an out-of-control level denoted by $\mu+\delta\sigma$. At some later time, the monitoring process detects this shift and action is taken to restore the process to its in-control state; the cycle then repeats.

In his first model, Baker shows that the long-run average time cost is

$$ATC1 = C1\ n + \{C2\ (1+a\ E[T]) + C3/a'\}/\{E[T] + 1/a'\}$$

where

C1 n = Cost of taking a sample of size n.

C2 = Cost of shutting down the process and searching for an assignable cause.

C3 = Cost of operating out of control for one period.

$a$ = Probability of getting an out-of-control signal when the process is in control.

$a'$ = Probability of getting an out-of-control signal when the process is out of control.

E[T] = Expected value of number of periods that the process remains in control, T; T is assumed to be a random variable with a discrete probability distribution $p(t) = Pr\{T=t\}$, $t=0,1,\ldots$

In his second model, Baker assumes that the time in control is not independent of the number of false alarms that occur and shows that the relevant cost function is

$$ATC2 = C1\ n + (\ C2 + C3\ E[S']\ )/E[D]$$

where C1, C2, and C3 are the same costs defined above, and

E[S'] = Expected value of the run length out of control prior to a particular signal for action.

E[D] = Expected value of the length of a cycle, i.e., the number of periods following the conclusion of a search until the next signal for action.

Baker points out that if the distribution of the duration of the process in control is geometric, then the cost function ATC2 reduces to ATC1. He also investigates in some detail the case where the discrete probability function used to model the process failure mechanism is Poisson and compares it to the usual geometric process; in the Poisson

case, smaller sample sizes and narrower control limits result than those that would be economically optimal in the geometric case; the narrower limits arise because false alarms can be beneficial; this is due to the fact that a false alarm can delay a true shift because the in-control run length does not have the memoryless property of the geometric distribution. He concludes that substantial cost penalties may be incurred if an incorrect process failure mechanism is assumed and that, consequently, it is essential to examine and understand the physical behavior of the deterioration process so that the principle of economic design can be usefully and validly implemented.

I. S. Gibra (1971) proposed a single assignable cause economical model for the determination of the parameters (the sample size, n, the control limits, $k\sigma$, and the intersample interval, h) of an X-bar chart. His model is similar to Duncan's model (1956) in that the process is assumed to be in operation during the search for an assignable cause and in that the time between assignable causes follows an exponential distribution with parameter $\theta$. However, Gibra's model assumes that the sum of times required to take and inspect a sample, compute and plot a sample average, and to discover and eliminate the assignable cause, has an Erlang distribution with parameters $\lambda$ and r. In the development of the model, Gibra introduces the concept of worst cycle quality level (WCQL). He defines a quality cycle as the interval between two successive periods of statistical stability and the worst cycle quality level as the permissible mean expected number of nonconformings produced within a quality cycle. The value of the WCQL

determines an upper bound for the mean expected number of nonconforming units produced during some known production periods. His objective is to determine the optimal parameters of the X-bar chart so as to minimize the cost function associated with the statistical phase of production, subject to the restriction: if the mean shifts by $\pm \delta\sigma$, this shift will be detected and eliminated within a prescribed time interval, say, R, with a specified probability.

Gibra formulates a cost function that includes the cost of inspection and charting, a cost incurred for detecting and eliminating the assignable cause, and a penalty cost due to nonconforming units. The expected total cost per unit time is then

$$E(C) = \frac{C4\ n + C5}{h} + C2\ \Gamma\ \theta + \frac{C1\ a\ \Gamma\ \theta}{\exp(\theta h) - 1} + C3(1-\Gamma)(Wo - W1)$$

where

$a$ = Probability of a false alarm.

$\Gamma = 1/[\ \theta h/p + \theta h/\{\exp(\theta h)-1\} + r\theta/\lambda\ ]$ = Long run probability that process is in state of control.

C1 = Cost for looking for an assignable cause when false alarm is signalled.

C2 = Cost of detecting and eliminating an assignable cause.

C3 = Penalty cost incurred per nonconforming item.

C4 = Cost/unit of inspection and sampling.

C5 = Overhead cost/inspected sample for maintaining the X-bar chart.

h = Sample interval.

n = Sample size.

p = Probability that an assignable cause is detected when the process mean sustains a shift of $\delta\sigma$.

Wo = Expected number of nondefective items produced per unit of time when the process is in a state of control.

W1 = Expected number of nondefective items produced per unit of time when the process is in a state of out of control.

Gibra (1967) also investigated the optimal economical design of an X-bar chart used to monitor a production process in which the mean drifts constantly over time; this kind of situations occur in tool wear in machining, drawing stamping, and molding operations. The relevant cost function in this case is

$$E(C) = [Cr + C3 \text{ a } i]/[W1 \text{ a } \Gamma + W2 \text{ a } (1-\Gamma)] - C3$$

where

E(C) = Total expected cost per unit.

Cr = Cost cf resetting the process mean to its original value.

C3 = Penalty cost incurred per nonconforming item.

a = Length of production run in units of time.

i = Production rate in pieces per unit time.

W1 = Average proportion of nondefective items produced per unit time.

$\Gamma$ = Average proportion of time that the process is in state of statistical stability.

W2 = Average proportion of nondefective items per unit time due to the combined effect of the drift and the shift.

Then, the optimal control procedure determines decision rules for shutting the process down for adjustment due to drift, as well as for the occurrence of an assignable cause. The control rules minimizes adjustment costs and costs due to the production of nonconforming items.

A. J. Duncan (1971) extended his single assignable cause model for the X-bar chart to allow for the occurrence of s independent assignable causes. His models assumes that the process is either in control or it has been disturbed by the occurrence of the jth assignable cause which produces a shift in the process mean of $\delta(j)\sigma$ where $\sigma$ is the standard deviation (assumed to be fixed and known) of X and $\delta(j)$ is a positive constant. When the process is in control, the occurrence times of the various assignable causes are assumed to be independent exponential random variables with parameters $\theta(j)$, $j = 1, 2, 3, \ldots, s$, and when the process has been disturbed by a given assignable cause, the models assumes that the process is free from the occurrence of other assignable causes. Under these assumptions, Duncan shows that the expected loss-cost per unit time is:

$$E(L) = \frac{\sum_{j=1}^{s} \theta(j)B(j)M(j) + \theta\, A\, T + \sum_{j=1}^{s} \theta(j)W(j)}{1 + \sum_{j=1}^{s} \theta(j)B(j)} + \frac{b + cn}{h}$$

where

$B(j) = h/P(j) - \tau(j) + gn + D(j)$ = Average total time between the occurrence of the jth assignable cause and its discovery.

h = Time between samples.

P(j) = Probability that a point falls outside the control limits after the occurrence of the jth assignable cause.

$\tau(j) = \{1-[1+\theta(j)h]\exp(-\theta(j)h)\}/\{\theta(j)[1-\exp(-\theta(j)h)]\}$ = Average time of occurrence of assignable cause j within the sample interval, given that the jth cause occurs between two samples.

g = (Time required to take and interpret a sample of size n)/n.

D(j) = Average time taken to discover the jth assignable cause; it is assumed that the process is kept running at least until the assignable cause is found.

M(j) = Increases loss per unit time of operation due to the presence the jth assignable cause.

A = Average number of false alarms before the occurrence of an assignable cause.

$\theta = \sum\theta(j)$; $1/\theta$ = Expected time at which the process goes out of control.

T = Cost per occasion of looking for an assignable cause when none exists.

W(j) = Average cost of finding the jth assignable cause when it occurs.

b = Cost per sample of sampling, testing, and plotting that is independent of the sample size.

c = Variable cost per item of sampling, testing, and plotting.

Duncan uses direct search methods to find a local minimum of this cost function; initially he treats the sample size, n, as a continuous variable, but the fractional n yielded by the search procedure is rounded both up and down, and then the search procedure is applied again to find the minimizing values for h, the sampling interval, and k, the control limit factor, for each of these two n's; the lower of these two minima is then selected as the final local minimum. The study of this model reveals the existence of readily acceptable (local minimum) solutions that are relatively stable with respect to model changes, including marked changes in the distribution of assignable causes; in some cases, there are also found economically better solutions that would not be as readily acceptable as those offered by the local minima (e.g., the control limits might fall at $\pm 6\sigma$). Duncan argues that as extensions of the model approach reality, only local-minimum solutions will remain and shows that these solutions can be well approximated by solutions of single-cause models and, consequently, in practice it may be sufficient to use single-cause models. W. K. Chiu (1973) pointed out that some numerical results in Duncan's paper are in error; the inaccuracies are due to two sources: an error in Duncan's computer program for the calculation of the $r(j)$'s and the use of single precision arithmetic. Even though the qualitative conclusions of Duncan's paper (1971) are based on numerical study, they appear to remain valid when these inaccuracies are removed.

The assumption that once the process shifts into an out-of-control state no further quality deterioration is possible is often unrealistic. In the same paper, Duncan (1971) also presents a "double occurrence" model. In this formulation it is assumed that following an initial shift a second occurrence of an assignable cause is possible; to simplify the analysis it is assumed that the joint effect of the two assignable causes produces always a shift of $\Delta\sigma$ in the process mean regardless of what two assignable causes occur jointly. This modification in the process model has little effect on the minimum cost solution, although it does produce some changes in the behavior of the cost surface.

H. A. Knappenberger and A. H. E. Grandage (1969) also proposed a model for the economic design of a control chart when there is a multitude of assignable causes. They assume that the process mean, $\mu$, is a continuous random variable that can be approximated by a discrete random variable; one value of the discrete random variable, $\mu(o)$, is associated with the in control state of the process and the remaining values, $\mu(1)$, $\mu(2)$, ..., $\mu(s)$, are associated with out-of-control values of the process mean. They also assume that the time that the process remains in control is exponentially distributed. Knappenberger and Grandage minimize the expected cost per unit produced rather that the expected cost per time as Duncan did later in 1971. Another major difference in their modeling approach in comparison to Duncan's is that there is no constraint on the number of assignable causes that can

occur; that is, the process can shift from one out-of-control state to another, as long as the shift results in further deterioration. Their model is based on the following additional assumptions:

a. The production process is stopped while action signals are investigated.

b. The cost of investigating both real and false alarms is the same.

c. The delay period (the time of taking a sample, inspecting it, and charting the result) is equal to zero. Consequently, the model neglects the expected cost of nonconforming items produced during the delay period.

d. When the process goes out of control it stays out of control until the assignable cause is detected.

e. When the process goes out of control it will not improve. This means that the process mean can only shift to worse values.

f. Only one shift is allowed during a sampling interval.

The expected total cost per unit produced is calculated as the sum of three components:

a. The expected cost of investigating and correcting (if necessary) the process when the control procedure indicates that the process is out of control.

b. The expected cost associated with the production of nonconforming items.

c. The expected cost of sampling and testing.

The combination of these three cost gives an expected total cost:

$$E(C) = \frac{C1 + C2\ n}{m} + \frac{C3}{m}\sum_{i=1}^{s} q(i)\alpha(i) + C4\sum_{i=1}^{s} f(i)\pi(i)$$

where,

C1 + C2 n = Cost of taking a sample of size n.

C3 = Expected cost of investigating and correcting a process that is apparently out of control.

C4 = Cost associated with producing a nonconforming unit of product.

m = Expected number of units produced between samples.

q(i) = Conditional probability of an out-of-control signal when the process given that $\mu = \mu(i)$, i=1,2,...,s.

$\alpha(i)$ = Probability that $\mu = \mu(i)$ at the time the test is performed.

f(i) = Conditional probability of producing a nonconforming item given $\mu = \mu(i)$, i=1,2, ..., s.

$\pi(i)$ = Probability that the process is in state i, i=1,2,...,s.

Knappenberger and Grandage do not derive an optimal solution analytically. Instead, a two stage procedure is developed for choosing the optimal parameters of the chart. In the first stage, the expected cost function is computed for a wide variety of the parameters of the X-bar chart, for cost coefficients, and for the desired values of the a priori distribution parameters. From these results, preliminary estimates of the optimal values of the X-bar chart parameters are obtained. In the second stage, these preliminary estimates are used as the starting point for a search method designed to locate the optimal

values of the control chart parameters within any desired accuracy. The solutions to 81 numerical examples are presented and a limited sensitivity analysis is conducted. By appropriate definition of the cost of investigating action signals, this model produces results consistent with Duncan's multiple-cause model.

Duncan's multiple-cause model seems to have a more realistic cost structure than the Knappenberger-Grandage model, in that the different costs associated with searching for different assignable causes are explicitly treated in the model. However, the Knappenberger-Grandage model allows continued deterioration of quality beyond the initial shift, which may be a more realistic representation of the actual behavior of production processes than the single or double-shift multiple-cause Duncan model. Furthermore, the Knappenberger and Grandage model has fewer parameters to estimate than Duncan's and, consequently, is more appealing to practitioners.

T. J. Lorenzen and L. C. Vance (1986), in an attempt to unify the methodology of control chart design, presented a general method for determining the economic design of a control chart regardless of the statistic used. Their model assumes that when the process goes out of control, it shifts to a known state and cannot return to an in-control state without intervention and that the in-control time is distributed as an exponential random variable with mean $1/\lambda$. In order to develop a model for and to minimize the expected cost per unit time, they define a quality cycle as the time between the start of successive in-

control periods; this entire cycle is represented in Figure 1. Then, the expected cost per unit time can be computed as the ratio of expected cost per cycle to the expected cycle time.

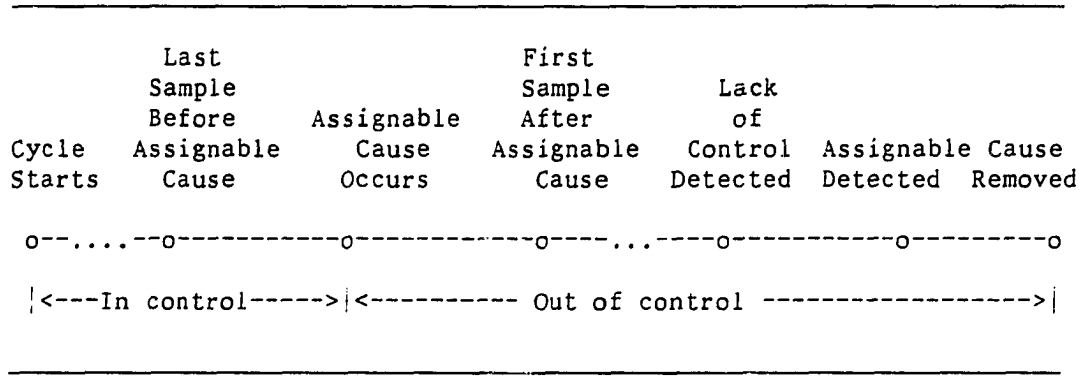



Figure 1. Diagram of in-control and out-of-control states of a process

The authors show that the relevant cost function is:

$$C = \frac{C0/\lambda + C1[-\tau + nE + h(ARL2) + \delta 1\ T1 + \delta 2\ T2] + sY/ARL1 + W}{1/\lambda + (1-\delta 1)sT0/ARL1 - \tau + nE + h(ARL2) + T1 + T2} + \frac{(a+bn)\ [\ 1/\lambda - \tau + nE + h(ARL2) + \delta 1\ T1 + \delta 2\ T2\ ]}{h\ \{\ 1/\lambda + (1-\delta 1)sT0/ARL1 - \tau + nE + h(ARL2) + T1 + T2\ \}}$$

where

$n$ = Sample size.

$h$ = Time between samples.

$\tau = [1-(1+\lambda h)\exp(-\lambda h)]/[\lambda(1-\exp(-\lambda h))]$ = Expected time of occurrence of an assignable cause, given that it occurs between two successive samples ($0 < \tau < h$).

$s = \exp(-\lambda h)/[1-\exp(-\lambda h)]$ = expected number of samples taken while the process is in control.

ARL1 = In-control average run length.

ARL2 = Average run length while the process is out of control because of a slip of size $\Delta\sigma$ in the control parameter.

E = Expected time to sample and chart one item.

T0 = Expected search time when a false signal is given.

T1 = Expected time to discover an assignable cause.

T2 = Expected time to repair the process.

$\delta 1$ = 1 if production continues during searches,

= 0 if production stops during searches.

$\delta 2$ = 1 if production continues during repair,

= 0 if production stops during repair.

C0 = Quality cost per unit time while producing in control.

C1 = Quality cost per unit time while producing out of control (C1 > C0).

Y = Cost of investigating a false alarm.

W = Cost of investigating, locating, and repairing an assignable cause.

a = Cost per sample of sampling, testing, and plotting that is independent of the sample size.

b = Variable cost per item of sampling, testing, and plotting.

Lorenzen and Vance show that if the sample size and the control limits are given (and, consequently, ARL1 and ARL2 are fixed known quantities), the optimum sample interval, h, can be found by solving a quadratic equation in h and using the positive root as an initial point for Newton's method to solve the first order condition for a minimum of

the expected cost. They suggest a minimization technique for the case in which n, $\Delta$, and h are unknown (the control chart is assumed to have control limits of the form $\pm\Delta\sigma$); the algorithm is fast in that it minimizes the cost function in a few seconds on current personal computers. The authors also perform a sensitivity analysis to quantify the effect of changing the sampling frequency to a more natural interval and to quantify uncertainties in process specifications; they find that the minimal cost can be sensitive to uncertainties in process specifications, but the sampling plan will be nearly optimal from a cost standpoint.

### C. Control Charts with Warning Lines

In most economic studies of control charts, only those with action limits are considered despite the fact that, in practice, X-bar charts are seldom used without warning limits or other modifications because it is generally thought that charts with warning limits are more efficient than charts with only action limits.

One modification which has been widely used is a run test on sample means; a run test calls for corrective action when a certain number of points out of a predetermined number of observations fall outside a specified warning line. More precisely, warning lines are drawn in less extreme positions than action lines; the occurrence of a number of points between the warning and the action lines should be considered as sufficient evidence for taking corrective action. Different rules have

been suggested for such schemes. The work of G. H. Weiler (1953) showed that the sequential use of runs in X-bar charts leads to significant reduction in inspection costs, and that in many cases it will be advantageous to introduce it instead of the conventional control chart with only action lines. The conventional chart is to be preferred only when it is possible to take large samples and sequential procedures are not desirable.

P. G. Moore (1958) followed Weiler's suggestion of stopping production when a specified number, r, of means in succession fall over the control limit set up for the scheme and computed the average run length for some particular schemes; see Table 1.

Table 1. Average run length for schemes suggested by Moore (with a single control limit at $\mu+k\sigma$)

| In-control average run length | | | | | | | |
|---|---|---|---|---|---|---|---|
| ARLo = 1000 | | | | ARLo = 200 | | | |
| k = 3.090 | 1.850 | 1.261 | 0.888 | 2.576 | 1.452 | 0.906 | 0.556 |
| r = 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |

He also computed, for three different shifts in the process mean, the probabilities that after a specified number of samples, m, have been drawn there will have been at least one stoppage due to r successive means falling beyond the control limits prescribed and showed that although there is always some gain in using a higher value of r, for a fixed in-control average run length, the actual gain obtained when increasing r by one decreases as r gets larger; his results suggest that there is practically no gain in probability for values of r

greater than 3 and that there is a significant advantage of using r = 2 over using r = 1. Moore recommends the use of this kind of schemes for situations in which the value of the population mean of items being manufactured may change slightly due to tool wear, or a fresh batch of raw material, or a slight variation in the voltage of the power supply and so forth.

One of the earliest works in the calculation of A.R.L. for inspection schemes using control charts with warning limits is due to E. S. Page (1955). In his paper Page considers some schemes based on the observations from the last few samples, provides tables for their average run lengths, and suggests a method for controlling both the mean and the standard deviation of a population on a single chart. The control rules that he considers in detail for controlling the mean of a normal population and their A.R.L.s are:

R1: Take corrective action if (i) two points out of any sequence of n fall between the warning lines or (ii) any point falls outside the action lines.

$$ARL1 = \frac{1 - Po + P1 - P1\,Po^{n-1}}{(1 - Po)(1 - Po - P1\,Po^{n-1})}$$

where Po, P1, and P2 are the probabilities that a given point falls between the warning lines, between the warning lines and the action lines, and outside the action lines respectively.

R2: Take corrective action if (i) n consecutive points fall between the warning and the action lines or (ii) any point falls outside the action lines.

$$ARL2 = (1 - P1^{n})/(1 - Po - P1 + Po\ P1^{n})$$

R3: Plot the means on a chart on which are drawn two warning and two action lines. Take corrective action if (i) any point falls outside the action lines, or (ii) n consecutive points fall outside the warning lines, or (iii) two out of any set of n consecutive points fall outside opposite warning lines.

$$ARL3 = \frac{(1 - RS - R^{n} - S^{n} + SR^{n} + RS^{n})}{P2 + RS(1+Po) + Po(R^{n} + S^{n} - SR^{n} - RS^{n})}$$

where R (S) is the probability that a sample point falls between the upper (lower) warning and action lines.

J. I. Weindling, S. B. Littauer, and J. Tiago de Oliveira (1970) proposed a control chart with warning limits in order to increase the sensibility to small shifts in process mean. The authors established a pair of warning limits, located inside the action limits, and assumed that the occurrence of any of the following events would constitute an action signal: (i) a single sample mean falls outside the action limits, (ii) r consecutive sample averages fall between the upper action and warning limits, or (iii) r consecutive sample averages fall between the lower warning and action limits. Using Markov theory, they derive the expression for the A.R.L.,

$$ARL = \frac{1}{Po + P1^{r}(1-P1)/(1-P1^{r}) + P2^{r}(1-P2)/(1-P2^{r})}$$

where,

Po = Probability of a sample average falling outside action limits.

P1 = Probability of a sample average falling between the upper action and warning limits.

P2 = Probability of a sample average falling between the lower action and warning limits.

They discuss the effects of action and warning limits on the average run length. Their modified chart detects small shifts by means of the occurrence of critical run accumulations in the warning regions and large shifts by means of single samples outside the action limits. They also compare the modified control chart with the traditional chart (only with action lines) and find their chart to be more sensitive for small and moderate-sized shifts in the process mean.

G. R. Gordon and J. I. Weindling (1975) have considered the economic design of warning limit control chart schemes; they consider a production process monitored by a general control chart that calls for action if any of the following three events occur:

1. The last sample mean lies outside the action limits.
2. The last r samples fall between the upper action and warning limits.
3. The last r samples fall between the lower action and warning limits.

A single assignable cause is assumed to randomly occur which shifts the distribution of the attribute values of parts produced to known values; it is assumed that the probability of the shift occurring during an

interval of time t is equal to $1-\exp(-\lambda t)$, where $\lambda$ is a known positive parameter. Since it is also assumed that parts are produced with a constant rate, R, a linear relationship exists between the number of parts, $\nu$, and the time required to produce them, t, by letting $\theta = \lambda/R$ be the mean number of assignable causes during the time required by the production process to produce one part, the probability of the occurrence of an assignable cause during the production of $\nu$ parts is equal to $1-\exp(-\theta\nu)$. The authors consider three types of costs in their model: the cost of sampling (which includes the cost of selecting the parts to be sampled, the cost of making the measurements, the cost of interpretation, and the cost of maintaining the control chart), cost of searching for an assignable cause and cost of process adjustment if found, and cost of defectives (which includes costs of reworking and/or scrapping of the final product, costs of returns, costs of failures in service, replacement, and loss of good will).

In order to obtain an expression for the average cost per good part produced, Gordon and Weindling consider the process as being composed of two distinct types of cycles. The type 1 cycle is defined as the occurrence of a spurious action signal before the occurrence of an assignable cause; cycle type 2 is defined as the arrival of an assignable cause before a spurious action signal occurs and then, after some additional samples, the occurrence of an action signal. The average cost per good part produced, A, can then be expressed as:

$$A = \frac{P1\ E(C|1) + P2\ E(C|2)}{P1\ E(G|1) + P2\ E(G|2)}$$

where,

Pi = Probability of cycle i occurring; i = 1,2.

E(C|i) = Expected cost per type i cycle occurs; i=1,2.

E(G|i) = Expected number of good parts produced during given that a type i cycle occurs; i = 1,2.

The authors present general expressions for the variables E(C|1), E(C|2), E(G|1), and E(G|2) in terms of the average number of samples until an action signal is obtained given that a shift does not occur first, the average number of samples taken before a shift occurs, the mean number of samples after a shift occurs until an action signal occurs, the fraction of the sampling interval before the occurrence of a shift, the fraction defective while the process is in control, the fraction of defective parts produced by the process when it is out of control, and the sample size. They use Markov modeling to determine the probabilities P1 and P2 and the expected values that comprise the average cost expression. They also present a numerical example which indicates the variation in the average cost expression for various control plans and they discuss some optimization considerations for the average cost expression.

W. K. Chiu and K. C. Cheung (1977) investigated the economic design of X-bar charts with both warning and action limits, based on a process model similar to the Gordon-Weindling model. They use a three-dimensional pattern search to optimize the cost function and they make comparisons among the minimum cost designs of X-bar charts with and

without warning limits and of Cusum charts and they find no appreciable differences. Thus, for practical application, they recommend the use of X-bar charts with warning limits as they are much easier to handle than Cusum charts, and they provide more psychological protection than X-bar charts with action limits only. They also suggest a simplified economic scheme for the design of control parameter values and propose that in practice the band enclosed by the warning limits should be 0.85 times as wide as the band enclosed by the action limits rather than 2/3 as has been commonly practiced.

D. J. Wheeler (1983) provides tables of the power function of an X-bar chart for the set of detection rules that appear in the Statistical Quality Control Handbook published by Western Electric. His first table contains the probabilities of obtaining an action signal when the process is under control (C. W. Champ (1986) points out that these probabilities are incorrect). His second table covers eleven sizes of shifts in the process average, and gives the probabilities that the X- bar chart will indicate a lack of control in the direction of the shift within a given number of samples. He also presents tables for the proportion of product outside the specifications in terms of the shifts in the process mean. The method used to construct these tables is a systematic enumeration of all possible configurations of the control chart.

C. W. Champ (1986) considers the detection rules recommended by the Western Electric's Statistical Quality Control Handbook. He outlines a general methodology for computing the distribution of the run

length and the average run length based on Markov theory and provides a method for obtaining the minimum number of states for the Markov chain representation of the chart. He also computes the average run length of an X-bar chart as a function of the process mean for fourteen different detection rules. Even though most of his numerical results are for X-bar charts, his methodology is not limited to this chart; his method can be applied to other control charts such as the R chart, np chart and c chart. He also finds that X-bar charts with supplementary runs rules were more sensitive to small to moderate changes in the mean that the corresponding X-bar chart, but not as sensitive as the corresponding Cusum chart for small, medium, and moderately large shifts in the process mean. In addition, Champ studies and presents some numerical results for multidimensional Shewhart control charts with supplementary runs rules.

### D. Simplified Designs for Control Charts

Economic models are relatively complex and may not be useful for practical applications because in practice the values of most costs and risk parameters are rarely available and in many occasions cannot be estimated precisely. Thus, for basic charts such as the X-bar, the u chart and the p chart, several rules of thumb for the selection of the sample size, n, the sampling period (hours between samples), h, and the control limits, k (the number of standard deviations above or below the center line), have been developed and applied.

Although the X-bar chart has probably been the most studied control chart, some significant attention has been devoted also to the design of the p chart, used to control a Bernoulli process, and to the u chart, used to control the number of defects per unit. As in the X-bar chart, the design of these charts requires the specification of the three parameters n, h and k. Table 2 summarizes the most widely used simplified designs.

Table 2. Simple designs for control charts

| Chart | Proposed by: | Year | Sample size | Sampling period | Control Limits |
|---|---|---|---|---|---|
| X-bar | Ishikawa | 1976 | 5 | 1 | $3\sigma$ |
| X-bar | Juran et al. | 1974 | 4 | Not given | $3\sigma$ |
| X-bar | Feigenbaum | 1961 | 5 | 1 | $3\sigma$ |
| X-bar | Burr | 1953 | 4 or 5 | Not given | $3\sigma$ |
| p | Grant & Leavenworth | 1980 | 100% inspec. | 8 | $3\sigma$ |
| p | Ishikawa | 1976 | >50, 3<np<4 | 8 | $3\sigma$ |
| p | Juran et al. | 1974 | >50,np>4 | Not given | $3\sigma$ |
| p | Juran et al. | 1971 | 9(1-p)/p | Not given | $3\sigma$ |
| p | Feigenbaum | 1961 | 25 | 1 or 8 | $3\sigma$ |
| p | Cowden | 1957 | np>25 | Not given | $3\sigma$ |
| p | Burr | 1953 | np> 1 | Not given | $3\sigma$ |
| u | Ishikawa | 1976 | 2 or 3 | 8 | $3\sigma$ |
| u | Juran et al. | 1974 | 1 | Not given | $3\sigma$ |

# III. PROBLEM STATEMENT, RESEARCH OBJECTIVES AND METHODOLOGY

## A. Problem Statement

Current control schemes fall basically into two categories: either they are complicated and complex, requiring the simultaneous application of multiple stopping rules, or they are simplified designs relatively insensitive to moderate shifts in the process mean. Complex schemes are cumbersome to apply in practice because the person in charge of taking the samples and plotting them is asked to check for the violation of a multitude of stopping rules, some of these requiring keeping track of up to ten sample points in a control chart with up to seven warning lines; this process is time consuming and prone to errors. On the other hand, simplified control schemes, even though they can be easily implemented, usually are far from optimal from an economical view point. The need for simple control schemes with in-control average run lengths comparable to those of more complex control schemes and short out-of-control average run lengths is evident.

In this research, we are not concerned with comparing the effects of different ways of collecting the data, for example whether it is preferable to take small samples frequently or larger samples more rarely. Although comparisons like these are important in practice (when they can be made), the quality control engineer has gotten used to small samples (n=5) for controlling or estimating the quality of the output of a production process. There are several valid reasons for this

choice; for example, in the case in which the parameter being controlled is the mean, a small sample will easily detect a very serious shift; in addition, in most cases, the fraction of production that may be inspected is determined by a limitation on the man-hours available. Consequently, our concern is to assess control schemes for obtaining a signal for action when the observations are obtained in the same way.

Another reason why we are not considering the effects of the sampling interval and sample size in a given control scheme is because the economic models for control charts have provided a good qualitative insight on the values of these parameters. For example, in an X-bar chart, the sample size is mainly determined by the size of the shift, $\delta$, in the process mean to be detected; for small shifts, $\delta < 0.50\sigma$, the sample size can be as large as 35 or more; moderate shifts, $\sigma < \delta < 2\sigma$, usually require sample size between 10 and 25 and relatively large shifts, $\delta > 2\sigma$, often result in a small sample size (between 3 and 10). The sampling interval is largely determined by the hourly penalty cost for producing items when the process is in the out-of-control state; larger values of this cost imply more frequent sampling while smaller values of this cost imply less frequent sampling. It is also reassuring to know that numerical studies have indicated that the optimal control chart design is relatively insensitive to misspecification of the cost parameters but relatively sensitive to the magnitude of the shift in the process mean.

For a control chart with action limit only and the sample size fixed, only the location of the action lines can be changed. These type of changes affect the level of the under control A.R.L. but they do not change the basic shape of the A.R.L. curve; as a result, this type of control charts is relatively insensitive to small changes in the process mean, unless the sample size is made large, or the control limit is decreased to a point where an excessive number of false signal will result. For control charts with warning and action lines the situation is different since there is an infinite number of action and warning limit combinations that will result in the same under control average run length. Thus, fixing the in-control A.R.L. defines a family of A.R.L. curves and it is possible to select, within this family, the curve that minimizes the A.R.L. at a given out-of-control state as measured by a shift in the process mean. The objective of this research is to identify these curves.

In this research we assume that the in-control state of the process being monitored can be described by a random variable with a known distribution. The control scheme used to control the process consists of a set of rules of the form: declare the process to be out-of-control if k out of m consecutive points fall in the interval (a,b) ($a < b$), or k out of m consecutive points fall in the interval (-b,-a). We are specially interested in investigating the average run length properties of these type of schemes for relatively small values of m, say $m < 5$.

To be more specific, consider a continuous random variable X that measures the quality of a production process; we observe successively

the independent samples averages $\bar{X}(1)$, $\bar{X}(2)$, ..., and assume that the $\bar{X}(i)$'s are independent and normally distributed with mean $\mu$ and with known and fixed standard deviation $\sigma/\sqrt{n}$, where n is the sample size. For convenience and without loss of generality, we assume that the control scheme is based on the standardized sample means,

$$Z(i) = [X(i) - \mu o]/[\sigma/\sqrt{n}], \quad i = 1,2,...,$$

where $\mu o$ is the desired in-control value for expected value of the random variable X.

Denote by R[k,m,a,b] a control rule that calls for action if k out of m consecutive standardized sample averages fall in the interval (-b, -a), or k out of m consecutive standardized sample averages fall in the interval (a,b). Then, a control scheme, S, that calls for an action if any of the rules Rj, j=1, 2, ... r, can be described as $S = \{ Rj[ k(j), m(j), a(j), b(j) ] \mid j = 1,2,...,r\}$. Using this notation the control scheme recommended by the Western Electric Company's Statistical Quality Control Handbook (1956) can be described as

$$\{ R1[1,1,3,\infty], R2[2,3,2,\infty], R3[4,5,1,\infty], R4[8,8,0,\infty] \},$$

and the control scheme recommended by Grant and Leavenworth (1988) can be described as

$$\{ R1[7,7,0,\infty], R2[10,11,0,\infty], R3[12,14,0,\infty], R4[14,17,0,\infty], R5[16,20,0,\infty] \}$$

We are interested in control schemes that produce a large number of samples before receiving an action signal when the process is in state of control (i.e., a large in-control average run length) and that also produce a small number of samples before receiving an action signal

when the process mean has shift from the target value $\mu = \mu o$ to the out-of-control value, say $\mu = \delta$. Since the average run length as a function of the magnitude of the process mean is a strictly decreasing function, the A.R.L. curve has a shape similar to that of the operating characteristic of a two-tailed test. Then, we will say that, given two control schemes, say S1 and S2, S1 is more powerful than S2 if the in-control average run length of S1 is greater than or equal to the in-control average run length of S2 and the out-of-control average run length of S1 is less than the out-of-control average run length of S2. Figure 2 illustrates this situation.

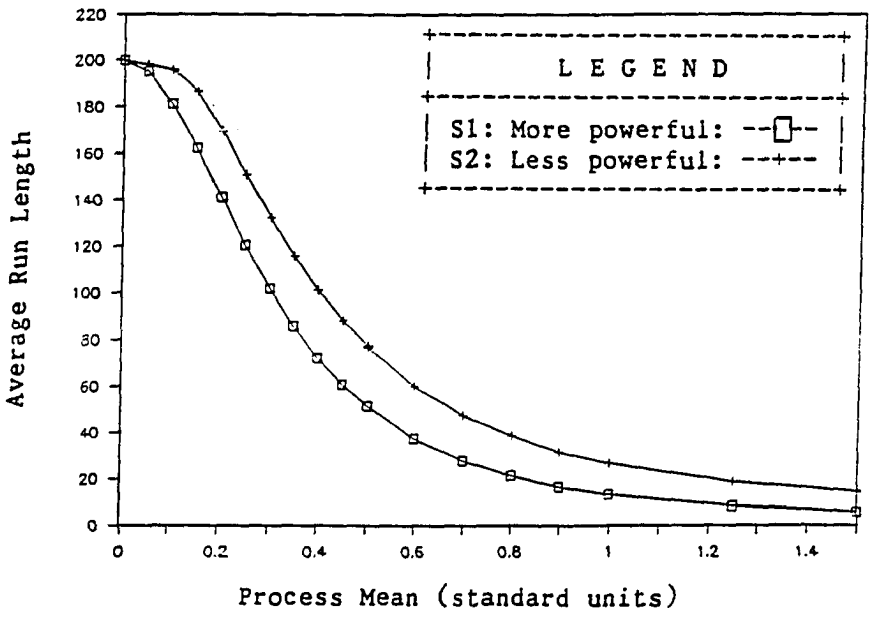



Figure 2. Power comparison of two control schemes

For a given control scheme, S, if we consider only the k(j)'s and the m(j)'s to be known and fixed, the average run length (the expected number of samples taken before receiving an action signal) depends upon

the process mean, $\mu$, and the control limits a(j), b(j), j = 1,2,...,r. In addition, notice that, by letting the control limits vary, we have defined a family of control schemes in which each member is identified by a particular set of values for the control limits. For instance, the control scheme suggested by Wetherill (see Section I.B.4) { R1[1,1,3,$\infty$], R2[2,2,2,$\infty$] } is a member of the family { { R1[1,1,x,$\infty$], R2[2,2,y,$\infty$] } | $x \geq y$ }.

The average run length is a function of the control limits, a(j), b(j), j=1,2,3,...,r and the process mean, $\mu$, that is, ARL = ARL(a,b,$\mu$), where $\underline{a}$ = [a(1), a(2), ..., a(r)] and $\underline{b}$ = [b(1), b(2), ... b(r)]. Then, given a family of control schemes, F, with a fixed in-control average run length, ARLo, we want to identify its most powerful members. This problem can be described in mathematical terms as follows:

Given ARLo, r, $\delta$, and the integer positive numbers m(1), m(2), ..., m(r), k(1), k(2), ..., k(r) that define the family of control schemes

$$F = \{ S \mid S = \{ R1[k(1),m(1),a(1),b(1)], R2[k(2),m(2),a(2),b(2)], \ldots \ldots Rr[k(r),m(r),a(r),b(r)] \} \}$$

find optimal control limits, say, $\underline{a}^* = [a^*(1), a^*(2), \ldots, a^*(r)]$, and $\underline{b}^* = [b^*(1), b^*(2), \ldots, b^*(r)]$ such that, for all $\underline{a}$ and $\underline{b}$ such that ARL($\underline{a}$,$\underline{b}$,0) = ARLo,

$$ARL(\underline{a}^*, \underline{b}^*, \mu) < ARL(\underline{a}, \underline{b}, \mu) \text{ for all } -\delta < \mu < \delta$$

$$\text{and } ARL(\underline{a}^*, \underline{b}^*, 0) = ARLo,$$

## B. A General Methodology for Determining the Average Run Length

A control scheme of the type previously discussed can be represented by a finite Markov chain with a single absorbing state. The absorbing state represents an action signal given by the control chart and the transient states indicate the status of the chart with respect to each control rule. The run length is then the number of transition steps in which the Markov chain is in a transient state; consequently, the average run length can be computed as the mean time to absorption.

When solving a problem concerning the motion of an absorbing Markov chain within the set of its transient states there is no loss of generality if all its recurrent states are assumed to be absorbing; consequently, in such problems, the transition matrix of the Markov chain can be assumed to have the form

$$P = \begin{vmatrix} I & O \\ R & Q \end{vmatrix}$$

where I is an identity matrix whose order is equal to the number of absorbing states in the Markov chain, Q is a square matrix whose order is equal to the number of transient states, O is a null matrix, and R is a matrix containing the transition probabilities from each transient state to each absorbing state.

The inverse of the matrix (I-Q), A, (where I is an identity matrix whose order is equal to the number of transient states), is called the fundamental matrix of the absorbing Markov chain. The moments of the random variable T, the time to absorption (the number of transition

steps in which the Markov chain is in a transient state), can be expressed in terms of the fundamental matrix, A; in particular,

$$\underline{E}(T) = A\,\underline{e}$$

$$\underline{E}(T^2) = (2A - I)\,A\,\underline{e}, \text{ and}$$

$$\underline{E}(T^3) = [\,6A\,(A - I) + I\,]\,A\,\underline{e}$$

where $\underline{e}$ is a column vector with all its components equal to 1 (M. Iosifescu, 1980). The ith component of the vector $\underline{E}(T)$ gives the expected absorption time given that the process starts in the ith transient state.

To be more specific and to illustrate how a control scheme can be represented by a Markov chain, consider the following example. Imagine a situation in which an increase in the mean of a given process is to be detected. Furthermore, assume that the following rule is being used to control such a process:

A positive shift in the process mean has occurred if two out three successive sample means fall above the $+2\sigma$ limit.

Figure 3 illustrates this control scheme. To define the non-absorbing states of the Markov chain representation of the control chart, one must examine what information the control rule requires one to remember each time that a sample is taken. The rule "stop if two out three successive sample means fall above the $+2\sigma$ limit" requires one only to keep track of the last two sample points and their location in the control chart relative to the control limit. If we let "A" represent a point falling in the zone above the $2\sigma$ limit and "B" represent a point falling below the $2\sigma$ limit, at any time, we can

imagine the chart as being in one out of three possible transient states:

BB = The last two sample points are below the $2\sigma$ limit.

AB = The last sample point is below the $2\sigma$ limit and the previous sample point is above the $2\sigma$ limit.

BA = The last sample point is above the $2\sigma$ limit and the previous sample point is below the $2\sigma$ limit.

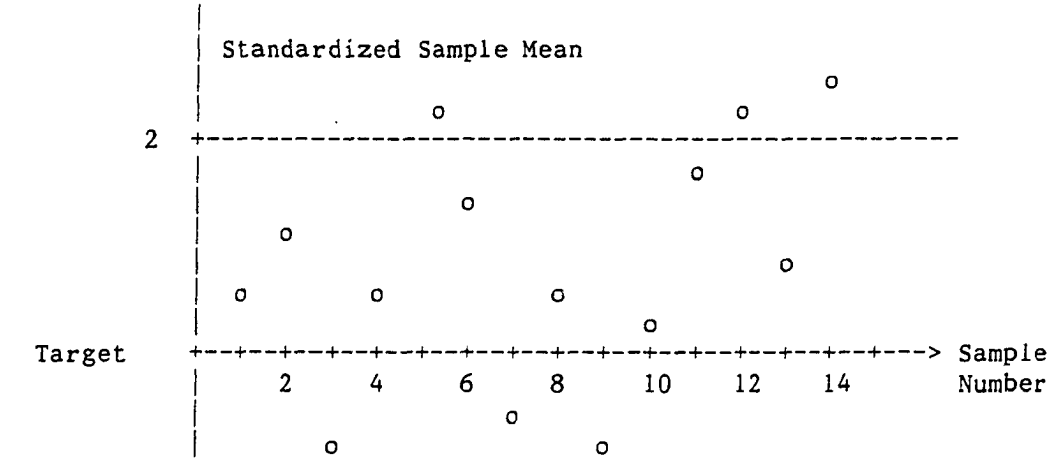


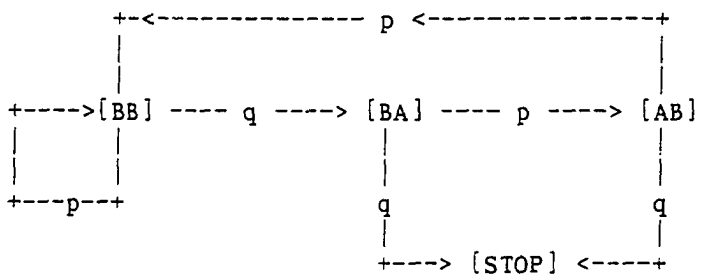

Figure 3. A control chart to detect a positive shift in the process mean

The absorbing state of the Markov chain represents an action signal, that is, the event "two out three successive sample means fall above the $+2\sigma$ limit" (BAA, ABA).

Let $p = \text{Prob}\{\text{Standardized sample mean} < 2\} = \Phi(2-\mu)$, and

$q = \text{Prob}\{\text{Standardized sample mean} > 2\} = 1-p = 1-\Phi(2-\mu)$,

where $\Phi(x)$ is the standard normal cdf, and $\mu$ is the process mean. Then, the Markov chain representation of the control scheme is




and the corresponding transition probability matrix is:

$$P = \begin{array}{cc} & \begin{array}{cccc} (0) & (1) & (2) & (3) \end{array} \\ \begin{array}{c} \text{State} \\ (0)\ \text{STOP} \\ (1)\ \text{BB} \\ (2)\ \text{BA} \\ (3)\ \text{AB} \end{array} & \begin{bmatrix} \text{STOP} & \text{BB} & \text{BA} & \text{AB} \\ 1 & 0 & 0 & 0 \\ 0 & p & q & 0 \\ q & 0 & 0 & p \\ q & p & 0 & 0 \end{bmatrix} \end{array} = \begin{bmatrix} I & O \\ R & Q \end{bmatrix}$$

where $Q = \begin{bmatrix} p & q & 0 \\ 0 & 0 & p \\ p & 0 & 0 \end{bmatrix}$, $I = [\,1\,]$, $O = [\,0 \quad 0 \quad 0\,]$, and $R = \begin{bmatrix} 0 \\ q \\ q \end{bmatrix}$.

The expected times to absorption, $\underline{E}(T)$, can be computed as

$$\underline{E}(T) = (I - Q)^{-1} \underline{e}\ , \text{ where } \underline{e} = [1, 1, 1]'.$$

The vector $\underline{E}(T)$ contains the expected number of transitions starting from a given transient state until absorption. If we assume that we start in state BB, the average run length is equal to the first component of the vector $\underline{E}(T)$, or

$$ARL(\mu) = [\ 2 - p^2\ ] / [\ 1 - p - qp^2\ ]$$

$$= [\ 2 - p^2\ ] / \{\ (1-p)^2\ (1+p)\ \}$$

$$= [\ 2 - \Phi(2-\mu)^2\ ] / \{\ [1-\Phi(2-\mu)]^2\ [1+\Phi(2-\mu)]\ \}$$

This Markov chain representation of a control scheme can be easily extended to control charts in which several control rules are used. The transition probabilities, and, thus, the average run length, depend on the control and warning limits and the process mean. Numerical methods can be used to compute the fundamental matrix of the Markov chain and to find the values of the control and warning limits that give the most powerful control scheme.

### C. Robustness of the Most Powerful Control Schemes

The most powerful control schemes to be found in this research are the most powerful in the sense that if the production process goes out of control because of a shift in the process mean, these schemes will detect such a shift, in average, in the minimum number of samples. In practice, however, a change in the process mean might be accompanied also by a change in the distribution of the random variable being monitored or trends also might appear.

Monte Carlo simulation will be used to evaluate the behavior of the most powerful control schemes under a variety of out-of-control situations (other than a shift in the process mean) and they will be compared to the behavior of a control scheme widely used in practice.

The following out-of-control situations will be studied:

1. The random variable monitored is distributed according to a double exponential distribution with mean $\mu$ and variance $\lambda^2/2$, that is,

$$f(x) = (\lambda/2) \exp[\, -\lambda|x-\mu|\, ], \quad -\infty < x < +\infty$$

2. The random variable monitored is distributed according to a Cauchy distribution with mean $\mu$ and scale parameter $\sigma$,

$$f(x) = 1/\{\pi\sigma[\, 1 + \{(x - \mu)/\sigma\}^2\, ]\}, \quad -\infty < x < +\infty\ .$$

3. The random variable monitored is the result of a mixed auto-regressive-moving average process of first order,

$$x[n] = \epsilon[n] + \beta\ \epsilon[n-1] - \alpha\ x[n-1], \quad n = 1,2,3,...$$

where $\alpha$ and $\beta$ are constants and the $\epsilon[i]$'s, $i = 1,2,3,...$ are independent and normally distributed with mean zero and standard deviation $\sigma$.

# III. DETERMINATION AND EVALUATION OF THE MOST POWERFUL CONTROL SCHEMES

In this chapter, four different families of control schemes are studied and their most powerful members identified. The Markov chain representations of these control schemes are given and these representations are used to compute their average run length properties.

## A. The { R1[1,1,x,∞], R2[2,3,y,∞] } Family of Control Schemes

In this section, the following family of control schemes is studied and optimized: those control schemes that declare the process out of control if

a. a single sample mean falls x or more standard units away from the target mean,

b. two of the last three sample means fall y or more standard units above the target mean, or

c. two of the last three sample means fall y or more standard units below the target mean.

Figure 4 illustrates this type of control scheme.

To define the nonabsorbing states of the Markov chain representation of this control scheme, it is necessary to keep track only of the last two sample points and their location relative to the control limits. If we let "A" represent a point falling in the interval (+y, +x), "O" represent a point falling in the interval (-y, +y), and "B" represent a point falling in the interval (-x, -y), at any

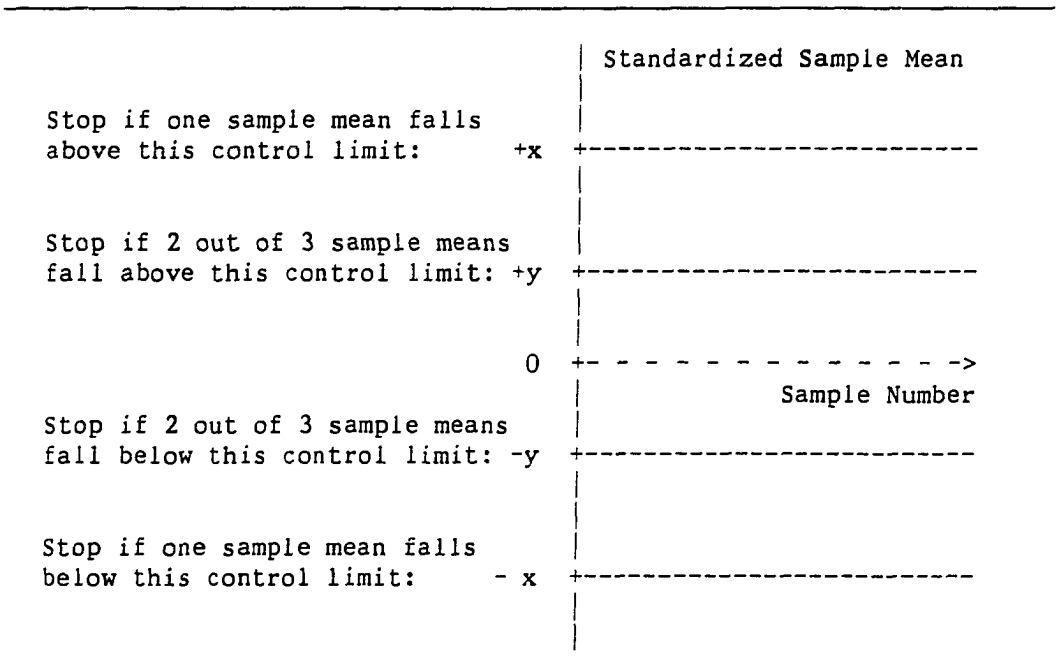



Figure 4. The { R1[1,1,x,∞], R2[2,3,y,∞] } control scheme

time, the control chart can be thought of as being in one out of the eight possible states:

State No. 0: STOP: An out-of-control signal has been obtained.

State No. 1: OO: The last two sample points are in the interval (-y,+y).

State No. 2: OA: The last sample point is in the interval (+y, +x) and the previous sample point is in (-y, +y).

State No. 3: AO: The last sample point is in the interval (-y, +y) and the previous sample point is in (+y, +x).

State No. 4: AB: The last sample point is in the interval (-x, -y) and the previous sample point is in (+y, +x).

State No. 5: BA: The last sample point is in the interval (+y, +x) and the previous sample point is in (-x, -y).

State No. 6: BO: The last sample point is in the interval (-y, +y) and the previous sample point is in (-x, -y).

State No. 7: OB: The last sample point is in the interval (-x, -y) and the previous sample point is in (-y, +y).

Let $\Phi(\ )$ be the standard normal c.d.f and $\mu$ the process mean, then

Po = Prob{Standardized sample mean falls in the interval (-y, +y)}

$= \Phi(y-\mu) - \Phi(-y-\mu)$,

Pa = Prob{Standardized sample mean falls in the interval (+y, +x)}

$= \Phi(x-\mu) - \Phi(y-\mu)$, and

Pb = Prob{Standardized sample mean falls in the interval (-x, -y)}

$= \Phi(-y-\mu) - \Phi(-x-\mu)$,

and the transition probability matrix of the Markov chain representation of the control scheme is

$$P = \left|\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1-Po-Pa-Pb & Po & Pa & 0 & 0 & 0 & 0 & Pb \\ 1-Po-Pb & 0 & 0 & Po & Pb & 0 & 0 & 0 \\ 1-Po-Pb & Po & 0 & 0 & 0 & 0 & 0 & Pb \\ 1-Po & 0 & 0 & 0 & 0 & 0 & Po & 0 \\ 1-Po & 0 & 0 & Po & 0 & 0 & 0 & 0 \\ 1-Po-Pa & Po & Pa & 0 & 0 & 0 & 0 & 0 \\ 1-Po-Pa & 0 & 0 & 0 & 0 & Pa & Po & 0 \end{array}\right| = \left|\begin{array}{c|c} I & O \\ \hline R & Q \end{array}\right|$$

The expected times to absorption, $\underline{ET}$, can be computed as

$$\underline{ET} = \begin{vmatrix} 1-Po & -Pa & 0 & 0 & 0 & 0 & -Pb \\ 0 & 1 & -Po & -Pb & 0 & 0 & 0 \\ -Po & 0 & 1 & 0 & 0 & 0 & -Pb \\ 0 & 0 & 0 & 1 & 0 & -Po & 0 \\ 0 & 0 & -Po & 0 & 1 & 0 & 0 \\ -Po & -Pa & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -Pa & -Po & 1 \end{vmatrix}^{-1} \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix} = \begin{vmatrix} ET(1) \\ ET(2) \\ ET(3) \\ ET(4) \\ ET(5) \\ ET(6) \\ ET(7) \end{vmatrix}$$

The ith component of the vector $\underline{ET}$, ET(i), gives the expected number of transitions starting from state i ( i = 1,2,...,7 ) until absorption. If it is assumed that at time zero the process is in control, that is, in state 1 (00), the average run length (for a given pair of control limits, x and y, and a fixed process mean, $\mu$) is equal to the first component of the vector $\underline{ET}$, or $ARL(x,y;\mu) = ET(1)$.

The problem is to determine the values of the control limits, x and y, such that, for a given in-control average run length, the out-of-control run length is minimized. For the purpose of this discussion, it will be assumed that the process is out of control if a shift of at least one standard deviation has occurred in the process mean. Thus, in mathematical terms the problem is:

Given ARLo and the family of control schemes

$F = \{ S \mid S = \{ R1[1,1,x,\infty], R2[2,3,y,\infty] \}, x \geq y \}$

find optimal control limits, say x* and y*, such that for all x and y such that $ARL(x, y; 0) = ARLo$,

$$ARL(x^*, y^*; 1) \leq ARL(x, y; 1)$$

and $ARL(x^*, y^*; 0) = ARLo$.

This problem is equivalent to the equality-constrained minimization problem,

(P1) Minimize $h(x,y) = ARL(x,y;\ 1)$ subject to

$ARL(x,y;\ 0) = ARLo$, and $0 \leq y \leq x$.

One way to solve this problem is by using penalty methods. Instead of minimizing h(), a new objective function, g() is minimized. The function g() is constructed from the original objective function and the constraints in such way that g() includes a "penalty" term which increases the value of g() whenever a constraint is violated with larger violations resulting in larger increases.

In particular, problem (P1) can be solved by solving

(P2) Minimize $g(x,y) = ARL(x,y;\ 1) + k \mid ARL(x,y;0) - ARLo \mid$

where k is a positive constant. Because of the simplicity of the constraints $0 \leq y \leq x$, they can be handled implicitly in the optimization process and they are not included in the function g(). In addition, notice that the constraint $ARL(x,y;\ 0) = ARLo$ can be rewritten in the form $x = f(ARLo,\ y)$ and problem (P2) is then equivalent to

(P3) Minimize $g(y) = ARL(\ f(ARLo,y),y;1) + k|ARL(f(ARLo,y),y;0)-ARLo|$

Problem (P3) is a nonlinear optimization problem in one variable and can be solved using standard optimization methods.

1. Evaluation of the Modified Objective Function: g(y)

The evaluation of the function

$g(y) = ARL(\ f(ARLo,y),y;1) + k \mid ARL(f(ARLo,y),y;\ 0) - ARLo \mid$

requires the evaluation of the functions ARL(x,y; $\mu$) and f(ARLo,y). At first glance, it seems that the evaluation of ARL(x,y; $\mu$) requires the inversion of the matrix A = (I-Q); however, a closer look to the problem reveals that in order to evaluate ARL(x,y; $\mu$), it is only necessary to find the value of ET[1] in the following system of linear equations:

$$a[1,1]\,ET[1] + a[1,2]\,ET[2] + a[1,3]\,ET[3] + \ldots + a[1,7]\,ET[7] = 1$$
$$a[2,1]\,ET[1] + a[2,2]\,ET[2] + a[2,3]\,ET[3] + \ldots + a[2,7]\,ET[7] = 1$$
$$a[3,1]\,ET[1] + a[3,2]\,ET[2] + a[3,3]\,ET[3] + \ldots + a[3,7]\,ET[7] = 1$$
$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$
$$a[7,1]\,ET[1] + a[7,2]\,ET[2] + a[7,3]\,r[3] + \ldots + a[7,7]\,ET[7] = 1$$

where a[i,j] represents the element in the ith row and the jth column of the matrix A = (I - Q) and ET[i] is the average run length given that the control chart started in state i, i = 1,2,..,7, j = 1,2,...,7. Notice that the values of the a[i,j]'s are known once x, y and $\mu$ are fixed. By performing Gaussian row operations on this system of linear equations, it can be reduced to an equivalent lower triangular system:

$$a'[1,1]\,ET[1] = b[1]$$
$$a'[2,1]\,ET[1] + a'[2,2]\,ET[2] = b[2]$$
$$a'[3,1]\,ET[1] + a'[3,2]\,ET[2] + a'[3,3]\,ET[3] \quad \ldots = b[3]$$
$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$
$$a'[7,1]\,ET[1] + a'[7,2]\,ET[2] + a'[7,3]\,ET[3] + \ldots + a'[7,7]\,ET[7] = b[7]$$

from which the value of ARL(x,y; $\mu$) can be easily computed as b[1]/a'[1,1]. In order to reduce the round-off error, partial pivoting

should be used during the triangularization process (Johnson and Dean Riess, 1982, pp. 27-32).

Given a desired in-control average run length, ARLo, and a value for the inner control limit, y, finding the value of the outer control limit, x = f(ARLo,y), that results in the desired ARL, requires the solution of the nonlinear equation ARL(x,y;0) = ARLo. The use of methods that require derivatives, such as Newton's method, to solve this nonlinear equation is practically out of the question. Consequently, the value of x=f(ARLo,y), will obtained using the secant method (Johnson and Dean Riess, 1982, pp. 166-169). The secant method was chosen to solve ARL(x,y;0) = ARLo because its rate of convergence is stronger than linear, it is easy to program, and it does not require derivative evaluations.

Algorithm GEVAL, given below, describes the steps necessary to evaluate the modified objective function g(y).

Algorithm GEVAL

This algorithm may be used to evaluate the modified objective function g(y).

Given: A procedure to initialize the matrix A = (I-Q).

A procedure to triangularize a matrix.

y = Inner control limit.

ARLo = Desired in-control ARL.

$\mu$ = Out-of-control process mean.

Nmax = Maximum number of iterations for the secant method.

Tol = Tolerance to be used in the secant method.

k = Penalty weight.

1. Compute two initial estimates for the value of x = f(ARLo, y). (Two fairly good initial estimates are x1 = y + |3-y|/4 and x2 = 3.0).
2. Use these estimates of x as starting points for the secant method to solve the equation ARL(x,y;0) = ARLo.
3. Initialize matrix A = (I-Q) using the given inner control limit, y, the computed outer control limit, x, and the out-of-control process mean, $\mu$.
4. Triangularize the matrix A and compute ARL(x,y;$\mu$).
5. If | ARL(x,y;0) - ARLo | < Tol then let g(y) = ARL(x,y;$\mu$); otherwise let g(y) = ARL(x,y;$\mu$) + k | ARL(x,y;0) - ARLo |.

2. Minimization of the Modified Objective Function: g(y)

The minimization of the modified objective function, g(), is carried out using an algorithm that does not require the evaluation of the derivative of that function. Although, in general, algorithms using the derivative are somehow more powerful than those using only the function, the fact that in this case it is not possible to obtain an explicit formula for the derivative of g(y) makes the application of such methods undesirable; the application of such methods to this problem would require the numerical evaluation of the derivative of g() which is a particularly unstable process and quite difficult to analyze carefully. The method used to find the global minimum of the function

g() is based on inverse parabolic interpolation (Press et al., 1986). The basic idea behind this method is, first, to find an interval (y[1], y[3]) on which the objective function is convex and preferably containing the global minimum y*. Then, select another point, y[2], in that interval, find a second degree polynomial interpolating g() at y[1], y[2], and y[3], find the minimum of this interpolating polynomial, and use this minimum as an approximation to the minimum of g(). This procedure is repeated until some criteria for the accuracy of y* is satisfied. The formula for the abscissa, y, which is the minimum of a parabola through three points (y[i], g[i]), i = 1, 2, 3 is

$$y = y[2] - \frac{1}{2}\,\frac{(y[2]-y[1])^2\,(g[2]-g[3]) - (y[2]-y[3])^2\,(g[2]-g[1])}{(y[2]-y[1])\,(g[2]-g[3]) - (y[2]-y[3])\,(g[2]-g[1])}$$

Algorithm GMIN integrates the basic ideas presented previously and provides more details on the procedure used to minimize the modified objective function.

Algorithm GMIN

Given: A procedure to evaluate the function g(y).

ARLo = Desired in-control average run length.

Nmax = Maximum number of iterations.

Yerror = Maximum error allowed in the optimal value for the inner control limit.

1. Find an interval, I, containing y*, where g(y*) = min {g(y)}.
2. Select three values of y in I: y[1] < y[2] < y[3].

3. Evaluate g at each of these points and let g[i] = g(y[i]), i=1,2,3.
4. Compute d = | y[3] - y[1] | and set n = 1.
5. While (n ≤ Nmax and d > Yerror) do:
   5.1. If g(y) is not convex in the interval (y[1], y[3]) then select another set of initial values for y[1], y[2], and y[3] and go to step 3.
   5.2. If g(y) is convex in the interval (y[1], y[3]) then
      5.2.1. Carry out a quadratic interpolation through the points (y[1], g[1]), (y[2], g[2]), and (y[3], g[3]) to estimate y*. Let y' be the estimate of y*.
      5.2.2. Evaluate g at y': g' = g( y' ).
      5.2.3. Let g[k] = max{ g[1], g[2], g[3] }
      5.2.4. Replace y[k] by y' and g[k] by g'.
   5.3. Increment n by one
   5.4. Compute d = max{ y[1],y[2],y[3] } - min{ y[1],y[2],y[3] }

   End while.
6. If d ≤ Yerror then
   6.1. Let g[j] = Min{ g[1], g[2], g[3] }
   6.2. Let y* = y[j], compute x* = f(ARLo, y*) and stop.
7. If n > Nmax then the maximum number of iterations has been exceeded; stop.

In addition to the algorithms previously described, another algorithm is necessary to compute the cumulative standard normal probabilities. The normal probability approximation given by W. J.

Cody (1969) and presented and discussed by W. J. Kennedy and J. E. Gentle (1980) was used to compute the transition probabilities and initialize the matrix (I-Q).

Figure 5 shows the graph of g() for in-control average run lengths equal to 200, 300, and 500, and k = 1, that is,

$$g(y) = ARL(\, f(ARLo,y),y;\ 1\ ) + |\ ARL(\, f(ARLo,y),y;\ 0\ ) - ARLo\ |,$$

for ARLo = 200, 300, 500, and Figure 6 shows all the control-limit combinations resulting in in-control average run lengths of 200, 300, and 500, that is, ARL(x,y; 0) = ARLo, or equivalently, x = f(ARLo, y), for ARLo = 200, 300, and 500.

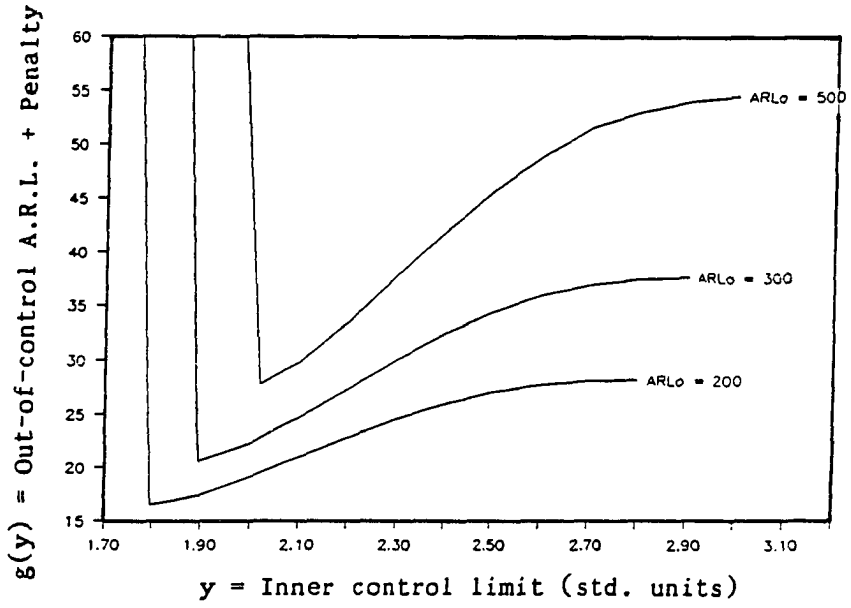



Figure 5. Graph of the modified objective function vs. the inner control limit for ARLo = 200, 300, and 500, and k = 1

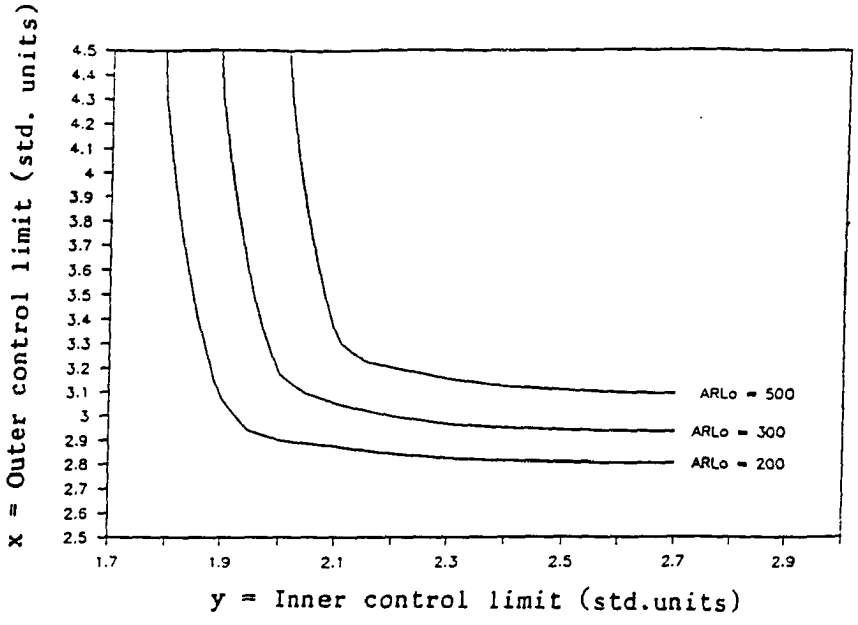



Figure 6. Control-limit combinations giving a fixed ARLo. Graph of x = f(ARLo, y), ARLo = 200, 300, 500

Program P2311A.BAS written in TURBO BASIC and listed in Appendix A implements algorithms GEVAL and GMIN to find the optimal control limits for different values of the in-control average run length. A modified version of this program was used to generate the data necessary to graph the functions shown in Figure 5 and Figure 6. A summary of the results produced by this program is presented in Table 3.

The analysis of the behavior of this control scheme for different combinations of the control limits indicates that:

a. The optimal values for the control limits correspond to relative small values of the inner control limit (Figure 5).

b. For a fixed in-control average run length, ARLo, each value of the inner control limit is associated to a value of the outer control

limit by the implicit function ARL(x,y;0) = ARLo. Moreover, for small values of the inner control limit, y, a small decrease in y causes an extremely large increase in the value of the outer control limit, x (Figure 6).

Table 3. Optimal control limits for the { R1[1,1,x,∞], R2[2,3,y,∞] } family of control schemes (output from P2311A.BAS). Values of process mean and control limits are given in standard units

| Average Run Length | | Inner Control Limit: | Outer Control Limit: |
|---|---|---|---|
| In Control $\mu = 0$ | Out of Control $\mu = 1$ | y | x |
| 100.00 | 11.461 | 1.642 | 4.081 |
| 150.00 | 14.162 | 1.725 | 4.270 |
| 200.00 | 16.533 | 1.795 | 4.310 |
| 250.00 | 18.657 | 1.853 | 4.277 |
| 300.00 | 20.615 | 1.893 | 4.313 |
| 350.00 | 22.446 | 1.927 | 4.302 |
| 400.00 | 24.156 | 1.956 | 4.317 |
| 450.00 | 25.826 | 1.981 | 4.367 |
| 500.00 | 27.406 | 2.003 | 4.298 |

These observations and the results obtained from the optimization, (the relative large values of the outer control limit) strongly suggest that, in order to find the global minimum, the out-of-control average run length should also be evaluated at the "extreme" point defined by ARL(+∞,y;0) = ARLo. Program P2311A.BAS can be easily modified to achieve this purpose. The out-of-control average run lengths obtained by the elimination of the rule "stop if a sample mean falls outside the $\pm x\sigma$ limits", that is, by setting x = +∞, are presented in Table 4. These results show that for a fixed in-control average run length, the

out-of-control run length is minimized if the outer control limit is eliminated. Figure 7 displays graphically the same results presented in Table 4.

Table 5 shows the average run lengths as functions of the process mean (in standard units) for the optimal control limits corresponding to in-control average run lengths of 100, 200, 300, 400, and 500. These results were generated using program P2311B.BAS listed in Appendix A; this program can be easily changed to compute average-run-length curves for other control limits.

Table 4. Optimal control limit for the {R2[2,3,y,∞]} family of control schemes. Values of the process mean and the control limit are given in standard units

| Average Run Length | | |
|---|---|---|
| In Control $\mu = 0$ | Out of Control $\mu = 1$ | Control Limit: y |
| 100.00 | 11.436 | 1.614 |
| 150.00 | 14.146 | 1.717 |
| 200.00 | 16.496 | 1.787 |
| 250.00 | 18.618 | 1.837 |
| 300.00 | 20.576 | 1.882 |
| 350.00 | 22.409 | 1.916 |
| 400.00 | 24.144 | 1.947 |
| 450.00 | 25.797 | 1.974 |
| 500.00 | 27.381 | 1.995 |

Table 5. Optimal average run lengths for the { R2[2,3,y,∞] } family of control schemes

| Process Mean | Optimal Control Limits (in std. units) 1.614 | 1.787 | 1.882 | 1.947 | 1.995 |
|---|---|---|---|---|---|
| $\mu$ | Average Run Length | | | | |
| 0.00 | 100.0 | 200.0 | 300.0 | 400.0 | 500.0 |
| 0.05 | 98.6 | 196.1 | 293.8 | 391.2 | 487.8 |
| 0.10 | 94.2 | 185.5 | 276.3 | 366.3 | 455.4 |
| 0.15 | 87.6 | 169.9 | 251.0 | 330.7 | 409.3 |
| 0.20 | 79.8 | 151.8 | 221.9 | 290.3 | 357.3 |
| 0.25 | 71.4 | 133.2 | 192.5 | 249.9 | 305.7 |
| 0.30 | 63.2 | 115.5 | 165.0 | 212.4 | 258.4 |
| 0.35 | 55.6 | 99.4 | 140.4 | 179.3 | 216.8 |
| 0.40 | 48.7 | 85.3 | 119.1 | 151.0 | 181.5 |
| 0.45 | 42.6 | 73.1 | 100.9 | 127.1 | 151.9 |
| 0.50 | 37.2 | 62.7 | 85.7 | 107.1 | 127.4 |
| 0.60 | 28.7 | 46.6 | 62.4 | 76.9 | 90.4 |
| 0.70 | 22.4 | 35.1 | 46.1 | 56.1 | 65.3 |
| 0.80 | 17.7 | 26.9 | 34.7 | 41.6 | 48.0 |
| 1.00 | 11.4 | 16.5 | 20.6 | 24.1 | 27.4 |
| 1.20 | 8.2 | 11.1 | 13.3 | 15.3 | 17.0 |
| 1.40 | 6.1 | 7.8 | 9.2 | 10.3 | 11.3 |
| 1.60 | 4.7 | 5.9 | 6.7 | 7.4 | 7.9 |
| 1.80 | 3.9 | 4.6 | 5.1 | 5.6 | 5.9 |

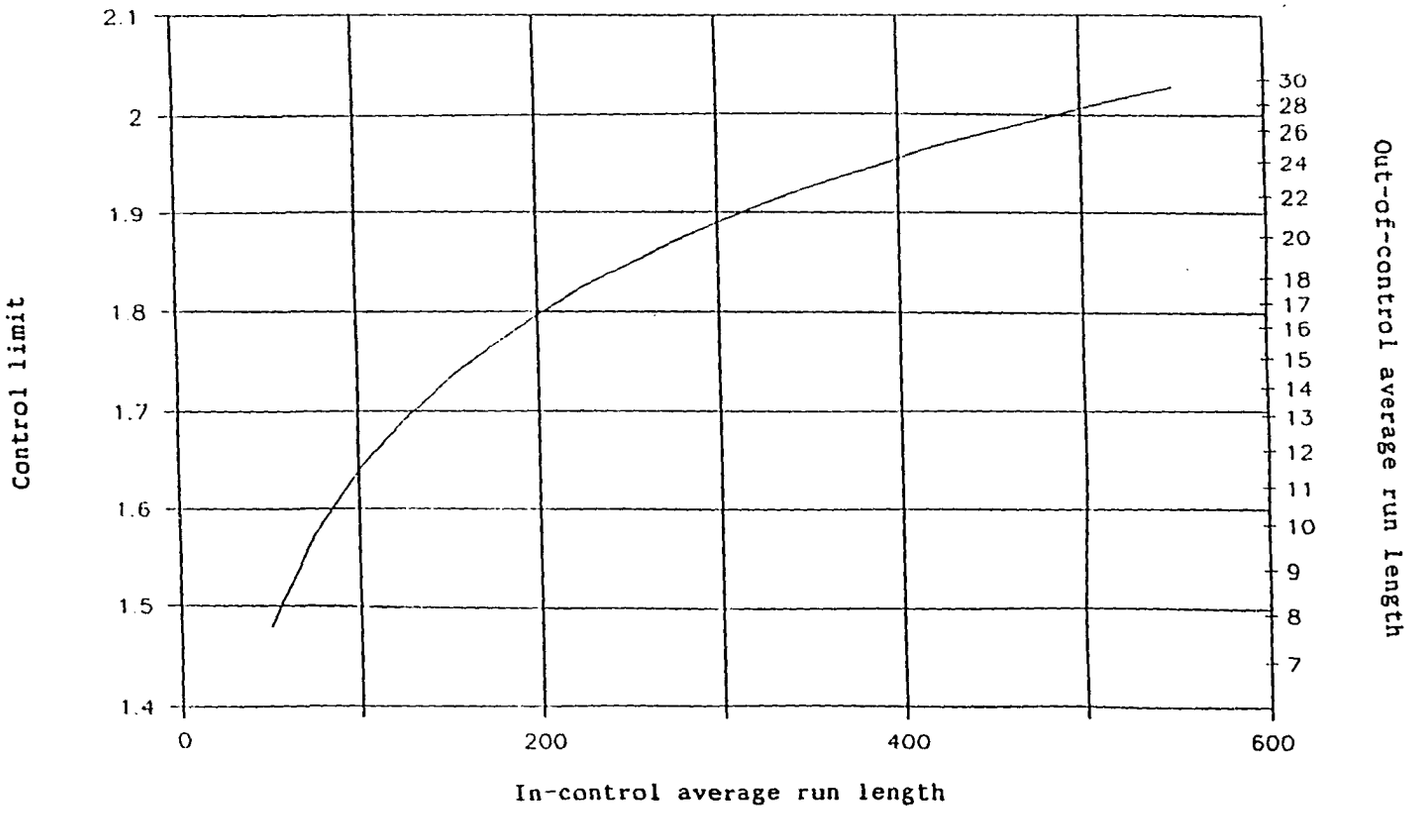



Figure 7. Optimal control limits for the { R2[2,3,y,$\alpha$] } family of control schemes. Values of the control limit are given in standard units

The optimal control schemes reduce significantly the average number of samples required to detect a shift in the process mean when they are compared to nonoptimal schemes; this fact is illustrated in Table 6. In this table, the column labeled ARL1 contains the average run lengths corresponding to the optimal control scheme giving an in-control average run length of 200; the column labeled ARL2 contains the average run lengths corresponding to the nonoptimal scheme { R1[1,1,2.843,$\infty$], R2[2,3,2.200,$\infty$] }. The comparison of these two schemes in relative terms (column ARL2/ARL1) indicates that one might expect up to a 38% reduction in the number of samples required to detect a shift of one standard deviation in the process mean and, in absolute terms (column ARL2-ARL1), one might expect to take up to 20 fewer samples to detect small shifts in the process mean. This reduction is even more accentuated for larger in-control average run lengths; the column labeled ARL3 contains the optimal control scheme giving an in-control average run length of 500; the column labeled ARL4 contains the average run lengths corresponding to the scheme { R1[1,1,3.107,$\infty$], R2[2,3,2.500,$\infty$] }. The comparison of these two schemes reveals that one might expect a reduction of up to 67% in the number of samples required to detect a moderate shift in the process mean and that, for small shifts in the process mean, one might expect to take up to 66 fewer samples with the optimal control scheme. These kinds of reductions are by no means negligible.

Table 6. Average-run-length comparisons between optimal and non-optimal control schemes

| Proc. Mean $\mu$ | ARL1 | ARL2 | ARL2 / ARL1 | ARL2-ARL1 | ARL3 | ARL4 | ARL4 / ARL3 | ARL4-ARL3 |
|---|---|---|---|---|---|---|---|---|
| 0.0 | 200.0 | 200.0 | 1.00 | .0 | 500.0 | 500.0 | 1.00 | .0 |
| 0.1 | 185.5 | 190.3 | 1.03 | 4.8 | 455.4 | 472.8 | 1.04 | 17.5 |
| 0.2 | 151.8 | 165.5 | 1.09 | 13.7 | 357.3 | 405.0 | 1.13 | 47.7 |
| 0.3 | 115.5 | 135.0 | 1.17 | 19.5 | 258.4 | 323.4 | 1.25 | 65.1 |
| 0.4 | 85.3 | 106.0 | 1.24 | 20.7 | 181.5 | 248.1 | 1.37 | 66.6 |
| 0.5 | 62.7 | 81.7 | 1.30 | 18.9 | 127.4 | 186.7 | 1.47 | 59.3 |
| 0.6 | 46.6 | 62.6 | 1.34 | 16.0 | 90.4 | 139.6 | 1.54 | 49.2 |
| 0.7 | 35.1 | 48.1 | 1.37 | 13.0 | 65.3 | 104.6 | 1.60 | 39.3 |
| 0.8 | 26.9 | 37.2 | 1.38 | 10.2 | 48.0 | 78.7 | 1.64 | 30.7 |
| 0.9 | 21.0 | 29.0 | 1.38 | 8.0 | 36.0 | 59.7 | 1.66 | 23.7 |
| 1.0 | 16.5 | 22.8 | 1.38 | 6.3 | 27.4 | 45.7 | 1.67 | 18.4 |
| 1.1 | 13.5 | 18.2 | 1.35 | 4.7 | 21.4 | 35.4 | 1.65 | 13.9 |
| 1.2 | 11.1 | 14.6 | 1.32 | 3.6 | 17.0 | 27.6 | 1.63 | 10.6 |
| 1.3 | 9.2 | 11.9 | 1.29 | 2.7 | 13.7 | 21.8 | 1.59 | 8.1 |
| 1.4 | 7.8 | 9.8 | 1.26 | 2.0 | 11.2 | 17.4 | 1.55 | 6.2 |
| 1.5 | 6.7 | 8.2 | 1.22 | 1.5 | 9.4 | 14.1 | 1.50 | 4.7 |
| 1.6 | 5.9 | 6.9 | 1.18 | 1.1 | 7.9 | 11.5 | 1.45 | 3.6 |
| 1.7 | 5.2 | 5.9 | 1.14 | 0.7 | 6.8 | 9.5 | 1.40 | 2.7 |
| 1.8 | 4.6 | 5.1 | 1.11 | 0.5 | 5.9 | 8.0 | 1.35 | 2.0 |
| 1.9 | 4.1 | 4.4 | 1.07 | 0.3 | 5.2 | 6.7 | 1.29 | 1.5 |

### B. The { R1[1,1,x,∞], R3[3,4,y,∞] } Family of Control Schemes

In this section, the following family of control schemes is studied and optimized: those control schemes that declare the process out of control if

a. a single sample mean falls x or more standard units away from the target mean,

b. three of the last four sample means fall y or more standard units above the target mean, or

c. three of the last four sample means fall y or more standard units below the target mean.

Figure 8 illustrates this type of control schemes.

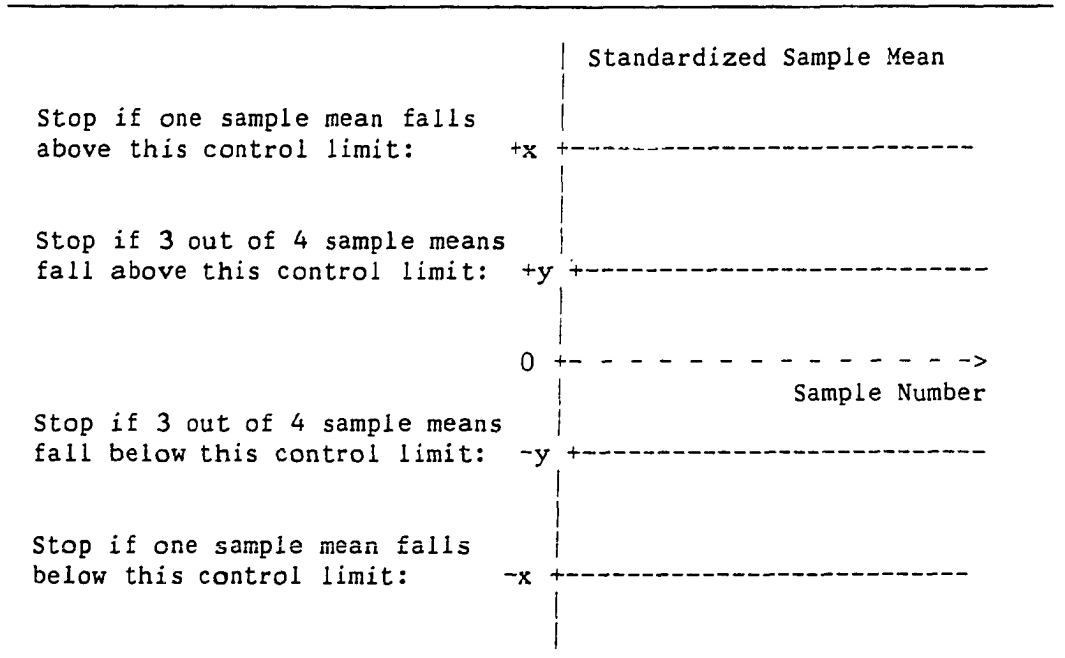



Figure 8. The { R1[1,1,x,∞], R3[3,4,y,∞] } control scheme

To define the nonabsorbing states of the Markov chain representation of this control scheme, it is necessary to keep track only of the last three sample points and their location relative to the control limits. If we let "B" represent a point falling in the interval (+y, +x), "O" represent a point falling in the interval (-y, +y), and "C" represent a point falling in the interval (-x, -y), at any time, the control chart can be thought of as being in one of the following twenty five transient states:

1 ) OOO  2 ) OOB  3 ) OOC  4 ) OBO  5 ) OBB

| | | | | |
|---|---|---|---|---|
| 6 ) OBC | 7 ) OCO | 8 ) OCB | 9 ) OCC | 10 ) BOO |
| 11 ) BOB | 12 ) BOC | 13 ) BBO | 14 ) BBC | 15 ) BCO |
| 16 ) BCB | 17 ) BCC | 18 ) COO | 19 ) COB | 20 ) COC |
| 21 ) CBO | 22 ) CBB | 23 ) CBC | 24 ) CCO | 25 ) CCB |

with the obvious notation: "CBO", state No. 21, means that the last sample is in the interval (-y, +y), the next to the last sample is in (+y, +x) and the previous one in (-x, -y).

Let $\Phi(\ )$ be the standard normal cumulative distribution function, C.D.F., and $\mu$ the process mean, then

Po = Prob{Standardized sample mean falls in the interval (-y, +y)}

$= \Phi(y-\mu) - \Phi(-y-\mu)$,

Pb = Prob{Standardized sample mean falls in the interval (+y, +x)}

$= \Phi(x-\mu) - \Phi(y-\mu)$, and

Pc = Prob{Standardized sample mean falls in the interval (-x, -y)}

$= \Phi(-y-\mu) - \Phi(-x-\mu)$,

and the one-step, nonzero, transition probabilities, p(i,j)'s, between these nonabsorbing states are:

| | | |
|---|---|---|
| p( 1, 1) = Po, | p( 1, 2) = Pb, | p( 1, 3) = Pc, |
| p( 2, 4) = Po, | p( 2, 5) = Pb, | p( 2, 6) = Pc, |
| p( 3, 7) = Po, | p( 3, 8) = Pb, | p( 3, 9) = Pc, |
| p( 4, 10) = Po, | p( 4, 11) = Pb, | p( 4, 12) = Pc, |
| p( 5, 13) = Po, | | p( 5, 14) = Pc, |
| p( 6, 15) = Po, | p( 6, 16) = Pb, | p( 6, 17) = Pc, |
| p( 7, 18) = Po, | p( 7, 19) = Pb, | p( 7, 20) = Pc, |
| p( 8, 21) = Po, | p( 8, 22) = Pb, | p( 8, 23) = Pc, |
| p( 9, 24) = Po, | p( 9, 25) = Pb, | |
| p(10, 1) = Po, | p(10, 2) = Pb, | p(10, 3) = Pc, |
| p(11, 4) = Po, | | p(11, 6) = Pc, |
| p(12, 7) = Po, | p(12, 8) = Pb, | p(12, 9) = Pc, |
| p(13, 10) = Po, | | p(13, 12) = Pc, |
| p(14, 15) = Po, | | p(14, 17) = Pc, |
| p(15, 18) = Po, | p(15, 19) = Pb, | p(15, 20) = Pc, |
| p(16, 21) = Po, | | p(16, 23) = Pc, |
| p(17, 24) = Po, | p(17, 25) = Pb, | |

```
p(18,  1) = Po,     p(18,  2) = Pb,     p(18,  3) = Pc,
p(19,  4) = Po,     p(19,  5) = Pb,     p(19,  6) = Pc,
p(20,  7) = Po,     p(20,  8) = Pb,
p(21, 10) = Po,     p(21, 11) = Pb,     p(21, 12) = Pc,
p(22, 13) = Po,                         p(22, 14) = Pc,
p(23, 15) = Po,     p(23, 16) = Pb,
p(24, 18) = Po,     p(24, 19) = Pb,
p(25, 21) = Po,     p(25, 22) = Pb.
```

The problem of determining the values of the control limits, x and y, such that, for a given in-control average run length, the out-of-control average run length is minimized, can be solved using the same techniques used in the previous section.

Program P3411A.BAS written in TURBO BASIC and listed in Appendix B can be used to find the optimal control limits for different values of the in-control average run length. A summary of the results produced by this program is presented in Table 7. Figure 9 shows the graph of

Table 7. Optimal control limits for the { R1[1,1,x,∞], R3[3,4,y,∞] } family of control schemes (output from P3411A.BAS). Values of control limits and process mean are given in standard units

| Average Run Length | | Inner Control Limit | Outer Control Limit |
|---|---|---|---|
| In Control $\mu = 0$ | Out of Control $\mu = 1$ | y | x |
| 100.0 | 10.307 | 1.139 | 4.428 |
| 150.0 | 12.303 | 1.218 | 4.542 |
| 200.0 | 13.969 | 1.279 | 4.621 |
| 250.0 | 15.431 | 1.321 | 4.681 |
| 300.0 | 16.753 | 1.357 | 4.730 |
| 350.0 | 17.971 | 1.386 | 4.775 |
| 400.0 | 19.245 | 1.421 | 4.662 |
| 450.0 | 20.599 | 1.451 | 4.624 |
| 500.0 | 22.059 | 1.479 | 4.598 |

g(), the out-of-control average run length plus the penalty function, for different values of the in-control average run length; Figure 10 shows all the control-limit combinations resulting in in-control average run lengths of 100, 200, 300, 400, and 500. A modified version of the program P3411A.BAS was used to generate the data necessary to plot these functions.

The behavior of the A.R.L. of this control scheme, { R1[1,1,x,∞], R3[3,4,y,∞] }, is very similar to that of the control scheme { R1[1,1,x,∞], R2[2,3,y,∞] } discussed in the previous section and, as before, the examination of figures 9 and 10 and Table 7 strongly suggests that, in order to find the global minimum, the out-of-control average run length should also be examined at the "extreme" point defined by the equation ARL(+∞,y; 0) = ARLo. Program P3411A.BAS can be easily modified to achieve this purpose. The out-of-control average run lengths obtained by the elimination of the rule "stop if a sample mean falls outside the $\pm x\sigma$ limits", that is, by setting x = +∞, are presented in Table 8. These results show that for a fixed in-control average run length, the out-of-control run length is minimized if the outer control limit is eliminated. Figure 11 displays the results presented in Table 8.

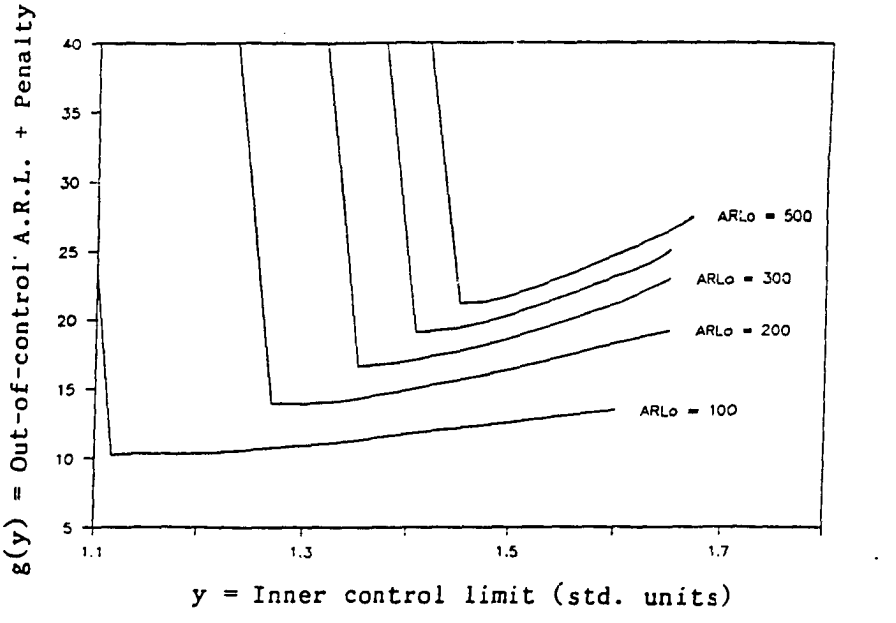



Figure 9. Graph of the modified objective function vs. the inner control limit for ARLo = 100, 200, ..., 500 and k = 1

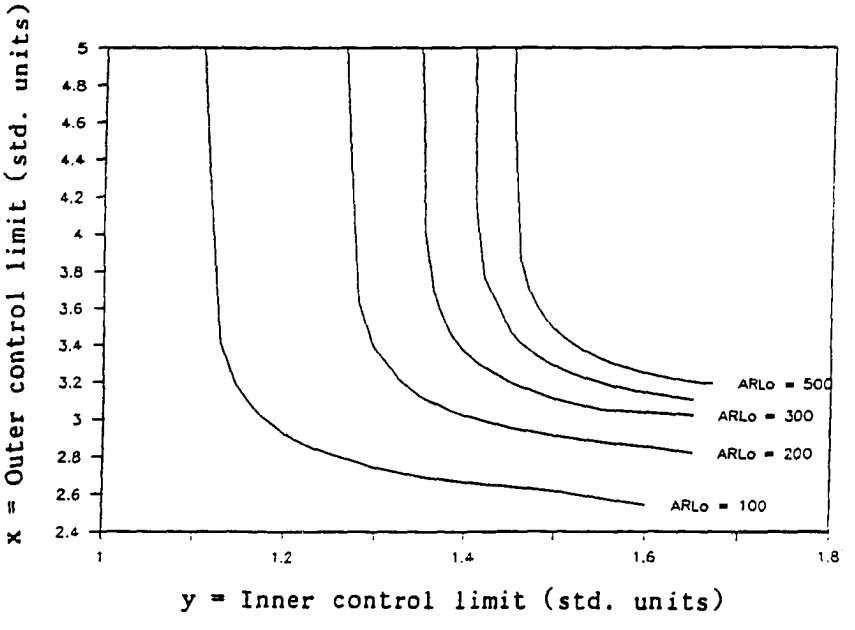



Figure 10. Control-limit combinations giving a fixed in-control average run length, ARLo; ARLo = 100, 200, ..., 500

Table 8. Optimal control limit for the { R3[3,4,y,∞] } family of control schemes. Values of the process mean and the control limit are given in standard units.

| Average Run Length | | Control Limit: y |
|---|---|---|
| In Control $\mu = 0$ | Out of Control $\mu = 1$ | |
| 100.0 | 10.264 | 1.118 |
| 125.0 | 11.339 | 1.168 |
| 150.0 | 12.303 | 1.208 |
| 175.0 | 13.166 | 1.241 |
| 200.0 | 13.967 | 1.269 |
| 225.0 | 14.714 | 1.293 |
| 250.0 | 15.420 | 1.315 |
| 275.0 | 16.095 | 1.334 |
| 300.0 | 16.740 | 1.351 |
| 325.0 | 17.357 | 1.367 |
| 350.0 | 17.951 | 1.382 |
| 375.0 | 18.527 | 1.395 |
| 400.0 | 19.105 | 1.408 |
| 425.0 | 19.650 | 1.419 |
| 450.0 | 20.218 | 1.430 |
| 475.0 | 20.701 | 1.440 |
| 500.0 | 21.195 | 1.450 |

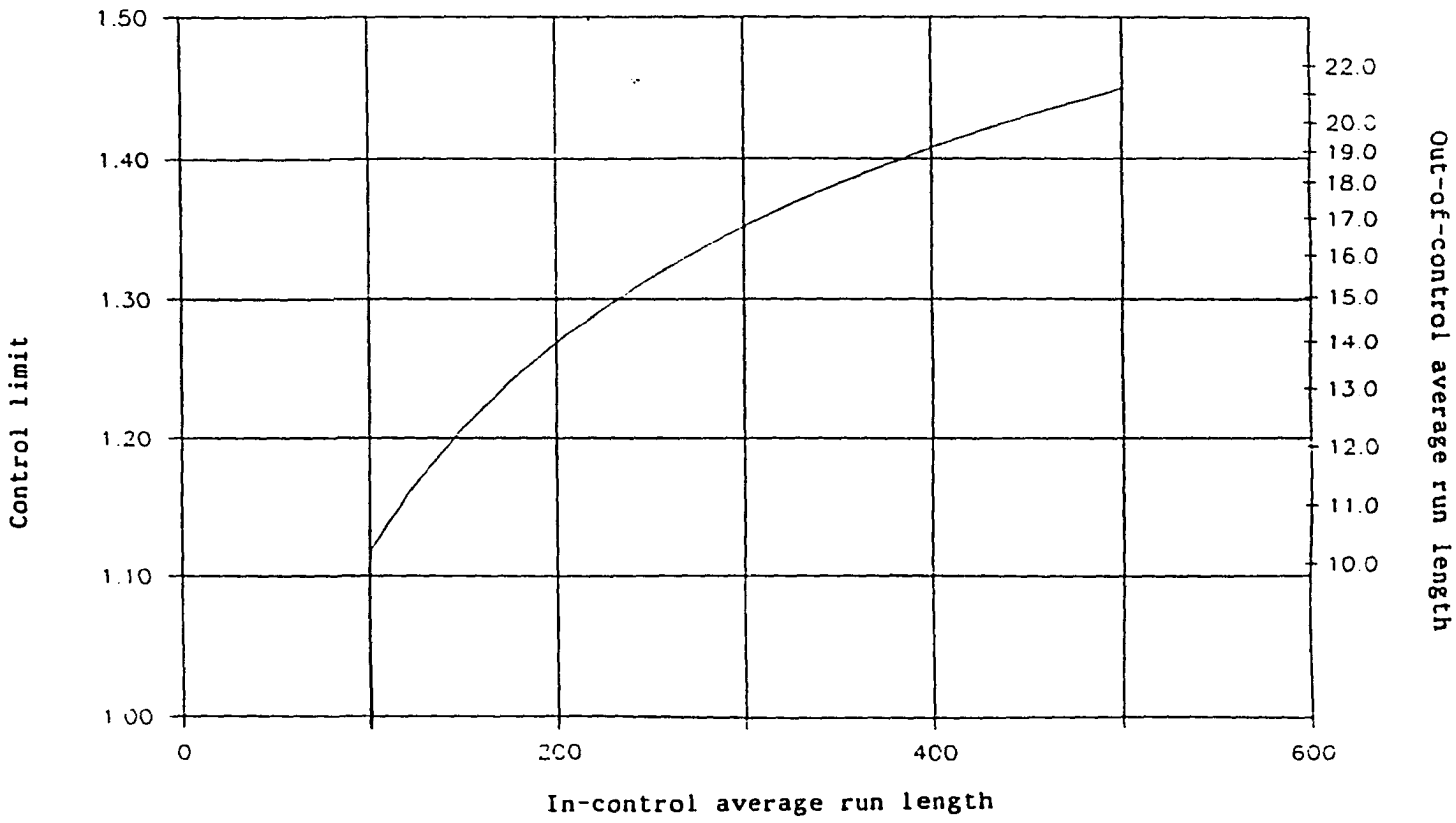



Figure 11. Optimal control limits for the { R3[ 3,4, y,∞] } family of control schemes. Values of the control limit are given in standard units

Table 9 shows the average run lengths as functions of the process mean (in standard units) for the optimal control limits corresponding to in-control average run lengths of 100, 200, 300, 400, and 500. These results were generated using program P3411B.BAS listed in Appendix B; this program can be easily changed to compute average-run-length curves for other control limits.

Table 9. Optimal average run lengths for the { R3[3,4,y,∞] } family of control schemes

| Process Mean | Optimal Control Limits (in std. units) 1.1178 | 1.2687 | 1.3515 | 1.4077 | 1.4499 |
|---|---|---|---|---|---|
| $\mu$ | Average | Run | Length | | |
| 0.00 | 100.0 | 200.0 | 300.0 | 400.0 | 500.0 |
| 0.05 | 98.6 | 195.0 | 291.9 | 388.6 | 484.8 |
| 0.10 | 93.2 | 181.7 | 269.7 | 356.9 | 443.2 |
| 0.15 | 85.3 | 162.9 | 238.9 | 313.5 | 386.7 |
| 0.20 | 76.3 | 142.1 | 205.4 | 266.9 | 326.8 |
| 0.25 | 67.1 | 121.7 | 173.3 | 222.8 | 270.7 |
| 0.30 | 58.4 | 103.1 | 144.7 | 184.2 | 222.1 |
| 0.35 | 50.6 | 87.0 | 120.4 | 151.7 | 181.6 |
| 0.40 | 43.8 | 73.4 | 100.2 | 125.1 | 148.7 |
| 0.45 | 37.9 | 62.1 | 83.6 | 103.5 | 122.1 |
| 0.50 | 32.9 | 52.7 | 70.1 | 86.0 | 100.8 |
| 0.60 | 25.2 | 38.6 | 50.1 | 60.5 | 70.0 |
| 0.70 | 19.7 | 29.0 | 36.8 | 43.7 | 50.0 |
| 0.80 | 15.7 | 22.3 | 27.7 | 32.4 | 36.6 |
| 0.90 | 12.8 | 17.6 | 21.4 | 24.7 | 27.6 |
| 1.00 | 10.3 | 14.0 | 16.7 | 19.1 | 21.2 |
| 1.20 | 7.8 | 9.8 | 11.3 | 12.5 | 13.6 |
| 1.40 | 6.1 | 7.3 | 8.2 | 8.9 | 9.5 |
| 1.60 | 5.0 | 5.8 | 6.3 | 6.7 | 7.1 |
| 1.80 | 4.3 | 4.8 | 5.2 | 5.4 | 5.7 |
| 2.00 | 3.9 | 4.2 | 4.4 | 4.6 | 4.7 |

Table 10 depicts the ratio (as a function of the shift in the process mean) of the optimal A.R.L.s of the control scheme

{ R2[2,3,y,∞] } to the optimal A.R.L.s of the control scheme { R3[3,4,y,∞] } and it shows that there is a substantial improvement in the number of samples required to detect a small or moderate shifts in the process mean when the second control scheme is used. Notice also that, for a fixed shift in the process mean, the larger the in-control average run length, the more advantageous the control scheme { R3[3,4,y,∞] } becomes.

Table 10. Ratios of average run lengths: optimal control scheme { R2[2,3,y,∞] } to optimal control scheme { R3[3,4,y,∞] }

| Process Mean $\mu$ | In - Control Average Run Length 100.00 | 200.00 | 300.00 | 400.00 | 500.00 |
|---|---|---|---|---|---|
| 0.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.20 | 1.05 | 1.07 | 1.08 | 1.09 | 1.09 |
| 0.40 | 1.11 | 1.16 | 1.19 | 1.21 | 1.22 |
| 0.60 | 1.14 | 1.21 | 1.24 | 1.27 | 1.29 |
| 0.80 | 1.13 | 1.21 | 1.25 | 1.29 | 1.31 |
| 1.00 | 1.09 | 1.17 | 1.22 | 1.26 | 1.29 |
| 1.20 | 1.05 | 1.13 | 1.18 | 1.22 | 1.25 |
| 1.40 | 1.00 | 1.07 | 1.12 | 1.15 | 1.18 |
| 1.60 | 0.95 | 1.01 | 1.06 | 1.10 | 1.12 |
| 1.80 | 0.90 | 0.96 | 0.99 | 1.03 | 1.04 |

## C. The { R2[2,3,x,∞], R3[3,4,y,∞] } Family of Control Schemes

In this section, the following family of control schemes is studied and optimized: those control schemes that declare the process out of control if

a. two out of the last three sample means fall x or more standard units above the target mean, or

b. two out of the last three sample means fall x or more standard units below the target mean, or

c. three out of the last four sample means fall y or more standard units above the target mean, or

d. three out of the last four sample means fall y or more standard units below the target mean,"

Figure 12 illustrates this type of control schemes.

To define the nonabsorbing states of the Markov chain representation of this control scheme, it is necessary to keep track of the last three sample points and their location relative to the control limits. If we let "A" represent a sample mean falling in the interval (+x, +∞), "B" represent a sample mean falling in the interval (+y, +x), "O" represent a sample mean falling in the interval (-y, +y), "C" represent a sample mean falling in the interval (-x, -y), and "D" represent a sample mean falling in the interval (-∞, -x), the control chart, at any time, can be thought of as being in one of the transient states "OOO", "OOA", ..., "DCB", where the notation should be obvious: "DCB" means

that the last sample mean is in the interval (+y, +x), the next to the last sample mean is in the interval (-x, -y), and the previous one in $(-\infty,-x)$.

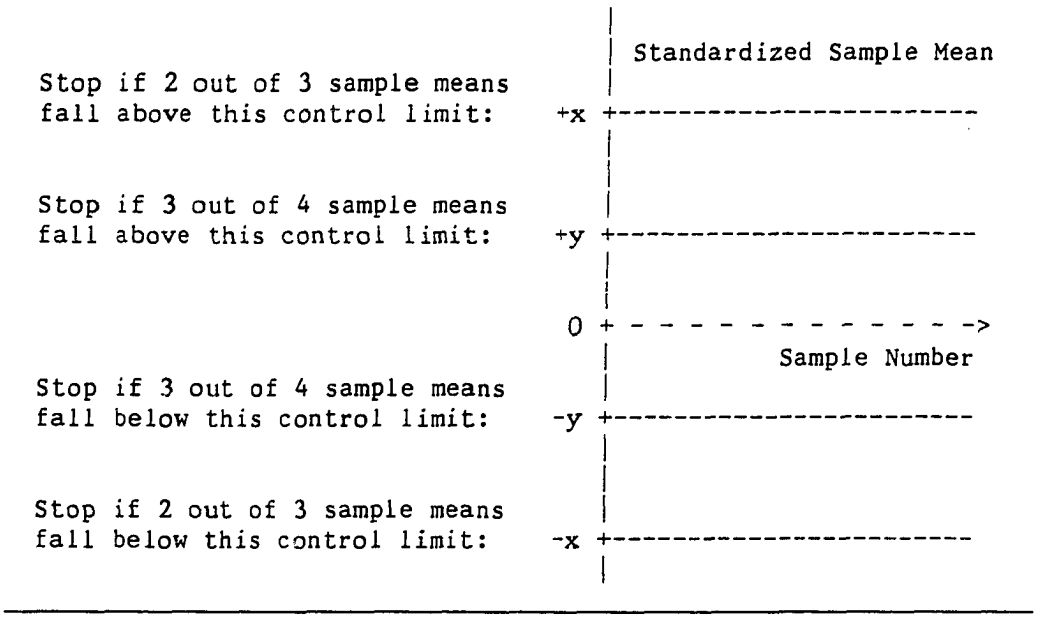



Figure 12. The { R2[2,3,x,∞], R3[3,4,y,∞] } control scheme

Notice that in order to determine the transient states of the Markov chain representation of this control scheme, it is necessary to examine the 125 possible arrangements of five objects (A, B, C, D, and O) taken 3 at a time. After these transient states have been defined, the transition probabilities among them have to be defined. This task, although simple, is very time consuming, tedious and prone to errors if done "by hand." To eliminate the possibility of a mistake and to speed up the generation of the transition probabilities, a set of subroutines

in TURBO BASIC was used to generate the transient states, their transition probabilities, and to write the code necessary to initialize the corresponding (I-Q) matrix; these subroutines (file CHECKST.INC) and the main program (file GENSTATE.BAS) are listed in Appendix C; these subroutines can be easily extended and used to generate the non-absorbing states and the transition probabilities for other control schemes. For the family of control schemes under consideration, there are 91 transient states:

| | | | | |
|---|---|---|---|---|
| 1 ) OOO | 2 ) OOA | 3 ) OOB | 4 ) OOC | 5 ) OOD |
| 6 ) OAO | | 7 ) OAB | 8 ) OAC | 9 ) OAD |
| 10 ) OBO | 11 ) OBA | 12 ) OBB | 13 ) OBC | 14 ) OBD |
| 15 ) OCO | 16 ) OCA | 17 ) OCB | 18 ) OCC | 19 ) OCD |
| 20 ) ODO | 21 ) ODA | 22 ) ODB | 23 ) ODC | |
| 24 ) AOO | | 25 ) AOB | 26 ) AOC | 27 ) AOD |
| 28 ) ABO | | | 29 ) ABC | 30 ) ABD |
| 31 ) ACO | | 32 ) ACB | 33 ) ACC | 34 ) ACD |
| 35 ) ADO | | 36 ) ADB | 37 ) ADC | |
| 38 ) BOO | 39 ) BOA | 40 ) BOB | 41 ) BOC | 42 ) BOD |
| 43 ) BAO | | | 44 ) BAC | 45 ) BAD |
| 46 ) BBO | | | 47 ) BBC | 48 ) BBD |
| 49 ) BCO | 50 ) BCA | 51 ) BCB | 52 ) BCC | 53 ) BCD |
| 54 ) BDO | 55 ) BDA | 56 ) BDB | 57 ) BDC | |
| 58 ) COO | 59 ) COA | 60 ) COB | 61 ) COC | 62 ) COD |
| 63 ) CAO | | 64 ) CAB | 65 ) CAC | 66 ) CAD |
| 67 ) CBO | 68 ) CBA | 69 ) CBB | 70 ) CBC | 71 ) CBD |
| 72 ) CCO | 73 ) CCA | 74 ) CCB | | |
| 75 ) CDO | 76 ) CDA | 77 ) CDB | | |
| 78 ) DOO | 79 ) DOA | 80 ) DOB | 81 ) DOC | |
| 82 ) DAO | | 83 ) DAB | 84 ) DAC | |
| 85 ) DBO | 86 ) DBA | 87 ) DBB | 88 ) DBC | |
| 89 ) DCO | 90 ) DCA | 91 ) DCB | | |

Let $\Phi()$ be the standard normal C.D.F and $\mu$ the process mean, then

Po = Prob{ a standardized sample mean falls in the interval (-y,+y) }

$= \Phi(y-\mu) - \Phi(-y-\mu)$,

Pa = Prob{ a standardized sample mean falls in the interval (+x,+$\infty$) }

$= 1 - \Phi(+x-\mu)$,

Pb = Prob{ a standardized sample mean falls in the interval (+y,+x) }

$= \Phi(x-\mu) - \Phi(y-\mu)$,

Pc = Prob{ a standardized sample mean falls in the interval (-x,-y) }

$= \Phi(-y-\mu) - \Phi(-x-\mu)$,

Pd = Prob{ a standardized sample mean falls in the interval $(-\infty,-x)$ }

$= \Phi(-x-\mu)$,

and the one-step, nonzero, transition probabilities, p(i,j)'s, between the nonabsorbing states are:

p( 1, 1) = Po, p( 1, 2) = Pa, p( 1, 3) = Pb, p( 1, 4) = Pc,
p( 1, 5) = Pd,
p( 2, 6) = Po, p( 2, 7) = Pb, p( 2, 8) = Pc, p( 2, 9) = Pd,
p( 3, 10) = Po, p( 3, 11) = Pa, p( 3, 12) = Pb, p( 3, 13) = Pc,
p( 3, 14) = Pd,
p( 4, 15) = Po, p( 4, 16) = Pa, p( 4, 17) = Pb, p( 4, 18) = Pc,
p( 4, 19) = Pd,
p( 5, 20) = Po, p( 5, 21) = Pa, p( 5, 22) = Pb, p( 5, 23) = Pc,
p( 6, 24) = Po, p( 6, 25) = Pb, p( 6, 26) = Pc, p( 6, 27) = Pd,
p( 7, 28) = Po, p( 7, 29) = Pc, p( 7, 30) = Pd,
p( 8, 31) = Po, p( 8, 32) = Pb, p( 8, 33) = Pc, p( 8, 34) = Pd,
p( 9, 35) = Po, p( 9, 36) = Pb, p( 9, 37) = Pc,
p(10, 38) = Po, p(10, 39) = Pa, p(10, 40) = Pb, p(10, 41) = Pc,
p(10, 42) = Pd,
p(11, 43) = Po, p(11, 44) = Pc, p(11, 45) = Pd,
p(12, 46) = Po, p(12, 47) = Pc, p(12, 48) = Pd,
p(13, 49) = Po, p(13, 50) = Pa, p(13, 51) = Pb, p(13, 52) = Pc,
p(13, 53) = Pd,
p(14, 54) = Po, p(14, 55) = Pa, p(14, 56) = Pb, p(14, 57) = Pc,
p(15, 58) = Po, p(15, 59) = Pa, p(15, 60) = Pb, p(15, 61) = Pc,
p(15, 62) = Pd,
p(16, 63) = Po, p(16, 64) = Pb, p(16, 65) = Pc, p(16, 66) = Pd,
p(17, 67) = Po, p(17, 68) = Pa, p(17, 69) = Pb, p(17, 70) = Pc,
p(17, 71) = Pd,
p(18, 72) = Po, p(18, 73) = Pa, p(18, 74) = Pb,
p(19, 75) = Po, p(19, 76) = Pa, p(19, 77) = Pb,
p(20, 78) = Po, p(20, 79) = Pa, p(20, 80) = Pb, p(20, 81) = Pc,
p(21, 82) = Po, p(21, 83) = Pb, p(21, 84) = Pc,
p(22, 85) = Po, p(22, 86) = Pa, p(22, 87) = Pb, p(22, 88) = Pc,
p(23, 89) = Po, p(23, 90) = Pa, p(23, 91) = Pb,
p(24, 1) = Po, p(24, 2) = Pa, p(24, 3) = Pb, p(24, 4) = Pc,
p(24, 5) = Pd,
p(25, 10) = Po, p(25, 13) = Pc, p(25, 14) = Pd,

```
p(26, 15) = Po,   p(26, 16) = Pa,   p(26, 17) = Pb,   p(26, 18) = Pc,
p(26, 19) = Pd,
p(27, 20) = Po,   p(27, 21) = Pa,   p(27, 22) = Pb,   p(27, 23) = Pc,
p(28, 38) = Po,   p(28, 41) = Pc,   p(28, 42) = Pd,
p(29, 49) = Po,   p(29, 52) = Pc,   p(29, 53) = Pd,
p(30, 54) = Po,   p(30, 57) = Pc,
p(31, 58) = Po,   p(31, 59) = Pa,   p(31, 60) = Pb,   p(31, 61) = Pc,
p(31, 62) = Pd,
p(32, 67) = Po,   p(32, 70) = Pc,   p(32, 71) = Pd,
p(33, 72) = Po,   p(33, 73) = Pa,   p(33, 74) = Pb,
p(34, 75) = Po,   p(34, 76) = Pa,   p(34, 77) = Pb,
p(35, 78) = Po,   p(35, 79) = Pa,   p(35, 80) = Pb,   p(35, 81) = Pc,
p(36, 85) = Po,   p(36, 88) = Pc,
p(37, 89) = Po,   p(37, 90) = Pa,   p(37, 91) = Pb,
p(38,  1) = Po,   p(38,  2) = Pa,   p(38,  3) = Pb,   p(38,  4) = Pc,
p(38,  5) = Pd,
p(39,  6) = Po,   p(39,  8) = Pc,   p(39,  9) = Pd,
p(40, 10) = Po,   p(40, 13) = Pc,   p(40, 14) = Pd,
p(41, 15) = Po,   p(41, 16) = Pa,   p(41, 17) = Pb,   p(41, 18) = Pc,
p(41, 19) = Pd,
p(42, 20) = Po,   p(42, 21) = Pa,   p(42, 22) = Pb,   p(42, 23) = Pc,
p(43, 24) = Po,   p(43, 26) = Pc,   p(43, 27) = Pd,
p(44, 31) = Po,   p(44, 33) = Pc,   p(44, 34) = Pd,
p(45, 35) = Po,   p(45, 37) = Pc,
p(46, 38) = Po,   p(46, 41) = Pc,   p(46, 42) = Pd,
p(47, 49) = Po,   p(47, 52) = Pc,   p(47, 53) = Pd,
p(48, 54) = Po,   p(48, 57) = Pc,
p(49, 58) = Po,   p(49, 59) = Pa,   p(49, 60) = Pb,   p(49, 61) = Pc,
p(49, 62) = Pd,
p(50, 63) = Po,   p(50, 65) = Pc,   p(50, 66) = Pd,
p(51, 67) = Po,   p(51, 70) = Pc,   p(51, 71) = Pd,
p(52, 72) = Po,   p(52, 73) = Pa,   p(52, 74) = Pb,
p(53, 75) = Po,   p(53, 76) = Pa,   p(53, 77) = Pb,
p(54, 78) = Po,   p(54, 79) = Pa,   p(54, 80) = Pb,   p(54, 81) = Pc,
p(55, 82) = Po,   p(55, 84) = Pc,
p(56, 85) = Po,   p(56, 88) = Pc,
p(57, 89) = Po,   p(57, 90) = Pa,   p(57, 91) = Pb,
p(58,  1) = Po,   p(58,  2) = Pa,   p(58,  3) = Pb,   p(58,  4) = Pc,
p(58,  5) = Pd,
p(59,  6) = Po,   p(59,  7) = Pb,   p(59,  8) = Pc,   p(59,  9) = Pd,
p(60, 10) = Po,   p(60, 11) = Pa,   p(60, 12) = Pb,   p(60, 13) = Pc,
p(60, 14) = Pd,
p(61, 15) = Po,   p(61, 16) = Pa,   p(61, 17) = Pb,
p(62, 20) = Po,   p(62, 21) = Pa,   p(62, 22) = Pb,
p(63, 24) = Po,   p(63, 25) = Pb,   p(63, 26) = Pc,   p(63, 27) = Pd,
p(64, 28) = Po,   p(64, 29) = Pc,   p(64, 30) = Pd,
p(65, 31) = Po,   p(65, 32) = Pb,
p(66, 35) = Po,   p(66, 36) = Pb,
p(67, 38) = Po,   p(67, 39) = Pa,   p(67, 40) = Pb,   p(67, 41) = Pc,
p(67, 42) = Pd,
```

```
p(68, 43) = Po,   p(68, 44) = Pc,   p(68, 45) = Pd,
p(69, 46) = Po,   p(69, 47) = Pc,   p(69, 48) = Pd,
p(70, 49) = Po,   p(70, 50) = Pa,   p(70, 51) = Pb,
p(71, 54) = Po,   p(71, 55) = Pa,   p(71, 56) = Pb,
p(72, 58) = Po,   p(72, 59) = Pa,   p(72, 60) = Pb,
p(73, 63) = Po,   p(73, 64) = Pb,
p(74, 67) = Po,   p(74, 68) = Pa,   p(74, 69) = Pb,
p(75, 78) = Po,   p(75, 79) = Pa,   p(75, 80) = Pb,
p(76, 82) = Po,   p(76, 83) = Pb,
p(77, 85) = Po,   p(77, 86) = Pa,   p(77, 87) = Pb,
p(78,  1) = Po,   p(78,  2) = Pa,   p(78,  3) = Pb,   p(78,  4) = Pc,
p(78,  5) = Pd,
p(79,  6) = Po,   p(79,  7) = Pb,   p(79,  8) = Pc,   p(79,  9) = Pd,
p(80, 10) = Po,   p(80, 11) = Pa,   p(80, 12) = Pb,   p(80, 13) = Pc,
p(80, 14) = Pd,
p(81, 15) = Po,   p(81, 16) = Pa,   p(81, 17) = Pb,
p(82, 24) = Po,   p(82, 25) = Pb,   p(82, 26) = Pc,   p(82, 27) = Pd,
p(83, 28) = Po,   p(83, 29) = Pc,   p(83, 30) = Pd,
p(84, 31) = Po,   p(84, 32) = Pb,
p(85, 38) = Po,   p(85, 39) = Pa,   p(85, 40) = Pb,   p(85, 41) = Pc,
p(85, 42) = Pd,
p(86, 43) = Po,   p(86, 44) = Pc,   p(86, 45) = Pd,
p(87, 46) = Po,   p(87, 47) = Pc,   p(87, 48) = Pd,
p(88, 49) = Po,   p(88, 50) = Pa,   p(88, 51) = Pb,
p(89, 58) = Po,   p(89, 59) = Pa,   p(89, 60) = Pb,
p(90, 63) = Po,   p(90, 64) = Pb,
p(91, 67) = Po,   p(91, 68) = Pa,   p(91, 69) = Pb.
```

The problem of finding the values of the control limits, x and y, such that, for a given in-control A.R.L., the out-of-control A.R.L. is minimized, can be solved using penalty functions and inverse parabolic interpolation to approximate the global minimum. Program P3423A.BAS written in TURBO BASIC and listed in Appendix D can be used to find the optimal control limits for different values of the in-control A.R.L. The results produced by this program are presented in Table 11 and Figure 13.

Figure 14 shows the control-limit pairs resulting in in-control average run lengths of 100, 200, ..., 500, that is, the graph of ARL(x,y;0) = ARLo (or equivalently, x = f(ARLo,y) ) for ARLo = 100,

200, ..., 500. Figure 15 shows the out-of-control A.R.L., ARL1, as a function of the inner control limit, y, and the in-control A.R.L., ARLo, that is, ARL1 = ARL(x,y; 1) where the value of x is selected such that ARL(x,y;0) = ARLo. This graph clearly indicates that the optimal value for the inner control limit occurs at an interior point of the feasible region.

Table 11. Optimal control limits for the { R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes (output from P2334A.BAS). Values of control limits are given in standard units

| Average Run Length | | Control Limits | |
|---|---|---|---|
| In Control | Out of Control | y | x |
| 100.00 | 9.95 | 1.186 | 1.960 |
| 125.00 | 10.93 | 1.228 | 2.018 |
| 150.00 | 11.83 | 1.263 | 2.064 |
| 175.00 | 12.65 | 1.292 | 2.104 |
| 200.00 | 13.42 | 1.318 | 2.138 |
| 225.00 | 14.13 | 1.340 | 2.169 |
| 250.00 | 14.81 | 1.360 | 2.196 |
| 275.00 | 15.46 | 1.378 | 2.220 |
| 300.00 | 16.08 | 1.395 | 2.243 |
| 325.00 | 16.67 | 1.410 | 2.263 |
| 350.00 | 17.24 | 1.424 | 2.282 |
| 375.00 | 17.79 | 1.437 | 2.300 |
| 400.00 | 18.32 | 1.449 | 2.317 |
| 425.00 | 18.84 | 1.461 | 2.332 |
| 450.00 | 19.34 | 1.471 | 2.347 |
| 475.00 | 19.82 | 1.482 | 2.361 |
| 500.00 | 20.30 | 1.491 | 2.374 |

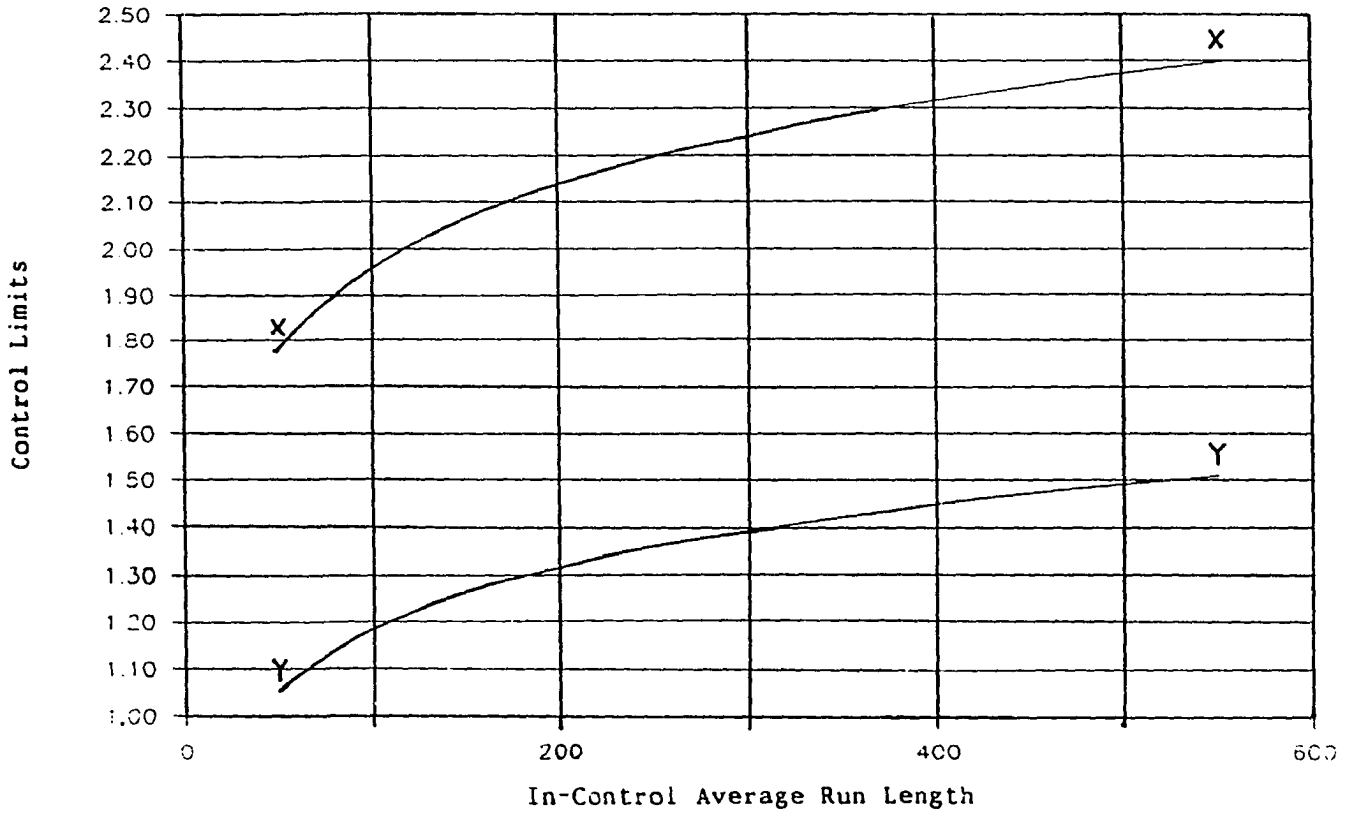



Figure 13. Optimal control limits for the { R2[ 2,3, x,∞], R3[ 3,4, y,∞] } family of control schemes. Values of the control limits are given in standard units

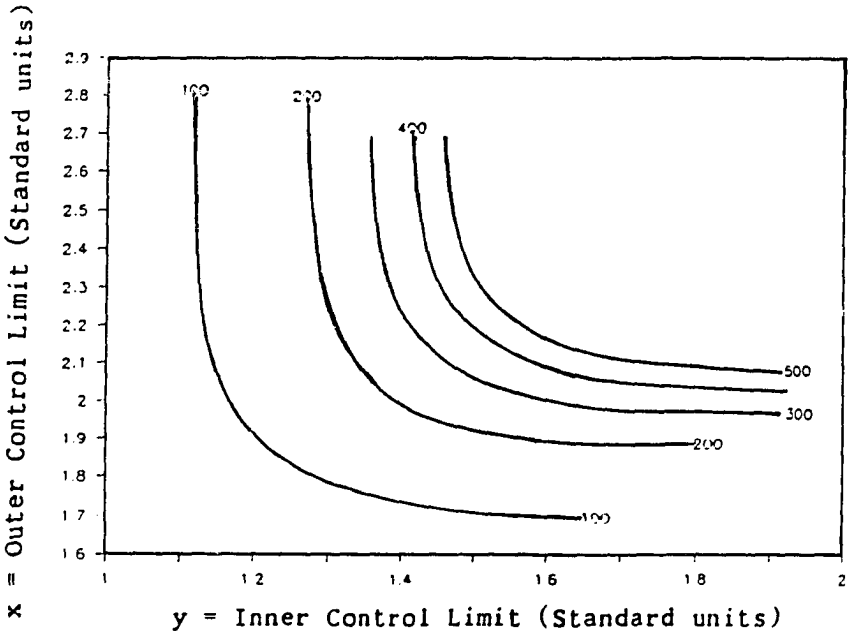



Figure 14. Control limit pairs for the control scheme { R2[2,3,x,∞], R3[3,4,y,∞] } giving a fixed in-control average run length, ARLo; ARLo = 100, 200, ..., 500

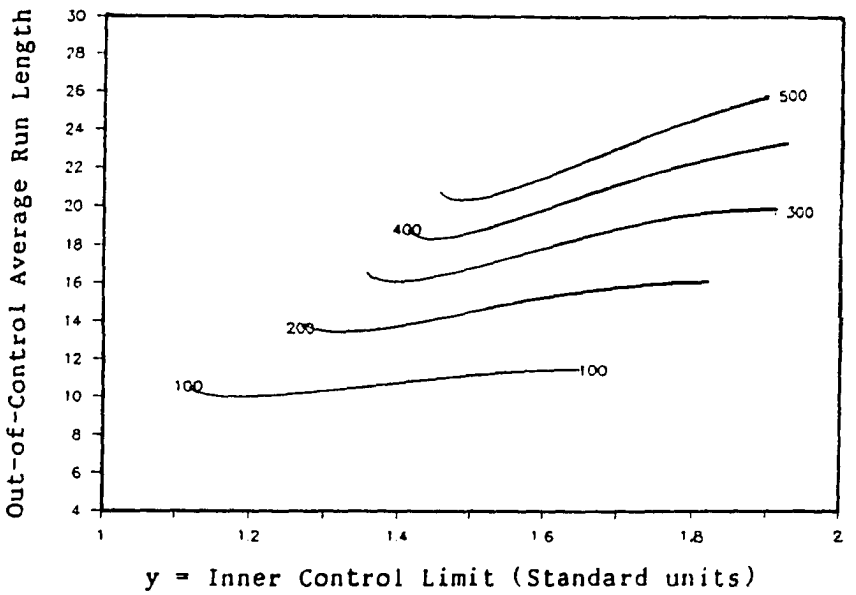



Figure 15. Graph of the out-of-control average run length vs. the inner control limit for ARLo = 100, 200, ..., 500

Table 12 shows the average run lengths, as functions of the shift in the process mean (in standard units), corresponding to the optimal control limits for in-control average run lengths of 100, 200, ..., 500. These results were generated using the program P3423B.BAS listed in Appendix D; this program can be easily modified to compute A.R.L. curves for other in-control average run lengths.

Table 12. Optimal average run lengths for the { R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes

| Process Mean | Optimal Control Limits (in std. units) y = 1.186 x = 1.960 | 1.318 2.138 | 1.395 2.243 | 1.449 2.317 | 1.491 2.374 |
|---|---|---|---|---|---|
| | Average Run Length | | | | |
| 0.00 | 100.0 | 200.0 | 300.0 | 400.0 | 500.0 |
| 0.05 | 98.0 | 195.1 | 291.7 | 388.1 | 484.4 |
| 0.10 | 92.5 | 181.5 | 269.2 | 356.0 | 442.4 |
| 0.15 | 84.6 | 162.4 | 238.0 | 312.2 | 385.5 |
| 0.20 | 75.4 | 141.3 | 204.1 | 265.2 | 325.1 |
| 0.25 | 66.1 | 120.6 | 171.8 | 221.0 | 268.8 |
| 0.30 | 57.4 | 101.9 | 143.1 | 182.2 | 220.1 |
| 0.35 | 49.6 | 85.8 | 118.7 | 149.8 | 179.6 |
| 0.40 | 42.7 | 72.1 | 98.5 | 123.1 | 146.6 |
| 0.45 | 36.9 | 60.8 | 82.0 | 101.6 | 120.1 |
| 0.50 | 31.9 | 51.5 | 68.5 | 84.2 | 99.0 |
| 0.60 | 24.3 | 37.5 | 48.7 | 58.9 | 68.4 |
| 0.70 | 18.8 | 28.0 | 35.6 | 42.3 | 48.5 |
| 0.80 | 14.9 | 21.4 | 26.6 | 31.2 | 35.4 |
| 0.90 | 12.1 | 16.7 | 20.4 | 23.6 | 26.5 |
| 1.00 | 10.0 | 13.4 | 16.1 | 18.3 | 20.3 |
| 1.25 | 6.7 | 8.4 | 9.7 | 10.7 | 11.6 |
| 1.50 | 5.0 | 5.9 | 6.6 | 7.1 | 7.6 |
| 1.75 | 3.9 | 4.5 | 4.9 | 5.2 | 5.5 |
| 2.00 | 3.3 | 3.7 | 3.9 | 4.1 | 4.3 |
| 2.25 | 2.9 | 3.2 | 3.3 | 3.5 | 3.6 |
| 2.50 | 2.6 | 2.8 | 3.0 | 3.0 | 3.1 |
| 2.75 | 2.4 | 2.6 | 2.7 | 2.7 | 2.8 |
| 3.00 | 2.3 | 2.4 | 2.5 | 2.5 | 2.5 |

Table 13 shows the ratios of the optimal A.R.L.s corresponding to the control scheme { R3[3,4,y,∞] } to those corresponding to the control scheme { R2[2,3,x,∞], R3[3,4,y,∞] } as functions of the shift in the process mean. These ratios show that some improvement (in the average run length) over the single rule "three out of four" is attained by combining the "two out of three" and the "three out of four" control rules. However, this reduction in the A.R.L. is almost negligible for small shifts in the process mean (less than one standard deviation) and large in-control average run lengths (larger than 300). On the other hand, the control scheme { R2[2,3,x,∞], R3[3,4,y,∞] } is clearly superior to the control scheme { R3[3,4,y,∞] } for the detection of large shifts in the process mean.

Table 13. Ratios of average run lengths: optimal control scheme { R3[3,4,y,∞] } to optimal control scheme { R2[2,3,x,∞], R3[3,4,y,∞] }

| Process Mean | In - Control Average Run Length | | | | |
|---|---|---|---|---|---|
| | 100 | 200 | 300 | 400 | 500 |
| 0.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.10 | 1.01 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.20 | 1.01 | 1.01 | 1.01 | 1.01 | 1.01 |
| 0.30 | 1.02 | 1.01 | 1.01 | 1.01 | 1.01 |
| 0.40 | 1.03 | 1.02 | 1.02 | 1.02 | 1.01 |
| 0.50 | 1.03 | 1.02 | 1.02 | 1.02 | 1.02 |
| 0.60 | 1.04 | 1.03 | 1.03 | 1.03 | 1.02 |
| 0.70 | 1.05 | 1.04 | 1.04 | 1.03 | 1.03 |
| 0.80 | 1.05 | 1.04 | 1.04 | 1.04 | 1.04 |
| 0.90 | 1.06 | 1.05 | 1.05 | 1.04 | 1.04 |
| 1.00 | 1.06 | 1.06 | 1.05 | 1.05 | 1.05 |
| 1.25 | 1.09 | 1.08 | 1.07 | 1.07 | 1.06 |
| 1.50 | 1.12 | 1.09 | 1.08 | 1.08 | 1.07 |
| 1.75 | 1.15 | 1.11 | 1.10 | 1.10 | 1.08 |
| 2.00 | 1.18 | 1.14 | 1.13 | 1.12 | 1.10 |

## D. The { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } Family of Control Schemes

The last family of control schemes studied in this Chapter is the family comprised of control schemes that declare the process out of control if

a. a single sample mean falls w or more standard units away from the sample mean, or

b. two out of the last three sample means fall x or more standard units above the target mean, or

c. two out of the last three sample means fall x or more standard units below the target mean, or

d. three out of the last four sample means fall y or more standard units above the target mean, or

e. three out of the last four sample means fall y or more standard units below the target mean."

Figure 16 illustrates this type of control schemes.

To define the nonabsorbing states of the Markov chain representation of this control scheme, it is necessary to keep track of the last three sample points and their location relative to the control limits. If we let "A" represent a sample mean falling in the interval (+x, +w), "B" represent a sample mean falling in the interval (+y, +x), "O" represent a sample mean falling in the interval (-y, +y), "C" represent a sample mean falling in the interval (-x, -y), and "D" represent a sample mean falling in the interval (-w, -x), the control chart, at any

time, can be thought of as being in one of the transient states "OOO", "OOA", ..., "DCB", where the notation is the same one used in the previous section; for example, "OOA" means that the last sample mean is in the interval (+x, +w) and the two previous ones are in the interval (-y, +y).

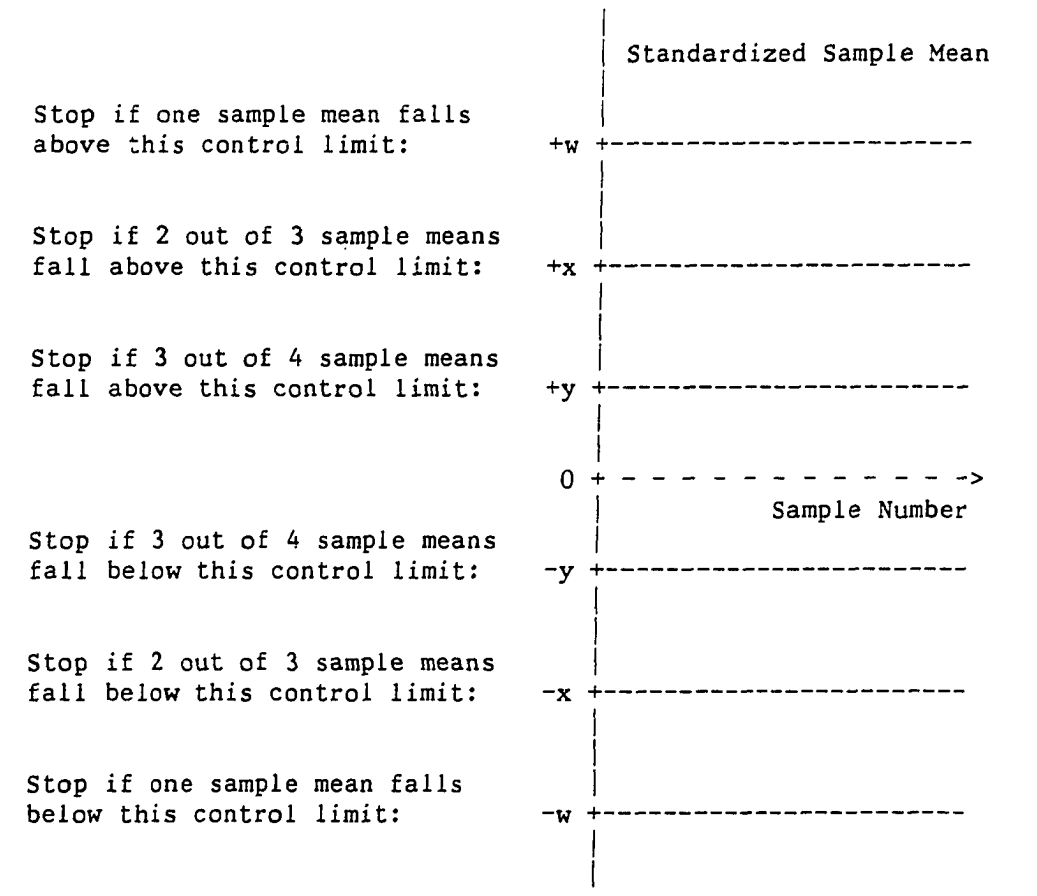



Figure 16. The { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } control scheme

Notice that the transient states of the Markov chain representation of this control scheme correspond to the same 91 transient states of the control scheme discussed in the previous section, { R2[2,3,x,∞], R3[3,4,y,∞] }. No more transient states are generated by the inclusion of the control rule R1[1,1,w,∞] because this control rule does not require any prior information about the status of the control chart to indicate an out-of-control situation since this rule generates an out-of-control signal based entirely on the current sample mean. Moreover, the transition probabilities among the transient states corresponding to the family { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } are the same as those corresponding to the family { R2[2,3,x,∞], R3[3,4,y,∞] } as long as the definitions of Pa and Pd are modified accordingly.

In summary, if Po, Pa, Pb, Pc, and Pd are defined as follows,

$Po = Prob\{$ a standardized sample mean falls in the interval $(-y,+y)\}$

$= \Phi(y-\mu) - \Phi(-y-\mu)$,

$Pa = Prob\{$ a standardized sample mean falls in the interval $(+x,+\infty)\}$

$= \Phi(+w-\mu) - \Phi(+x-\mu)$,

$Pb = Prob\{$ a standardized sample mean falls in the interval $(+y,+x)\}$

$= \Phi(x-\mu) - \Phi(y-\mu)$,

$Pc = Prob\{$ a standardized sample mean falls in the interval $(-x,-y)\}$

$= \Phi(-y-\mu) - \Phi(-x-\mu)$,

$Pd = Prob\{$ a standardized sample mean falls in the interval $(-\infty,-x)\}$

$= \Phi(-x-\mu) - \Phi(-w-\mu)$

(where $\Phi()$ is the standard normal C.D.F. and $\mu$ is the process mean), then the one-step, nonzero, transition probability between state i and state j, p(i,j), i = 1,2,3, ..., 91, j = 1,2,3, ..., 91, can be computed using the same formulas given on pages 85 to 87.

For given values of the control limits and the process mean, the expected run length (given that the chart starts in state "000") is the first component of the vector

$$\underline{ET} = (I - Q)^{-1} \underline{e}$$

where I is a 91x91 identity matrix, Q is a 91x91 matrix containing the transition probabilities between the transient states, and $\underline{e}$ is a 91-dimensional vector of 1's; consequently, the expected run length is a function of the process mean $\mu$ and the three control limits w, x, and y, that is, ARL = ARL(w,x,y; $\mu$). The problem is to determine the values of the control limits such that for a given in-control average run length, the out-of-control average run length is minimized; that is,

$$\text{Minimize} \quad h(w,x,y) = ARL(w,x,y;\ 1) \quad \text{subject to}$$

$$ARL(w,x,y;\ 0) = ARLo, \quad \text{and} \quad 0 \le y \le x \le w.$$

For a given in-control average run length, ARLo, the constraint ARL(w,x,y; 0) = ARLo can be used to express one of the control limits, say w, as a function of the other two control limits: w = f(x,y; ARLo); then the minimization problem becomes:

$$\text{Minimize} \quad g(x,y) = ARL(\ f(x,y;\ ARLo),\ x,\ y;\ 1) \quad \text{subject to}$$

$$0 \le y \le x \le f(x,y;\ ARLo).$$

Program P342311A.BAS written in TURBO BASIC and listed in Appendix E can be used to evaluate the objective function, g(x,y), (for a given ARLo) at different pairs (x,y) satisfying the inequality constraints. Figure 17 shows the graph of the out-of-control average run length as a function of the control limits x and y; the value of the outer control limit, w (not shown in the figure), was selected such that ARL(w,x,y; 0) = 200.

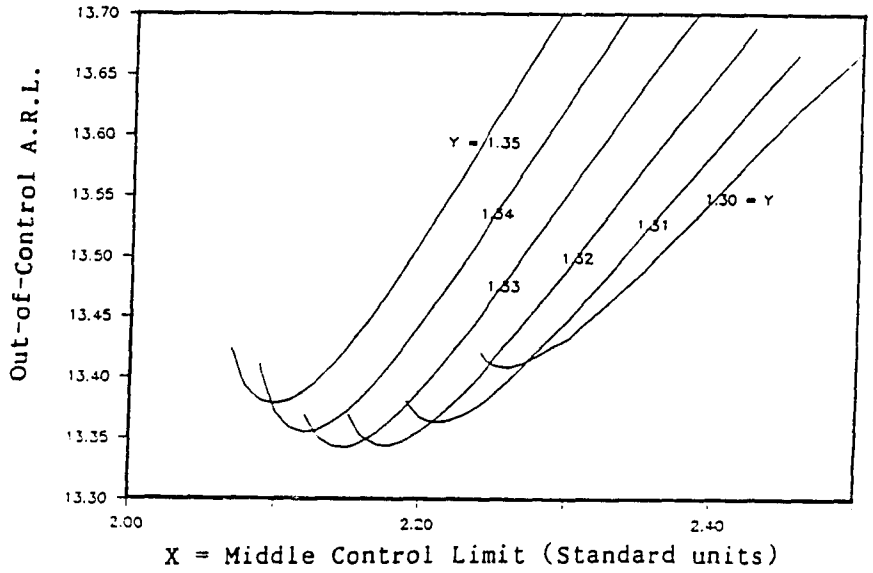



Figure 17. Out-of-control average run length (for an in-control A.R.L. of 200) as a function of the inner control limit, y, and the middle control limit, x

Figure 17 shows that, for ARLo = 200, the optimal value for the inner control limit, y, lies in the interval (1.31, 1.34) and that the optimal value for the middle control limit, x, lies in the interval (2.10, 2.20); moreover, for every fixed value of y, it is possible to

find a value for the middle control limit x such that the out-of-control average run length is minimized. Table 14 shows control-limit combinations in which the outer control limit, w, has been selected such that the in-control A.R.L. is equal to 200 and the middle control limit, x, has been selected such that, for the given value of the inner control limit, y, the out-of-control A.R.L. is minimized.

Table 14. Control limits for the family of control schemes { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } resulting in an in-control A.R.L. of 200. Control limits given in standard units

| Control Limits | | | Out-of-Control ARL |
|---|---|---|---|
| y | x | w | |
| 1.27 | 2.845 | 4.525 | 14.022 |
| 1.28 | 2.430 | 4.264 | 13.646 |
| 1.29 | 2.328 | 3.979 | 13.493 |
| 1.30 | 2.252 | 3.951 | 13.410 |
| 1.31 | 2.210 | 3.819 | 13.364 |
| 1.32 | 2.177 | 3.732 | 13.344 |
| 1.33 | 2.150 | 3.666 | (13.343) |
| 1.34 | 2.120 | 3.661 | 13.355 |
| 1.35 | 2.100 | 3.620 | 13.379 |
| 1.40 | 2.025 | 3.528 | 13.604 |

From Table 14, it is possible to obtain an approximation to the optimal solution to the problem

Min { ARL(w,x,y; 1) | ARL(w,x,y; 0) = 200, $0 \leq y \leq x \leq w$ };

this solution is

w = 3.666, x = 2.15, y = 1.33, and ARL(3.666, 2.15, 1.33; 1) = 13.343.

If a better approximation to the optimal is desired, the data presented in Table 14 and program P342311A.BAS can be used to obtain it; a better

value for the inner control limit, y, can be computed using inverse parabolic interpolation on the three best points,

$$y = 1.33 - \frac{(1.33-1.32)^2\,(13.343-13.355) - (1.33-1.34)^2\,(13.343-13.344)}{2[\,(1.33-1.32)(13.343-13.355)-(1.33-1.34)(13.343-13.344)\,]} =$$

$$= 1.33 - (-.0000011 \;/\; -.00026) = 1.325,$$

and the best values for x and w corresponding to this new value of y are 2.152 and 3.701, respectively. This procedure can be repeated until a specified tolerance is achieved. For practical purposes, two correct decimal places in the values of the control limits are sufficient.

Figure 18 shows the graph of the out-of-control A.R.L. as a function of the inner and middle control limits; the value of the outer control limit, not shown in the figure, was selected such that the in-control A.R.L. equals 400. The data necessary to graph this function was generated by the program P342311A.BAS. This figure shows that the optimal value for the middle control limit, x, lies in the interval (2.2, 2.4) and the optimal value for the inner control limit, y, lies in the interval (1.45, 1.50).

Table 15 shows control-limit combinations in which the outer control limit has been selected such that the in-control A.R.L. equals 400 and the middle control limit has been selected such that it minimizes the out-of-control A.R.L. for the given value of the inner control limit. From this table, it is possible to obtain an approximate optimal solution to the problem

$$\text{Min } \{\ ARL(w,x,y;\ 1)\ \mid\ ARL(w,x,y;\ 0) = 400,\ 0 \le y \le x \le w\ \};$$

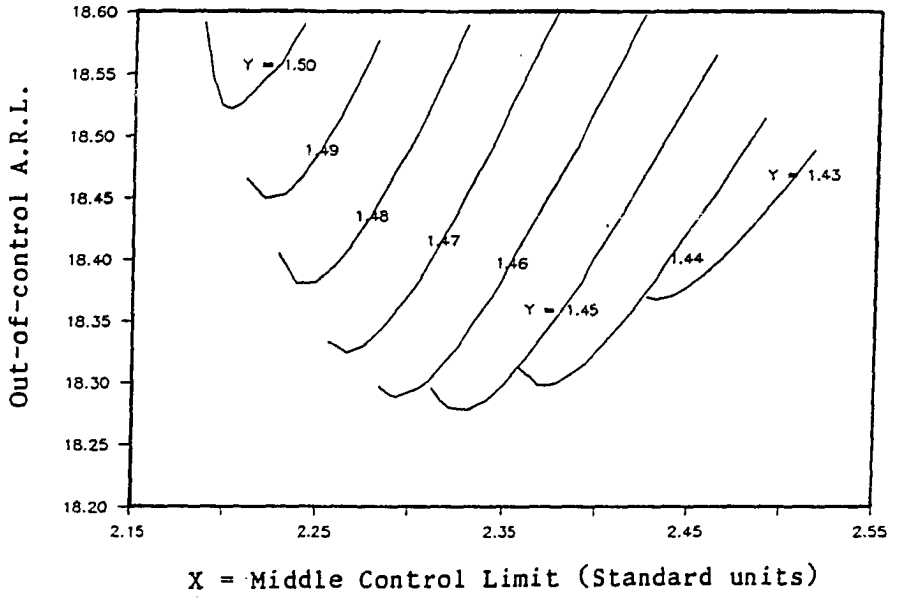



Figure 18. Out-of-control average run length (for an in-control A.R.L. of 400) as a function of the inner control limit, y, and the middle control limit, x

Table 15. Control limits for the { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes resulting in an in-control A.R.L. of 400. Control limits are given in standard units

| Control Limits | | | Out-of-Control ARL |
|---|---|---|---|
| y | x | w | |
| 1.420 | 2.529 | 4.348 | 18.514 |
| 1.430 | 2.445 | 4.218 | 18.385 |
| 1.435 | 2.409 | 4.167 | 18.340 |
| 1.440 | 2.378 | 4.125 | 18.308 |
| 1.445 | 2.351 | 4.091 | 18.286 |
| 1.450 | 2.327 | 4.062 | 18.275 |
| 1.455 | 2.306 | 4.039 | (18.273) |
| 1.460 | 2.288 | 4.020 | 18.280 |
| 1.465 | 2.273 | 4.004 | 18.295 |
| 1.470 | 2.260 | 3.990 | 18.316 |
| 1.475 | 2.249 | 3.978 | 18.344 |
| 1.480 | 2.240 | 3.966 | 18.376 |
| 1.490 | 2.224 | 3.937 | 18.454 |
| 1.500 | 2.212 | 3.897 | 18.542 |

this solution is

w=4.039, x=2.306, y=1.455, and ARL(4.039, 2.306, 1.455; 1)=18.273.

Table 16 shows the optimal control-limit combinations for different values of the in-control average run length and the corresponding expected out-of-control run lengths. These results are also displayed in figures 19, 20, 21, and 22.

Table 16. Optimal control limits for the { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes. Values of control limits are given in standard units

| Average Run Length | | Optimal Control Limits | | |
|---|---|---|---|---|
| In Control | Out of Control | y | x | w |
| 100.0 | 9.9 | 1.198 | 1.985 | 3.336 |
| 125.0 | 10.8 | 1.239 | 2.039 | 3.453 |
| 150.0 | 11.7 | 1.272 | 2.082 | 3.550 |
| 175.0 | 12.6 | 1.301 | 2.119 | 3.631 |
| 200.0 | 13.3 | 1.325 | 2.152 | 3.701 |
| 225.0 | 14.1 | 1.347 | 2.180 | 3.764 |
| 250.0 | 14.8 | 1.366 | 2.205 | 3.819 |
| 275.0 | 15.4 | 1.384 | 2.228 | 3.869 |
| 300.0 | 16.0 | 1.400 | 2.249 | 3.915 |
| 325.0 | 16.6 | 1.414 | 2.268 | 3.958 |
| 350.0 | 17.2 | 1.428 | 2.286 | 3.997 |
| 375.0 | 17.7 | 1.441 | 2.303 | 4.033 |
| 400.0 | 18.3 | 1.453 | 2.318 | 4.067 |
| 425.0 | 18.8 | 1.464 | 2.333 | 4.099 |
| 450.0 | 19.3 | 1.474 | 2.347 | 4.129 |
| 475.0 | 19.8 | 1.484 | 2.360 | 4.158 |
| 500.0 | 20.3 | 1.494 | 2.372 | 4.185 |

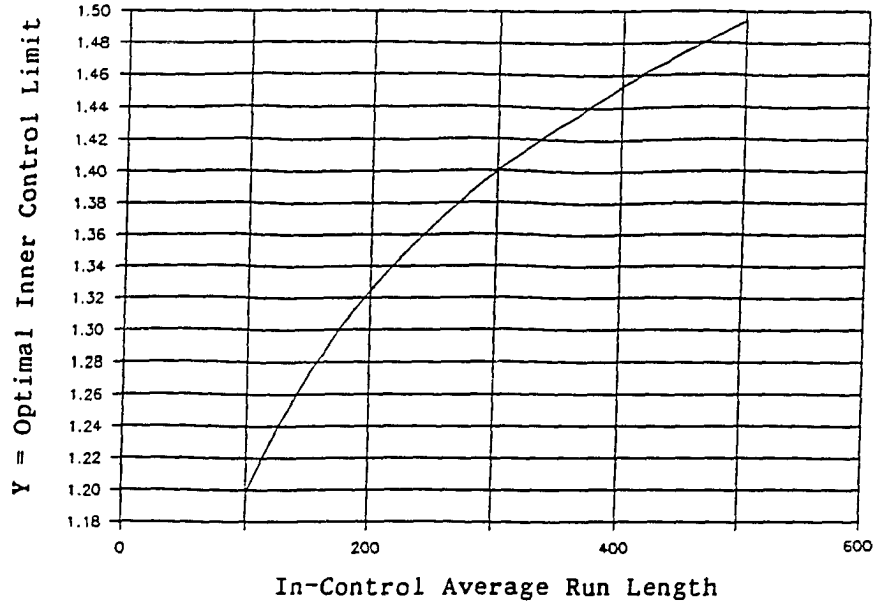



Figure 19. Optimal inner limits, y, for the { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes. Values of control limits are given in standard units

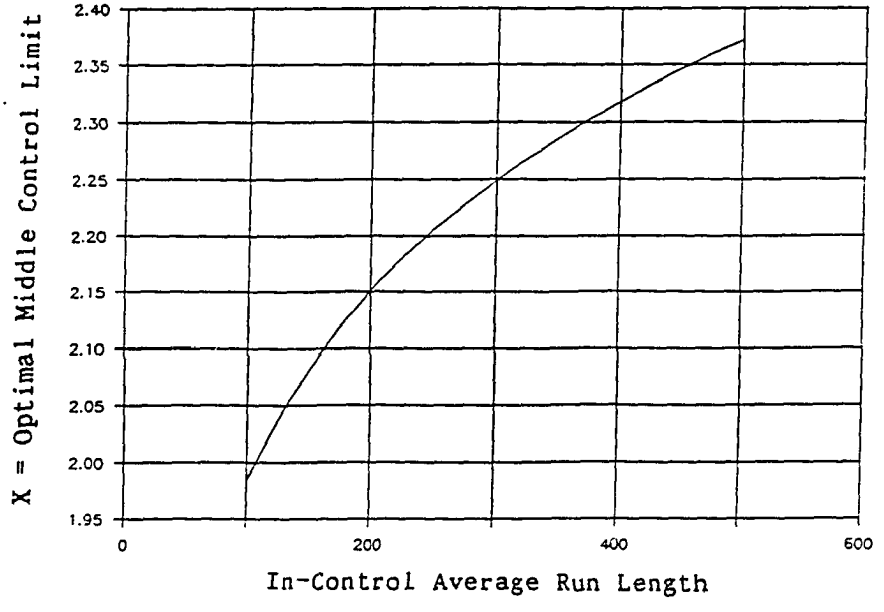



Figure 20. Optimal middle limits, x, for the { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes. Values of control limits are given in standard units

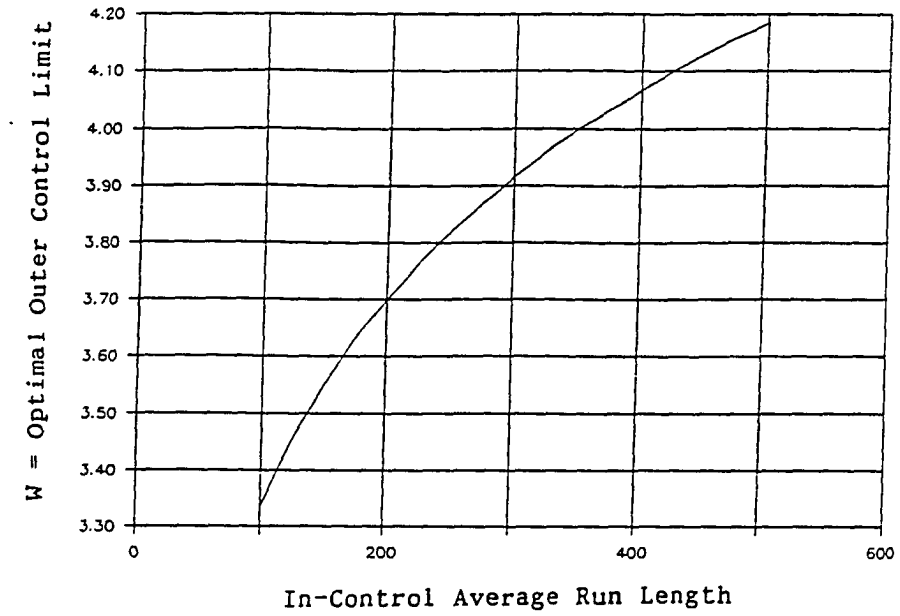



Figure 21. Optimal outer limits, w, for the { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes. Values of control limits are given in standard units

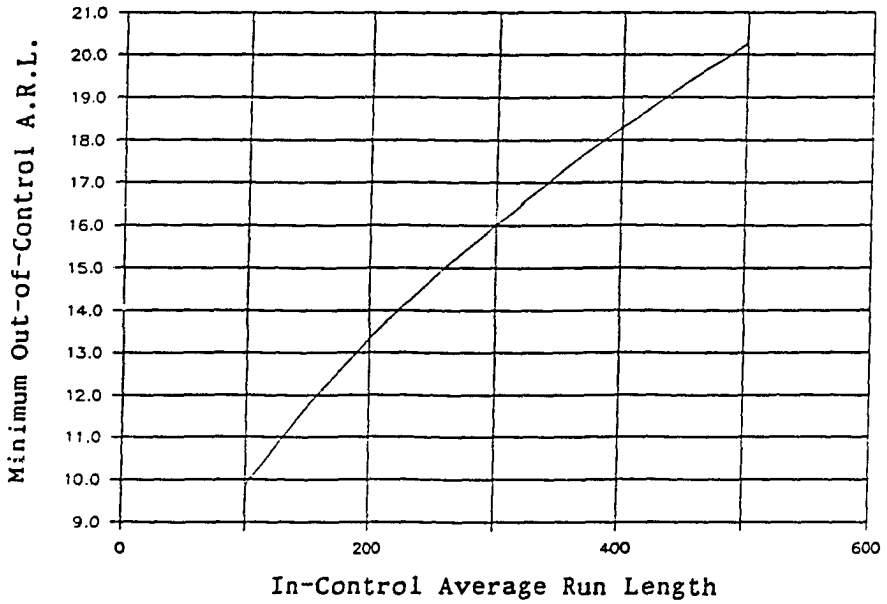



Figure 22. Optimal out-of-control A.R.L. for the { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes

Table 17 shows the A.R.L. curves, as functions of the shift in the process mean (in standard units), corresponding to the optimal control limits for in-control average run lengths of 100, 200, ..., 500. These results were generated using the program P342311B.BAS listed in Appendix E.

Table 17. Optimal average run lengths for the { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } family of control schemes. Values of control limits and process mean are given in standard units

| Process Mean | y = 1.20<br>x = 1.99<br>w = 3.34 | 1.33<br>2.15<br>3.70 | 1.40<br>2.25<br>3.92 | 1.45<br>2.32<br>4.07 | 1.49<br>2.37<br>4.19 |
|---|---|---|---|---|---|
| $\mu$ | Average Run Length | | | | |
| 0.00 | 100.0 | 200.0 | 300.0 | 400.0 | 500.0 |
| 0.05 | 98.0 | 195.1 | 291.7 | 388.1 | 484.4 |
| 0.10 | 92.6 | 181.5 | 269.2 | 356.2 | 442.6 |
| 0.15 | 84.7 | 162.5 | 238.1 | 312.4 | 385.9 |
| 0.20 | 75.6 | 141.4 | 204.3 | 265.5 | 325.6 |
| 0.25 | 66.3 | 120.8 | 171.9 | 221.2 | 269.3 |
| 0.30 | 57.6 | 102.1 | 143.2 | 182.5 | 220.6 |
| 0.35 | 49.7 | 85.9 | 118.8 | 150.0 | 180.0 |
| 0.40 | 42.9 | 72.2 | 98.6 | 123.3 | 147.0 |
| 0.45 | 37.0 | 60.9 | 82.1 | 101.7 | 120.4 |
| 0.50 | 32.0 | 51.5 | 68.6 | 84.3 | 99.2 |
| 0.60 | 24.3 | 37.5 | 48.8 | 59.0 | 68.5 |
| 0.70 | 18.8 | 28.0 | 35.5 | 42.3 | 48.6 |
| 0.80 | 14.8 | 21.3 | 26.6 | 31.2 | 35.4 |
| 0.90 | 12.0 | 16.7 | 20.4 | 23.6 | 26.5 |
| 1.00 | 9.9 | 13.3 | 16.0 | 18.3 | 20.3 |
| 1.25 | 6.6 | 8.3 | 9.6 | 10.7 | 11.6 |
| 1.50 | 4.8 | 5.8 | 6.5 | 7.0 | 7.5 |
| 1.75 | 3.7 | 4.4 | 4.8 | 5.1 | 5.4 |
| 2.00 | 3.1 | 3.6 | 3.8 | 4.0 | 4.2 |
| 2.25 | 2.6 | 3.0 | 3.2 | 3.3 | 3.4 |
| 2.50 | 2.3 | 2.6 | 2.8 | 2.9 | 2.9 |
| 2.75 | 2.0 | 2.3 | 2.5 | 2.5 | 2.6 |
| 3.00 | 1.8 | 2.1 | 2.2 | 2.3 | 2.3 |

Table 18 shows the ratios of the optimal A.R.L.s corresponding to the control scheme { R2[2,3,x,∞], R3[3,4,y,∞] } to those corresponding to the control scheme { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] } as functions of the shift in the process mean. These ratios show that very little improvement in the A.R.L. is attained by the addition of the control rule R1[1,1,w,∞] to the control scheme { R2[2,3,x,∞], R3[3,4,y,∞] } if the in-control A.R.L. is large (greater than 300) or the shift in the process mean is small or moderate (less than 2 standard units). Consequently, the control rule R1[1,1,w,∞] should be used only if economical considerations dictate that small or moderate shifts in the process mean are unimportant and that a relatively small in-control A.R.L. should be used.

Table 18. Ratios of average run lengths: optimal control scheme { R2[2,3,x,∞], R3[3,4,y,∞] } to optimal control scheme { R1[1,1,w,∞], R2[2,3,x,∞], R3[3,4,y,∞] }

| Process Mean | In - Control Average Run Length 100 | 200 | 300 | 400 | 500 |
|---|---|---|---|---|---|
| $\mu$ | R A T I O S | | | | |
| 0.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.20 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.40 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.60 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.80 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 1.00 | 1.01 | 1.00 | 1.01 | 1.00 | 1.00 |
| 1.25 | 1.02 | 1.01 | 1.01 | 1.00 | 1.00 |
| 1.50 | 1.05 | 1.01 | 1.01 | 1.01 | 1.01 |
| 1.75 | 1.04 | 1.02 | 1.01 | 1.01 | 1.01 |
| 2.00 | 1.07 | 1.04 | 1.02 | 1.02 | 1.02 |
| 2.25 | 1.11 | 1.06 | 1.05 | 1.05 | 1.05 |
| 2.50 | 1.14 | 1.07 | 1.07 | 1.05 | 1.05 |
| 2.75 | 1.19 | 1.12 | 1.09 | 1.07 | 1.07 |
| 3.00 | 1.28 | 1.15 | 1.12 | 1.10 | 1.09 |

## E. Comparison of Two Control Schemes

This section is devoted to the comparison of an optimal control scheme with the control scheme recommended by the Statistical Quality Control Handbook (Western Electric Company, 1956) and the evaluation of their performances under different out-of-control situations.

The control scheme suggested by Western Electric is used as a base for comparison because it is one of the most widely used and recommended control schemes. This control scheme declares the process to be out of control if a single sample mean falls above the $3\sigma$ limit or below the $-3\sigma$ limit, if two out of three successive sample means fall above the $2\sigma$ limit or below the $-2\sigma$ limit, if four out of five successive sample means fall above the $1\sigma$ limit or below the $-1\sigma$ limit, or if eight successive sample means fall above or below the target mean. The average run length properties of this control scheme have been studied and reported by Champ (1986). He reports an in-control A.R.L. of 91.75 for this scheme.

The Western Electric control scheme is compared to the member of the family of control schemes { R1[1,1,w,$\infty$], R2[2,3,x,$\infty$], R3[3,4,y,$\infty$] } with the same in-control A.R.L. and that minimizes the out-of-control A.R.L. for a shift in the process mean of one standard deviation; this control scheme is

{ R1[1,1,3.216,$\infty$], R2[2,3,1.962,$\infty$], R3[3,4,1.181,$\infty$] }

Table 19 shows the A.R.L. as a function of the shift in the process mean for these two control schemes. Notice that, although one of the

schemes requires only the last four samples to decide whether the process is under control or not and the other scheme requires eight samples to make the same decision, the difference between the A.R.L. of these two control schemes is very small. The Western Electric control scheme, Sa, is slightly superior to our optimized control scheme, S*, for small shifts in the process mean; however, the difference practically vanishes for moderate and large shifts in the process mean.

Table 19. Average run length as a function of the shift in the process mean for the control schemes Sa = { R1[1,1,3,∞], R2[2,3,2,∞], R3[4,5,1,∞], R4[8,8,0,∞] } and S* = { R1[1,1,3.216,∞], R2[2,3, 1.962,∞], R3[3,4,1.181,∞] }

| Mean Shift (in std. units) | Average Run Length Sa | S* |
|---|---|---|
| 0.00 | 91.7 | 91.7 |
| 0.05 | 90.3 | 90.3 |
| 0.10 | 84.3 | 84.5 |
| 0.15 | 75.9 | 77.5 |
| 0.20 | 66.8 | 69.4 |
| 0.25 | 57.9 | 61.2 |
| 0.30 | 49.8 | 53.3 |
| 0.35 | 42.7 | 46.2 |
| 0.40 | 36.6 | 40.0 |
| 0.45 | 31.5 | 34.6 |
| 0.50 | 27.3 | 30.0 |
| 0.60 | 20.9 | 22.9 |
| 0.70 | 16.4 | 17.8 |
| 0.80 | 13.2 | 14.2 |
| 0.90 | 10.9 | 11.5 |
| 1.00 | 9.2 | 9.5 |
| 1.25 | 6.5 | 6.3 |
| 1.50 | 4.9 | 4.7 |
| 1.75 | 3.8 | 3.7 |
| 2.00 | 3.1 | 3.0 |
| 2.25 | 2.6 | 2.6 |
| 2.50 | 2.2 | 2.2 |
| 2.75 | 1.9 | 2.0 |
| 3.00 | 1.7 | 1.8 |

The other question to be addressed in this section is: "How do these schemes compare to each other when the out-of-control situation does not correspond to a simple shift in the process mean?" To answer this question different alternative situations are considered: The process changes in such way that the continuous random variables X[i], i = 1,2,3,... that are being monitored and that measure the quality of the production process at time i are no longer normally distributed, but

1. they are independent and identically distributed according to a double exponential distribution with probability density function (p.d.f.)

$$f(x) = (\lambda/2)\ \mathrm{Exp}(\ -\lambda\ |\ x - \mu\ |\ ), \quad -\infty < x < +\infty,$$

and cumulative distribution function (c.d.f)

$$F(x) = \begin{cases} 0.50\ \mathrm{Exp}(\ \lambda(x-\mu)\ ), & x \le \mu \\ 1 - 0.50\ \mathrm{Exp}(-\lambda(x-\mu)), & x > \mu \end{cases}$$

2. they are independent and identically distributed according to a Cauchy distribution with p.d.f.

$$f(x) = \{\ \pi[\ 1 + ((x - \mu)/\sigma\ )^2\ ]\ \}^{-1}, \quad -\infty < x < +\infty,$$

and c.d.f.

$$F(x) = \frac{1}{\pi}\ \mathrm{Arctan}\ \frac{x - \mu}{\sigma} + \frac{1}{2}, \quad -\infty < x < +\infty.$$

3. they are the result of a mixed autoregressive-moving average process of first order, that is

$$X[n] = \epsilon[n] + \beta\ \epsilon[n-1] - \alpha\ X[n-1], \quad n = 1,2,3,...$$

where $\alpha$ and $\beta$ are constants and the $\epsilon[i]$'s are assumed to be independent and identically and normally distributed with mean zero and standard deviation $\sigma$.

The average run lengths for these out-of-control situations are estimated using Monte Carlo simulation. To carry out the simulation it is necessary to select particular values for the parameters of each of the three random processes described above. These values are selected such that the departure from the standard normal situation is not so obvious that any reasonable control scheme would detect the departure in a very few samples. For the double exponential case, two sets of parameter values are used: $\mu = 1$, $\lambda = \sqrt{2}$, and $\mu = 2$ and $\lambda = \sqrt{2}$ (in both cases the value of $\lambda$ is selected such that the variance is equal to 1). For the Cauchy distribution also two sets of parameter values are used: $\mu = 0$, $\sigma = 1$ (standard Cauchy) and $\mu = 0.50$ , $\sigma = 0.5011$ (this value for the scale parameter was selected because it minimizes the maximum absolute deviation between the standard normal c.d.f and a Cauchy c.d.f. with $\mu = 0$).

The mixed autoregressive-moving average (A.R.M.A.) process is used as an alternative model because it allows us to model situations in which there is a correlation between the random variables X[n] and X[n-1], n = 2,3,4,.... It can be shown that E( X[n] ) = 0 and that if the parameter $\alpha$ in the equation

$$X[n] = \epsilon[n] + \beta\ e[n-1] - \alpha\ X[n-1], \quad n = 1,2,3,...$$

is selected such that $-1 < \alpha < 1$ then the A.R.M.A. model is stable and hence the variance of X[n], Var( X[n] ), is finite and constant; moreover, the autocovariance, Cov( X[n], X[n-1] ), is independent of time n and the process is stationary, in other words, the joint distribution of any sequence of observations is the same no matter where in time the sequence is started (E. A. Robinson and M. T. Silvia, 1979). Given the values of $\alpha$, $\beta$ and $\sigma$, the variance and the autocovariance can be computed as

$$\text{Var}( X[n] ) = \frac{\sigma^2 (1 + \beta^2) - 2 \alpha \beta \sigma^2}{(1 + \alpha)(1 - \alpha)}$$

and

$$\text{Cov}( X[n], X[n-1] ) = \frac{\beta \sigma^2 (1 + \alpha^2) - \alpha \sigma^2 ( 1 + \beta^2 )}{(1 + \alpha)(1 - \alpha)}$$

In this study, the values of these parameters are selected such that Var( X[n] ) = 1. Three different sets of values for $\alpha$, $\beta$, and $\sigma$ are used in the simulation of the A.R.M.A. process; the first set, $\sigma = 0.316$, $\alpha = -0.938$, and $\beta = 0.100$, gives a high correlation between X[n] and X[n-1], 0.95, the second set of values, $\alpha = -0.515$, $\beta = 1.20$, and $\sigma = 0.447$, gives a correlation between X[n] and X[n-1] of 0.75, and the third set, $\alpha = -0.024$, $\beta = 1.20$, and $\sigma = 0.632$, gives a low correlation between X[n] and X[n-1], 0.50.

To carry out the simulations to estimate the average run lengths of the control schemes Sa = { R1[1,1,3,$\infty$], R2[2,3,2,$\infty$], R3[4,5,1,$\infty$], R4[8,8,0,$\infty$] } and S* = { R1[1,1,3.216,$\infty$], R2[2,3,1.962,$\infty$], R3[3,4,1.181,$\infty$] } under the different out-of-control situations

described above, six computer programs are used: programs DBLEXP1.BAS and DBLEXP.BAS simulate the control schemes, Sa and S*, respectively, when the process distribution has changed to a double exponential distribution, programs CAUCHY1.BAS and CAUCHY2.BAS simulate the control schemes, Sa and S*, respectively, when the process distribution has changed to a Cauchy distribution, and programs ARMA1.BAS and ARMA2.BAS simulate the control schemes, Sa and S*, respectively, when the process behaves according to an A.R.M.A. process of first order. All these programs share a common set of subroutines (SIM.INC) designed to generate random numbers according to different probability distributions, to check the different control rules after each sample is generated, and to collect and compute the basic statistics necessary for the comparison of the control schemes. Program listings are given in Appendix F.

A summary of the results produced by these programs is presented in Tables 20, 21, and 22. There is little difference in the average run

Table 20. Control-scheme comparisons: Double exponential distribution

| Distribution | Double exponential | | | |
|---|---|---|---|---|
| Parameters | $\mu$ = 1.000 $\lambda$ = 1.4142 | | $\mu$ = 2.000 $\lambda$ = 1.4142 | |
| Cntr.Scheme | Sa | S* | Sa | S* |
| No. of runs | 1500 | 1500 | 1500 | 1500 |
| A.R.L. | 8.795 | 9.279 | 3.183 | 3.157 |
| Std. dev. | 5.527 | 7.628 | 1.509 | 1.287 |
| 95% C.I. | (8.52, 9.07) | (8.89, 9.67) | (3.11, 3.26) | (3.09, 3.22) |

lengths of the control schemes when the distribution of the random variable being controlled changes to a double exponential distribution (with variance 1); this difference seems to become even smaller as the mean of the distribution, $\mu$, increases.

Table 21. Control-scheme comparisons: Cauchy distribution

| Distribution | Cauchy | | | |
|---|---|---|---|---|
| Parameters | $\mu = 0.000$ $\sigma = 1.0000$ | | $\mu = 0.500$ $\sigma = 0.5011$ | |
| Cntr.Scheme | Sa | S* | Sa | S* |
| No. of runs | 1500 | 1500 | 1500 | 1500 |
| A.R.L. | 4.711 | 4.812 | 7.698 | 8.764 |
| Std. dev. | 3.983 | 4.293 | 6.218 | 8.865 |
| 95% C.I. | (4.51, 4.91) | (4.60, 5.03) | (7.38, 8.01) | (8.32, 9.21) |

The average run lengths of the control schemes Sa and S* are practically equal, about 4.7, (see Table 21) when the observations come from a standard Cauchy distribution; however, for the non-standard Cauchy distribution, the more complex control scheme, Sa, detects the out-of-control situation faster.

The more complex control scheme, Sa, is clearly superior to the control scheme S* in detecting observations that are correlated (as in the A.R.M.A. process). Sa detects the departure from normality in about 20% fewer samples than S*.

Finally, we do not want to over emphasize the numerical results of these simulations; however, we want to stress the fact that control

schemes that perform well in certain out-of-control situations might not perform as well in others. Consequently, we recommend the use of the "optimized" control schemes only in those cases in which the sample size is large enough so we can count on the normality assumption.

Table 22. Control-scheme comparisons: A.R.M.A. process

| Mixed autoregressive-moving average process | | | | | | |
|---|---|---|---|---|---|---|
| Parameters | $a$ = -0.938<br>$\beta$ = 0.100<br>$\sigma$ = 0.316 | | $a$ = -0.515<br>$\beta$ = 1.200<br>$\sigma$ = 0.477 | | $a$ = -0.024<br>$\beta$ = 1.200<br>$\sigma$ = 0.632 | |
| Cntr.Scheme | Sa | S* | Sa | S* | Sa | S* |
| No. of runs | 1500 | 1500 | 1500 | 1500 | 1500 | 1500 |
| A.R.L. | 12.381 | 15.580 | 18.192 | 24.238 | 32.684 | 38.710 |
| Std. dev. | 6.348 | 13.413 | 13.830 | 21.950 | 29.045 | 34.190 |
| 95% conf. interval | (12.0, 12.7) | (14.9, 16.3) | (17.5, 18.9) | (23.1, 25.3) | (31.2, 34.2) | (37.0, 40.4) |

# V. CONCLUSIONS AND RECOMMENDATIONS

In this study, a new method for designing control schemes is presented and discussed. This method is based entirely on the average run length properties of the control schemes and it seeks the minimization of the out-of-control average run length (by the appropriate selection of the control limits) for a given in-control average run length. This method can be used in conjunction with economical-design methods for control charts or as an alternative to these methods when estimates of cost parameters are not available.

This research shows that a Markov chain approach can be used to obtain exact run length properties of control schemes and, combined with appropriate optimization techniques, it can be used to find optimal control limits, as well. Comparisons of control schemes with optimal control limits to other control schemes having the same on-target A.R.L. indicate that up to a 50% reduction in the number of samples required to detect a moderate shift in the process mean is possible; consequently, the used of the "optimized" control schemes is strongly recommended.

In this research four different families of control schemes were carefully examined:

$$F1 = \{ R1[1,1, w,\infty], R2[2,3, x,\infty] \},$$

$$F2 = \{ R1[1,1, w,\infty], R3[3,4, y,\infty] \},$$

$$F3 = \{ R2[2,3, x,\infty], R3[3,4, y,\infty] \}, \text{ and}$$

$$F4 = \{ R1[1,1, w,\infty], R2[2,3, x,\infty], R3[3,4, y,\infty] \}.$$

Let Si*, i = 1,2,3,4, denote the best control scheme of the family Fi, i = 1,2,3,4, for a given in-control A.R.L. This study shows that control scheme S2* detects small shifts in the process mean faster than the corresponding scheme S1* with the same on-target A.R.L.; however, scheme S1* detects large shifts in the mean faster than S2* if the corresponding in-control A.R.L. is relatively small (less than 200).

The contrast of schemes S2* and S3* reveals that, for small or moderate shifts in the process mean, the control scheme S3* is only slightly better than the corresponding control scheme S2* with the same in-control A.R.L.; consequently, we recommend using the more complex scheme, S3*, only if one is specially interested in decreasing the number of samples necessary to detect a large shift in the mean.

The comparison of control schemes S3* and S4* shows that S4* is significantly better than the corresponding control scheme S3* (with the same on-target A.R.L.) only for large shifts in the process mean and small and moderate in-control average run lengths.

These results suggest that only a small reduction in the out-of-control A.R.L. is possible by the inclusion of more control rules into a given control scheme. This is confirmed by the fact that the control schemes { R1[1,1, 3,∞], R2[2,3, 2,∞], R3[4,5, 1,∞], R4[1,1, 3,∞] } and { R1[1,1, 3.216,∞], R2[2,3, 1.962,∞], R3[3,4, 1.181,∞] }, for all practical purposes, have the same A.R.L. curve. However, simulation results suggest that complex control schemes, as the one recommended by the Western Electric Company, require, in average, fewer

samples to detect correlations in the observations and changes in the distribution of the random variable being monitored.

Finally, we would like to point out that the methods of analysis and design of control charts presented in this study are not limited to the X-bar chart or to the rules that were used in the study. These methods can be applied to other control charts such as the R chart, np chart, c chart and $S^2$ chart; this is an area that definitely deserves further study.

## VI. REFERENCES

Aroian, L. A. and Levene, H. "The Effectiveness of Quality Control Charts." Journal of the American Statistical Association, 44, No. 252 (1950): 520-529.

Baker, K. R. "Two Process Models in the Economic Design of an X-bar Chart." AIIE Transactions, 3, No. 4 (1971): 257-263.

Barish N. N. and Hauser N. "Economic Design for Control Decisions." Journal of Industrial Engineering, 14, No. 3 (1963): 125-134.

Bather, J. A. "Control Charts and the Minimization of Costs." Journal of the Royal Statistical Society, B, 25, No. 1 (1963): 49-80.

Burr, I. W. Engineering Statistics and Quality Control. New York: McGraw-Hill, 1953.

Champ, C. W. "Exact Results for Shewhart Control Charts with Supplementary Runs Rules." Ph.D. Thesis. University of Southwestern Louisiana, Lafayette, Louisiana, 1986.

Chiu, W. K. "Comments on the Economic Design of X-bar Charts." Journal of the American Statistical Association, 68, No. 344 (1973): 919-921.

Chiu, W. K. and Cheung, K. C. "An Economic Study of X-bar charts with Warning Limits." Journal of Quality Technology, 9, No. 4 (1977): 166-171.

Chiu, W. K. and Wetherill, G. B. "A Simplified Scheme for the Economic Design of X-bar Charts." Journal of Quality Technology, 6, No. 2 (1974): 63-69.

Cody, W. J. "Rational Chebyshev Approximation for the Error Function", Mathematical Computing, 23 (1969): 631-638.

Cowden, D. J. Statistical Methods in Quality Control. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1957.

Duncan, A. J. "The Economic Design of X-bar Charts Used to Maintain Current Control of a Process." Journal of the American Statistical Association, 51, No. 274 (1956): 228-242.

Duncan, A. J. "The Economic Design of X-bar Charts When There is a Multiplicity of Assignable Causes." Journal of the American Statistical Association, 66, No. 333 (1971): 107-121.

Duncan, A. J. Quality Control and Industrial Statistics. Homewood, Illinois: Richard D. Irwin, Inc. 1974.

Feigenbaum, A. V. Total Quality Control. New York: McGraw-Hill, 1961.

Gibra, I. N. "Economically Optimal Determination of the Parameters of X-bar Control Chart." Management Science, 17, No. 9 (1971): 635-646.

Gibra, I. N. "Optimal Control of Processes Subject to Linear Trends." The Journal of Industrial Engineering, 28, No. 1 (1967): 35-41.

Girshick, M. A. and Rubin, H. "A Bayes Approach to a Quality Control Model." Annals of Mathematical Statistics, 23, No. 23 (1952): 114-125.

Goel, A. L., Jain, S. C., and Wu, S. M. "An Algorithm for the Determination of the Economic Design of X-bar Charts Based on Duncan's Model." Journal of the American Statistical Association, 62, No. 321 (1968): 304-320.

Gordon, G. R. and Weindling, J. I. "A Cost Model for Economic Design of Warning Limit Control Charts Schemes." AIIE Transactions, 7, No. 3 (1975): 319-329.

Grant, E. L., and Leavenworth, R. S. Statistical Quality Control. New York: McGraw-Hill, 1988.

Iosifescu, M. Finite Markov Processes and Their Applications. New York: John Wiley and Sons, 1980.

Ishikawa, K. Guide to Quality Control. Tokyo: Asian Productivity Organization, 1976.

Johnson, L. W. and Dean Riess, R. Numerical Analysis. Philippines: Addison-Wesley Publishing Company, Inc., 1982.

Juran, J. M., Gryna, F. M., Jr., and Bingham, R. S., Jr. Quality Control Handbook. New York: McGraw-Hill, 1974.

Kennedy, W. J. and Gentle, J. E. Statistical Computing. New York: Marcel Dekker, Inc., 1980.

Knappenberger, H. A. and Grandage, A. H. E. "Minimum cost Quality Control Tests." AIIE transactions, 1 No. 1 (1969): 24-32.

Lorenzen, T. J. and Vance, L. C. "The Economic Design of Control Charts: A Unified Approach." Technometrics, 28, No. 1 (1986): 3-10.

Montgomery, D. C. "Economic Design of an X-bar Control Chart." Journal of Quality Technology, 14, No.1 (1982): 40-43.

Moore, P. G. "Some Properties of Runs in Quality Control Procedures." Biometrica, 45 (1958): 89-95.

Page, E. S. "Control Charts with Warning Lines." Biometrica, Vol 42 (1955): 100-115.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. Numerical Recipes. The Art of Scientific Computing. New York: Cambridge University Press, 1986.

Robinson, E. A. and Silvia, M. T. Digital Foundations of Time Series Analysis.1. The Box-Jenkins Approach. San Francisco: Holden-Day Inc, 1979.

Ross, S. M. "Quality Control Under Markovian Deterioration." Management Science, 17, No. 5 (1971): 587-596.

Savage, I. R. "Surveillance Problems." Naval Research Logistic Quarterly, 9, No. 2 (1962): 187-209.

Taylor, H. M. "Markovian Sequential Replacement Processes." Annals of Mathematical Statistics, 36 (1965): 187-209.

Weiler, G. H. "The Use of Runs to Control the Mean in Quality Control." Journal of the American Statistical Association, 48, No. 6 (1953): 816-825.

Weiler, G. H. "A New Type of Control Chart Limits for Means, Ranges, and Sequential Runs." Journal of the American Statistical Association, 49, No. 266 (1954): 298-314.

Weindling, J. I. Statistical Properties of a General Class of Control Charts Treated as a Markov Process. Ph.D. Dissertation, Columbia University, New York, 1967.

Weindling, J. I., Littauer, S. B., and Tiago de Oliveira, J. "Mean Action Time of the X-bar Control Chart with Warning Limits." Journal of Quality Technology, 2, No. 2, (1970): 79-85.

Western Electric Company. Statistical Quality Control Handbook. New York: Western Electric Company Inc., 1956.

Wetherill, G. B. Sampling Inspection and Quality Control. London: Methuen and Company, Ltd., 1977.

Wheeler, D. J. "Detecting a Shift in Process Average: Tables of the Power Function for X-bar Charts." Journal of Quality Technology, 15, No. 4 (1983): 155-169.

White C. C. "A Markov Quality Control Process Subject to Partial Observation." Management Science, 23, No. 8 (1974): 843-852.

## VII. ACKNOWLEDGEMENTS

The research recorded in this dissertation was done through a close working relationship with my major professors, Dr. Herbert T. David and Dr. Howard D. Meeks. I would like to express my sincere gratitude for their teachings, strong support and encouragement. Special thanks to Dr. H. T. David for his continuous guidance and valuable suggestions, to Dr. Stephen B. Vardeman for introducing me to the area of Quality Control and arising my interest for the study of control charts, and to Dr. Richard A. Groeneveld for introducing me to the subject of Markov processes.

Thanks to each member of my committee: Dr. John C. Even, Dr. Richard A. Groeneveld, and Dr. Vincent A. Sposito for their teachings, advices, and support during my years at Iowa State University.

Finally, thanks to my mother and my son for their endless understanding and encouragement.

# VIII. APPENDIX A: PROGRAMS P2311A.BAS AND P2311B.BAS

```
'   Noel Artiles.   September 1988.
'   File: P2311A.BAS
'
'   AVERAGE-RUN-LENGTH CALCULATIONS
'
'   RULES CONSIDERED IN THIS PROGRAM:
'
'          Stop if 2/3 points fall in (-oo, -b), or
'          stop if 2/3 points fall in ( b, +oo), or
'          stop if 1/1 point  fall in (-oo,-x) U (x, +oo).
'
'   This program computes values for b and x that will result in a
'   given in-control ARL, ARL0, and that minimize the out-of-control
'   average run length (process mean = 1).
'
'=====================================================================
CLS
DEFINT i-n                                      ' Define integer variables.
DEFDBL a-h,o-z                                  ' Define double prec. vars.
DEF FNfmax(x,y) = (x+y+abs(x-y))/2              ' Define Max{} function
DEF FNfmin(x,y) = (x+y-abs(x-y))/2              ' Definr Min{} function
'
n = 7:                  ' Size of matrix (I-Q)
n.max=50:               ' Maximum number of iterations for secant method
x.tol=1.0D-10           ' Tolerance in x.limit for secant method
CALL init.normal        ' Initialize constants for Normal.prob. routine
DIM a(0:7, 0:7)         ' Matrix a contains (I-Q) matrix
DIM clim(3), arlout(3)
'
format1$ = "B = #.#####   X = #.#####  ARLzero = ####.####"
format2$ = format1$+"  ARLone = ####.####"
'
blimit = 1.50
OPEN "P2311X1.PRN" FOR OUTPUT AS #1
'
'                                   Optimize for ARLo = 100, 125, ..., 500
'
FOR iar1 = 100 TO 500 STEP 25
   ar10  = iar1
   y.tol = ar10*(1.0E-8)
   pmean = 1.0
   PRINT USING "     ar10 = ###.##"; ar10
   '
   '                            Define an initial set of values for the
   '                            inner limit (blimit)
   FOR k = 1 TO 3
     clim(k)   = blimit
     arlout(k) = FNflagrange (blimit, pmean, ar10 )
     blimit    = blimit + 0.10
   NEXT k
```

```
 icounter = 1
 vlen = ABS( clim(3)-clim(1) )
 WHILE (icounter < 500  AND vlen > .0005 )
   CALL interp ( clim(), arlout(), flag, blimit)
   ysmall = arlout(1):  ismall = 1
   FOR i = 2 TO 3
      IF ysmall > arlout(i) THEN
         ysmall = arlout(i)
         ismall = i
      END IF
   NEXT i
   vlen = ABS( clim(3)-clim(1) )
   icounter = icounter + 1
   '
   IF flag = 1 THEN  '            If flag=1 the function is not convex
   '                              in current interval.
      PRINT
      PRINT "Warning: Function is not convex in current interval."
      PRINT "         Rerun the program with different starting points."
      PRINT
      STOP
                '
   ELSE         ' If function is convex in current interval,
                ' replace worst control limit by the approximation
                ' computed by subroutine "interp".
                '
      arl1     = FNflagrange (blimit, pmean, arl0 )
      arlworst = arlout(1)
      iworst   = 1
      FOR i = 2 TO 3
         IF arlworst < arlout(i) THEN
            arlworst = arlout(i)
            iworst   = i
         END IF
      NEXT i
      clim(iworst)   = blimit
      arlout(iworst) = arl1
   END IF
WEND
'
'                          Best inner control limit has been found.
'                          Compute corresponding outer control limit.
'
bestarl = arlout(1):  ibest = 1
FOR i = 2 TO 3
   IF bestarl > arlout(i) THEN
      bestarl = arlout(i)
      ibest   = i
   END IF
NEXT i
```

```
    blimit = clim(ibest)
    xlow   = FNfmax(1.5, blimit)
    x1     = xlow + ABS(4-xlow)/4
    X2     = 4.00
    CALL secant (x1, x2, n.max, x.tol, y.tol, ar10, ar1, xlimit )
    PRINT #1, USING format2$; blimit, xlimit, ar1, arlout(ibest)
    PRINT     USING format2$; blimit, xlimit, ar1, arlout(ibest)
    PRINT
NEXT iar1
CLOSE #1
PRINT "Done !!"
END
```

```
SUB interp ( x(1), y(1), flag, xvalue )
   '
   ' This function uses quadratic interpolation to approximate the minimum
   ' of a function. If the set of points (x1,y1), (x2,y2), and (x3, y3)
   ' does not fall on a convex function the value of flag is set to 1,
   ' otherwise it is set to zero.
   '
   LOCAL d32, d13, d21, rnum, rden, xsmall, ismall, yvalue, i, j
   '
   FOR i = 1 TO 3
      xsmall = x(i)
      ismall = i
      FOR j = i TO 3
         IF xsmall > x(j) THEN
            xsmall = x(j)
            ismall = j
         END IF
      NEXT j
      SWAP x(i),x(ismall)
      SWAP y(i),y(ismall)
   NEXT i
   d32 = x(3)-x(2)
   d13 = x(1)-x(3)
   d21 = x(2)-x(1)
   yvalue = y(1) + d21*( y(1)-y(3) )/d13
   IF yvalue > y(2) THEN flag = 0 ELSE flag = 1
   rnum = y(1)*d32*(x(3)+x(2))+y(2)*d13*(x(1)+x(3))+y(3)*d21*(x(2)+x(1))
   rden = y(1)*d32 + y(2)*d13 + y(3)*d21
   xvalue = rnum/rden/2
   '
END SUB 'interp
'
```

```
DEF FNflagrange (blimit, pmean, arl0)
  '
  ' Given the value of the inner control limit, blimit, this function
  ' computes the value of the outer control limit that will result in-
  ' control ARL of ARLo and computes and returns the value of ARL at
  ' pmean. If a value for the outer limit cannot be found the function
  ' adds a penalty to the value of the ARL at pmean.
  '
  SHARED n.max, x.tol, y.tol
  LOCAL xlow, xl, x2, arlnull, arl
  xlow = FNfmax(1.5, blimit)
  xl = xlow + ABS(3-XLOW)/4
  X2 = 3.00
  CALL secant (xl, x2, n.max, x.tol, y.tol, arl0, arlnull, xlimit )
  '
  CALL init.matrix (blimit, xlimit, pmean)      ' Initialize (I-Q)
  CALL arl( a(), n, arl)                        ' Compute ARL for pmean
  IF ABS(arlnull-arl0)< 10*y.tol THEN
     FN flagrange = arl
  ELSE
     FN flagrange = arl + ABS(arlnull-arl0)
  END IF
END DEF 'flagrange
'
```

```
SUB secant ( x1, x2, n.max,  x.tol, y.tol, ar10, ar1, xlimit )
    '
    ' Secant Method Routine
    '
    ' Input:   starting points:                x1 and x2
    '          maximum number of iterations :  n.max
    '          target average run length:      ar10 (for pmean=0)
    '          tolerances:                     x.tol, y.tol
    '          routine to initialize (I-Q):    init.matrix
    '          routine to compute ARL:         ar1 ( , , )
    '
    ' Output:  xlimit, and corresponding ARL, ar1.
    '
   SHARED A(), n, blimit, format3$
   LOCAL  f1, f2, f3, x3, k!, psec, qsec, zero
   zero = 1e-15
   '
   CALL init.matrix (blimit, x1, 0)     ' Compute ARL for x1
   CALL ar1( A(), n, ar1)
   f1=ar1-ar10
   CALL init.matrix (blimit, x2, 0)     ' Compute ARL for x2
   CALL ar1( A(), n, ar1)
   f2=ar1-ar10
   IF ABS(f2)>ABS(f1) THEN
      SWAP x1,x2
      SWAP f1,f2
   END IF
   '
   FOR k! = 1 TO n.max
      IF ABS(f2)<y.tol THEN
         xlimit=x2
         ar1=f2+ar10
         EXIT SUB
      END IF
      s = f2/f1
      psec = (x1-x2)*s
      qsec = 1-s:
   '
      IF ABS(qsec) > zero THEN
        x3 = x2 -psec/qsec
        IF x3>6.0                          THEN    x3=6.0
        IF x3<blimit                       THEN    x3=blimit
        IF ABS(x2-x3)<x.tol*ABS(x2)        THEN
           xlimit = x2
           ar1 = f2 +ar10
           EXIT SUB
        END IF
        CALL init.matrix (blimit, x3, 0)     ' Compute ARL for x3
        CALL ar1( A(), n, ar1)
        f3=ar1-ar10
```

```
        IF ABS(f3) > ABS(f2) THEN
           x1=x3:            f1=f3
        ELSE
           x1=x2:           f1=f2
           x2=x3:           f2=f3
        END IF
      END IF
   NEXT k!
   PRINT "No convergence after";k!;" iterations"
END SUB ' secant
```

```
'
SUB ar1 ( array(2), n, ar1.value )
      '
      ' This routine computes the average run length given the array (I-Q)
      ' Input:   square array "array( , )" containing (I-Q)
      '          array size, n
      ' Output:  average run length, ar1.value
      '          lower triangular array equivalent to "array(,)"
      '
      LOCAL zero, irow, index, big, nrow, jcol, fctr, frmt$
      zero  = 1E-10
      frmt$ = "Largest possible pivot element #.####### in row ###"
      '
      FOR irow = 1 TO N
         array(irow,0) = 1
      NEXT irow
      '
      FOR irow = N TO 1 STEP -1
         index = 1
         big   = ABS(array(index,irow))
         FOR nrow = 2 TO irow
            IF big   < ABS(array(nrow,irow)) THEN
               big   = ABS(array(nrow,irow))
               index = nrow
            END IF
         NEXT nrow
         IF ABS(big) <= zero      THEN
            PRINT
            PRINT "Error in SUB ar1 [...]: Pivot element is zero !!"
            PRINT USING frmt$; big, index
            STOP
         END IF
         FOR jcol = 0 TO irow
             SWAP array(index,jcol), array(irow,jcol)
         NEXT jcol
         FOR jcol = 0 TO irow-1
             array(irow,jcol) = array(irow,jcol)/array(irow,irow)
         NEXT jcol
         array(irow,irow) = 1
         FOR nrow = 1 TO irow-1
           fctr   = array(nrow,irow)
           IF ABS(fctr) > zero    THEN
             FOR jcol = 0 TO irow
               array(nrow,jcol) = array(nrow,jcol) - fctr*array(irow,jcol)
             NEXT jcol
           END IF
         NEXT nrow
      NEXT irow
      ar1.value = array(1,0)
END SUB 'ar1
```

```
'=============================================================================
'
' Note: File "NORMAL.INC" contains a subroutine to compute cumulative
'       standard normal probabilities.
'
$INCLUDE "NORMAL.INC"
'
'=============================================================================
'
SUB init.matrix (blimit, xlimit, pmean)
    '
    '   Initialize A = (I-Q) matrix
    '
    ' Input:  control limits:              xlimit, blimit
    '         process mean:                pmean
    '
    ' Output: A ( matrix [I-Q] )
    '
    SHARED A(), n
    LOCAL  i, j, nsize, z#, p1#, p2#, p3#, p4#, p5#, p6#
    LOCAL  pa, po, pb
'
    z# = ( xlimit-pmean ):    p1# = FNCDFnormal1#( z# )
    z# = ( blimit-pmean ):    p2# = FNCDFnormal1#( z# ) :      pa=p1#-p2#
    z# = (-blimit-pmean ):    p3# = FNCDFnormal1#( z# ) :      po=p2#-p3#
    z# = (-xlimit-pmean ):    p4# = FNCDFnormal1#( z# ) :      pb=p3#-p4#
    nsize = n
    FOR i = 1 TO nsize
       FOR j = 1 TO nsize:    a(i,j) = 0:                  NEXT j
    NEXT i
'
    a( 1,  1)=1-po:   a( 1,  2)=-pa:    a( 1,  3)=-pb:
    a( 2,  2)=1:      a( 2,  4)=-po:    a( 2,  6)=-pb:
    a( 3,  3)=1:      a( 3,  5)=-po:    a( 3,  7)=-pa:
    a( 4,  1)=-po:    a( 4,  3)=-pb:    a( 4,  4)=1
    a( 5,  1)=-po:    a( 5,  2)=-pa:    a( 5,  5)=1
    a( 6,  5)=-po:    a( 6,  6)=1
    a( 7,  4)=-po:    a( 7,  7)=1
    '
END SUB 'init.matrix
'
'=============================================================================
'   *** END OF FILE *** END OF FILE *** END OF FILE *** END OF FILE ***
'
```

```
'
'    Noel Artiles.   September 1988.
'    File: P2311B.BAS
'
'    AVERAGE-RUN-LENGTH CALCULATIONS
'
'    RULES CONSIDERED IN THIS PROGRAM:
'
'            Stop if 2/3 points fall in (-oo, -b), or
'            stop if 2/3 points fall in ( b, +oo), or
'            stop if 1/1 point  fall in (-oo,-x) U (x, +oo).
'
'    This program generate the ARL curves corresponding to the optimal
'    values of b and x. The in-control ARL considered are 100, 200, ...,
'    500 and the process mean varies from 0.00 to 2.00 by 0.10
'
'=====================================================================
'
CLS
DEFINT i-n                                      ' Define integer variables.
DEFDBL a-h,o-z                                  ' Define double prec. vars.
DEF FNfmax(x,y) = (x+y+abs(x-y))/2              ' Define Max{} function
DEF FNfmin(x,y) = (x+y-abs(x-y))/2              ' Definr Min{} function
'
n = 7:                    ' Size of matrix (I-Q)
n.max=50:                 ' Maximum number of iterations for secant method
x.tol=1.0D-10             ' Tolerance in x.limit for secant method
CALL init.normal          ' Initialize constants for Normal.prob. routine
DIM a(0:7, 0:7)           ' Matrix a contains (I-Q) matrix
DIM clim(3), arlout(3)
'
OPEN "P2311A1.PRN" FOR OUTPUT AS #1
FOR iarl = 100 TO 500 STEP 100
   arl0  = iarl
   y.tol = arl0*(1.0E-8)
   arl0log = LOG(arl0)
   blimit = 0.143377 + 0.380772*arl0log - 0.0133218*arl0log¢2
   xlimit = 10
   PRINT #1, USING "Control Limit = ##.####"; blimit
   FOR pmean = 0 to 2.00 step 0.05
      CALL init.matrix (blimit, xlimit, pmean)    ' Initialize (I-Q)
      CALL arl( a(), n, arl)                      ' Compute ARL for pmean
      PRINT #1, USING " ##.##   ####.###"; pmean, arl
      PRINT     USING " ##.##   ####.###"; pmean, arl
  NEXT pmean
  PRINT #1,
  PRINT
NEXT iarl
'
```

```
'
SUB arl ( array(2), n, arl.value )
      '
      ' This routine computes the average run length given the array (I-Q)
      ' Input:   square array "array( , )" containing (I-Q)
      '          array size, n
      ' Output:  average run length, arl.value
      '          lower triangular array equivalent to "array(,)"
      '
      LOCAL zero, irow, index, big, nrow, jcol, fctr, frmt$
      zero  = 1E-10
      frmt$ = "Largest possible pivot element #.####### in row ###"
      '
      FOR irow = 1 TO N
         array(irow,0) = 1
      NEXT irow
      '
      FOR irow = N TO 1 STEP -1
         index = 1
         big   = ABS(array(index,irow))
         FOR nrow = 2 TO irow
            IF big   < ABS(array(nrow,irow)) THEN
               big   = ABS(array(nrow,irow))
               index = nrow
            END IF
         NEXT nrow
         IF ABS(big) <= zero      THEN
            PRINT
            PRINT "Error in SUB arl [...]: Pivot element is zero !!"
            PRINT USING frmt$; big, index
            STOP
         END IF
         FOR jcol = 0 TO irow
             SWAP array(index,jcol), array(irow,jcol)
         NEXT jcol
         FOR jcol = 0 TO irow-1
             array(irow,jcol) = array(irow,jcol)/array(irow,irow)
         NEXT jcol
         array(irow,irow) = 1
         FOR nrow = 1 TO irow-1
           fctr   = array(nrow,irow)
           IF ABS(fctr) > zero    THEN
             FOR jcol = 0 TO irow
               array(nrow,jcol) = array(nrow,jcol) - fctr*array(irow,jcol)
             NEXT jcol
           END IF
         NEXT nrow
      NEXT irow
      arl.value = array(1,0)
END SUB 'arl
```

```
'=============================================================================
'
' Note: File "NORMAL.INC" contains a subroutine to compute cumulative
'       standard normal probabilities.
'
$INCLUDE "NORMAL.INC"
'
'=============================================================================
'
SUB init.matrix (blimit, xlimit, pmean)
   '
   '   Initialize A = (I-Q) matrix
   '
   ' Input:  control limits:              xlimit, blimit
   '         process mean:                pmean
   '
   ' Output: A ( matrix [I-Q] )
   '
   SHARED A(), n
   LOCAL  i, j, nsize, z#, p1#, p2#, p3#, p4#, p5#, p6#
   LOCAL  pa, po, pb
'
   z# = ( xlimit-pmean ):    p1# = FNCDFnormal#( z# )
   z# = ( blimit-pmean ):    p2# = FNCDFnormal#( z# ) :      pa=p1#-p2#
   z# = (-blimit-pmean ):    p3# = FNCDFnormal#( z# ) :      po=p2#-p3#
   z# = (-xlimit-pmean ):    p4# = FNCDFnormal#( z# ) :      pb=p3#-p4#
   nsize = n
   FOR i = 1 TO nsize
      FOR j = 1 TO nsize:    a(i,j) = 0:                  NEXT j
   NEXT i
'
   a( 1,  1)=1-po:   a( 1,  2)=-pa:    a( 1,  3)=-pb:
   a( 2,  2)=1:      a( 2,  4)=-po:    a( 2,  6)=-pb:
   a( 3,  3)=1:      a( 3,  5)=-po:    a( 3,  7)=-pa:
   a( 4,  1)=-po:    a( 4,  3)=-pb:    a( 4,  4)=1
   a( 5,  1)=-po:    a( 5,  2)=-pa:    a( 5,  5)=1
   a( 6,  5)=-po:    a( 6,  6)=1
   a( 7,  4)=-po:    a( 7,  7)=1
   '
END SUB 'init.matrix
'
'=============================================================================
'   *** END OF FILE *** END OF FILE *** END OF FILE *** END OF FILE ***
```

# IX. APPENDIX B: PROGRAMS P3411A.BAS AND P3411B.BAS

```
'
'   Noel Artiles.   September 1988.
'   File: P3411A.BAS
'
'   AVERAGE-RUN-LENGTH CALCULATIONS
'
'   RULES CONSIDERED IN THIS PROGRAM:
'
'           Stop if 3/4 points fall in (-oo, -b), or
'           stop if 3/4 points fall in ( b, +oo), or
'           stop if 1/1 point  fall in (-oo,-x) U (x, +oo).
'
'   This program computes values for b and x that will result in a
'   given in-control ARL, ARL0, and that minimize the out-of-control
'   average run length (process mean = 1).
'
'=================================================================
'
CLS
DEFINT i-n                                  ' Define integer variables.
DEFDBL a-h,o-z                              ' Define double prec. vars.
DEF FNfmax(x,y) = (x+y+abs(x-y))/2          ' Define Max{} function
DEF FNfmin(x,y) = (x+y-abs(x-y))/2          ' Definr Min{} function
'
n = 25                    ' Size of matrix (I-Q)
n.max=50:                 ' Maximum number of iterations for secant method
x.tol=1.0D-10             ' Tolerance in x.limit for secant method
CALL init.normal          ' Initialize constants for Normal.prob. routine
DIM a(0:25,0:25)          ' Matrix a contains (I-Q) matrix
DIM clim(3), arlout(3)
'
format1$ = "B = #.#####   X = #.#####  ARLzero = ####.####"
format2$ = format1$+"  ARLone = ####.####"
'
OPEN "P3411X1.PRN" FOR OUTPUT AS #1
'
'                              Optimize for ARLo = 100, 150, ..., 500
'
FOR iarl = 75 TO 525 STEP 25
   arl0  = iarl
   y.tol = arl0*(1.0E-8)
   pmean = 1.00
   PRINT USING "     arl0 = ###.##"; arl0
   '
   '                           Define an initial set of values for the
   '                           inner limit (blimit)
   blimit = 1.06 + .75*arl0/1000
```

```
 FOR k = 1 TO 3
   clim(k)   = blimit
   arlout(k) = FNflagrange (blimit, pmean, ar10 )
   blimit    = blimit + 0.075
 NEXT k
 icounter = 1
 vlen = ABS( clim(3)-clim(1) )
 WHILE (icounter < 500  AND vlen > .005 )
   CALL interp ( clim(), arlout(), flag, blimit)
   ysmall = arlout(1):  ismall = 1
   FOR i = 2 TO 3
      IF ysmall > arlout(i) THEN
         ysmall = arlout(i)
         ismall = i
      END IF
   NEXT i
   print using "###.######  "; ysmall, clim(ismall)
   vlen = ABS( clim(3)-clim(1) )
   icounter = icounter + 1
   '
   IF flag = 1 THEN  '             If flag=1 the function is not convex
   '                               in current interval.
      PRINT
      PRINT "Warning: Function is not convex in current interval."
      PRINT "         Rerun the program with different starting points."
      PRINT
      STOP
                     '
   ELSE              ' If function is convex in current interval,
                     ' replace worst control limit by the approximation
                     ' computed by subroutine "interp".
                     '
      ar11     = FNflagrange (blimit, pmean, ar10 )
      arlworst = arlout(1)
      iworst   = 1
      FOR i = 2 TO 3
         IF arlworst < arlout(i) THEN
            arlworst = arlout(i)
            iworst   = i
         END IF
      NEXT i
      clim(iworst)   = blimit
      arlout(iworst) = ar11
   END IF
 WEND
```

```
      '                          Best inner control limit has been found.
      '                          Compute corresponding outer control limit.
      '
      bestarl = arlout(1):  ibest = 1
      FOR i = 2 TO 3
         IF bestarl > arlout(i) THEN
            bestarl = arlout(i)
            ibest   = i
         END IF
      NEXT i
      blimit = clim(ibest)
      CALL limit.app (arl0, blimit, x1, x2)
      CALL secant (x1, x2, n.max, x.tol, y.tol, arl0, arl, xlimit )
      PRINT #1, USING format2$; blimit, xlimit, arl, arlout(ibest)
      PRINT     USING format2$; blimit, xlimit, arl, arlout(ibest)
      PRINT
   NEXT iarl
   CLOSE #1
   '
   PRINT "Done !!"
   END
   '
```

```
'
SUB interp ( x(1), y(1), flag, xvalue )
  '
  ' This function uses quadratic interpolation to approximate the minimum
  ' of a function. If the set of points (x1,y1), (x2,y2), and (x3, y3)
  ' does not fall on a convex function the value of flag is set to 1,
  ' otherwise it is set to zero.
  '
  LOCAL d32, d13, d21, rnum, rden, xsmall, ismall, yvalue, i, j
  '
  FOR i = 1 TO 3
     xsmall = x(i)
     ismall = i
     FOR j = i TO 3
        IF xsmall > x(j) THEN
           xsmall = x(j)
           ismall = j
        END IF
     NEXT j
     SWAP x(i),x(ismall)
     SWAP y(i),y(ismall)
  NEXT i
  d32 = x(3)-x(2)
  d13 = x(1)-x(3)
  d21 = x(2)-x(1)
  yvalue = y(1) + d21*( y(1)-y(3) )/d13
  IF yvalue > y(2) THEN flag = 0 ELSE flag = 1
  rnum = y(1)*d32*(x(3)+x(2))+y(2)*d13*(x(1)+x(3))+y(3)*d21*(x(2)+x(1))
  rden = y(1)*d32 + y(2)*d13 + y(3)*d21
  xvalue = rnum/rden/2
  '
END SUB 'interp
'
```

```
'
DEF FNflagrange (blimit, pmean, ar10)
   '
   ' Given the value of the inner control limit, blimit, this function
   ' computes the value of the outer control limit that will result in-
   ' control ARL of ARLo and computes and returns the value of ARL at
   ' pmean. If a value for the outer limit cannot be found the function
   ' adds a penalty to the value of the ARL at pmean.
   '
  SHARED n.max, x.tol, y.tol
  LOCAL xlow, x1, x2, arlnull, arl
  CALL limit.app (ar10, blimit, x1, x2)
  CALL secant (x1, x2, n.max, x.tol, y.tol, ar10, arlnull, xlimit )
  '
  CALL init.matrix (blimit, xlimit, pmean)     ' Initialize (I-Q)
  CALL arl( a(), n, arl)                       ' Compute ARL for pmean
  IF ABS(arlnull-ar10)< 10*y.tol THEN
     FN flagrange = arl
  ELSE
     FN flagrange = arl + ABS(arlnull-ar10)
  END IF
END DEF 'flagrange
```

```
'
SUB secant ( x1, x2, n.max,  x.tol, y.tol, arl0, arl, xlimit )
   '
   ' Secant Method Routine
   '
   ' Input:   starting points:                x1 and x2
   '          maximum number of iterations :  n.max
   '          target average run length:      arl0 (for pmean=0)
   '          tolerances:                     x.tol, y.tol
   '          routine to initialize (I-Q):    init.matrix
   '          routine to compute ARL:         arl ( , , )
   '
   ' Output:  xlimit, and corresponding ARL, arl.
   '
  SHARED A(), n, blimit, format3$
  LOCAL  f1, f2, f3, x3, k!, psec, qsec, zero
  zero = 1e-15
  '
  CALL init.matrix (blimit, x1, 0)     ' Compute ARL for x1
  CALL arl( A(), n, arl)
  f1=arl-arl0
  CALL init.matrix (blimit, x2, 0)     ' Compute ARL for x2
  CALL arl( A(), n, arl)
  f2=arl-arl0
  IF ABS(f2)>ABS(f1) THEN
     SWAP x1,x2
     SWAP f1,f2
  END IF
  '
  FOR k! = 1 TO n.max
     IF ABS(f2)<y.tol THEN
        xlimit=x2
        arl=f2+arl0
        EXIT SUB
     END IF
     s = f2/f1
     psec = (x1-x2)*s
     qsec = 1-s:
  '
     IF ABS(qsec) > zero THEN
       x3 = x2 -psec/qsec
       IF x3>6.0                        THEN    x3=6.0
       IF x3<blimit                     THEN    x3=blimit
       IF ABS(x2-x3)<x.tol*ABS(x2)      THEN
          xlimit = x2
          arl = f2 +arl0
          EXIT SUB
       END IF
```

```
        CALL init.matrix (blimit, x3, 0)     ' Compute ARL for x3
        CALL arl( A(), n, arl)
        f3=arl-arl0
        IF ABS(f3) > ABS(f2) THEN
           x1=x3:            f1=f3
        ELSE
           x1=x2:           f1=f2
           x2=x3:           f2=f3
        END IF
      END IF
   '
   NEXT k!
   PRINT "No convergence after";k!;" iterations"
END SUB ' secant
'
```

```
'
SUB arl ( array(2), n, arl.value )
      '
      ' This routine computes the average run length given the array (I-Q)
      ' Input:   square array "array( , )" containing (I-Q)
      '          array size, n
      ' Output:  average run length, arl.value
      '          lower triangular array equivalent to "array(,)"
      '
     LOCAL zero, irow, index, big, nrow, jcol, fctr, frmt$
     zero  = 1E-10
     frmt$ = "Largest possible pivot element #.####### in row ###"
     FOR irow = 1 TO N
        array(irow,0) = 1
     NEXT irow
     FOR irow = N TO 1 STEP -1
        index = 1
        big   = ABS(array(index,irow))
        FOR nrow = 2 TO irow
           IF big   < ABS(array(nrow,irow)) THEN
              big   = ABS(array(nrow,irow))
              index = nrow
           END IF
        NEXT nrow
        IF ABS(big) <= zero      THEN
           PRINT "Error in SUB arl [...]: Pivot element is zero !!"
           PRINT USING frmt$; big, index
           STOP
        END IF
        FOR jcol = 0 TO irow
            SWAP array(index,jcol), array(irow,jcol)
        NEXT jcol
        FOR jcol = 0 TO irow-1
           array(irow,jcol) = array(irow,jcol)/array(irow,irow)
        NEXT jcol
        array(irow,irow) = 1
        FOR nrow = 1 TO irow-1
          fctr   = array(nrow,irow)
          IF ABS(fctr) > zero    THEN
            FOR jcol = 0 TO irow
              array(nrow,jcol) = array(nrow,jcol) - fctr*array(irow,jcol)
            NEXT jcol
          END IF
        NEXT nrow
     NEXT irow
     arl.value = array(1,0)
END SUB 'arl
'
```

```
'
' Note: File "NORMAL.INC" contains a subroutine to compute cumulative
'       standard normal probabilities.
'
$INCLUDE "NORMAL.INC"
'
'=================================================================
'
SUB init.matrix (blimit, xlimit, pmean)
    '
    '   Initialize A = (I-Q) matrix
    '
    ' Input:  control limits:              xlimit, blimit
    '         process mean:                pmean
    '
    ' Output: A ( matrix [I-Q] )
    '
   SHARED A(), n
   LOCAL  i, j, nsize, z#, p1#, p2#, p3#, p4#, p5#, p6#
   LOCAL  pb, po, pc
    '
    ' Output from GS3411.BAS
    ' Non-absorbing states ...                    ¢
    '                                             |
    '  1 ) OOO      2 ) OOB      3 ) OOC    +x +-------
    '  4 ) OBO      5 ) OBB      6 ) OBC       |        B zone (2/3)
    '  7 ) OCO      8 ) OCB      9 ) OCC    +b +-------
    ' 10 ) BOO     11 ) BOB     12 ) BOC       |
    ' 13 ) BBO     14 ) BBC                  --+- -  "ok" zone
    ' 15 ) BCO     16 ) BCB     17 ) BCC       |
    ' 18 ) COO     19 ) COB     20 ) COC    -b +-------
    ' 21 ) CBO     22 ) CBB     23 ) CBC       |        C zone (2/3)
    ' 24 ) CCO     25 ) CCB                 -x +-------
'                                             |
   z# = ( xlimit-pmean ):    p1# = FNCDFnormal#( z# )
   z# = ( blimit-pmean ):    p2# = FNCDFnormal#( z# ) :      pb=p1#-p2#
   z# = (-blimit-pmean ):    p3# = FNCDFnormal#( z# ) :      po=p2#-p3#
   z# = (-xlimit-pmean ):    p4# = FNCDFnormal#( z# ) :      pc=p3#-p4#
   nsize = n
   FOR i = 1 TO nsize
      FOR j = 1 TO nsize:    a(i,j) = 0:                  NEXT j
   NEXT i
```

```
    a( 1,  1)=-po:    a( 1,  2)=-pb:    a( 1,  3)=-pc:
    a( 2,  4)=-po:    a( 2,  5)=-pb:    a( 2,  6)=-pc:
    a( 3,  7)=-po:    a( 3,  8)=-pb:    a( 3,  9)=-pc:
    a( 4, 10)=-po:    a( 4, 11)=-pb:    a( 4, 12)=-pc:
    a( 5, 13)=-po:    a( 5, 14)=-pc:
    a( 6, 15)=-po:    a( 6, 16)=-pb:    a( 6, 17)=-pc:
    a( 7, 18)=-po:    a( 7, 19)=-pb:    a( 7, 20)=-pc:
    a( 8, 21)=-po:    a( 8, 22)=-pb:    a( 8, 23)=-pc:
    a( 9, 24)=-po:    a( 9, 25)=-pb:
    a(10,  1)=-po:    a(10,  2)=-pb:    a(10,  3)=-pc:
    a(11,  4)=-po:    a(11,  6)=-pc:
    a(12,  7)=-po:    a(12,  8)=-pb:    a(12,  9)=-pc:
    a(13, 10)=-po:    a(13, 12)=-pc:
    a(14, 15)=-po:    a(14, 17)=-pc:
    a(15, 18)=-po:    a(15, 19)=-pb:    a(15, 20)=-pc:
    a(16, 21)=-po:    a(16, 23)=-pc:
    a(17, 24)=-po:    a(17, 25)=-pb:
    a(18,  1)=-po:    a(18,  2)=-pb:    a(18,  3)=-pc:
    a(19,  4)=-po:    a(19,  5)=-pb:    a(19,  6)=-pc:
    a(20,  7)=-po:    a(20,  8)=-pb:
    a(21, 10)=-po:    a(21, 11)=-pb:    a(21, 12)=-pc:
    a(22, 13)=-po:    a(22, 14)=-pc:
    a(23, 15)=-po:    a(23, 16)=-pb:
    a(24, 18)=-po:    a(24, 19)=-pb:
    a(25, 21)=-po:    a(25, 22)=-pb:
    '
    '
    FOR i=1 TO nsize:          a(i,i) = a(i,i) + 1:          NEXT i
    '
END SUB 'init.matrix
'
' ============================================================================
'
SUB limit.app (ar10, blimit, x1, x2)
'
' This subroutine returns approx. values for xlimit, x1 and x2, given the
' desired in-control ARL, ar10, and a value for blimit.
'
  LOCAL ar11n, xvalue
  ar11n  = LOG(ar10)
  xvalue = 0.67 + 0.286*blimit¢2 + 1.527*ar11n/blimit¢2
  xvalue = xvalue - 0.03029*ar11n - 3.871/blimit¢2
  x1     = FNfmax(xvalue-.10, blimit)
  x2     = FNfmax(xvalue, blimit) + 0.10
END SUB 'limit.app
'
'============================================================================
' END OF FILE * END OF FILE * END OF FILE * END OF FILE * END OF FILE *
```

```
'   Noel Artiles.   September 1988.
'   File: P3411B.BAS
'
'   AVERAGE-RUN-LENGTH CALCULATIONS
'   RULES CONSIDERED IN THIS PROGRAM:
'
'          Stop if 3/3 points fall in (-oo, -b), or
'          stop if 3/4 points fall in ( b, +oo), or
'          stop if 1/1 point  fall in (-oo,-x) U (x, +oo).
'
'   This program generate the ARL curves corresponding to the optimal
'   values of b and x. The in-control ARL considered are 100, 200, ...,
'   500 and the process mean varies from 0.00 to 2.00 by 0.10.
'
'=============================================================================
'
CLS
DEFINT i-n                                      ' Define integer variables.
DEFDBL a-h,o-z                                  ' Define double prec. vars.
DEF FNfmax(x,y) = (x+y+abs(x-y))/2              ' Define Max{} function
DEF FNfmin(x,y) = (x+y-abs(x-y))/2              ' Definr Min{} function
n = 25:                  ' Size of matrix (I-Q)
n.max=50:                ' Maximum number of iterations for secant method
x.tol=1.0D-10            ' Tolerance in x.limit for secant method
CALL init.normal         ' Initialize constants for Normal.prob. routine
DIM a(0:25, 0:25)        ' Matrix a contains (I-Q) matrix
DIM clim(3), arlout(3)
'
OPEN "P3411A1.PRN" FOR OUTPUT AS #1
FOR iarl = 100 TO 500 STEP 100
   arl0  = iarl
   y.tol = arl0*(1.0E-8)
   arl0log = LOG(arl0)
   blimit = -.18781 + 0.340719*LOG(arl0) - .0124218*LOG(arl0)¢2
   xlimit = 10
   PRINT #1, USING "Control Limit = ##.####"; blimit
   FOR pmean = 0 to 2.01 step 0.05
      CALL init.matrix (blimit, xlimit, pmean)    ' Initialize (I-Q)
      CALL arl( a(), n, arl)                      ' Compute ARL for pmean
      PRINT #1, USING " ##.##   ####.###"; pmean, arl
      PRINT     USING " ##.##   ####.###"; pmean, arl
  NEXT pmean
  PRINT #1,
  PRINT
NEXT iarl
'
' Note: File "NORMAL.INC" contains a subroutine to compute cumulative
'       standard normal probabilities.
'
$INCLUDE "NORMAL.INC"
```

```
'                          ** FILE: NORMAL.INC **
SUB init.normal
      '
      '   Subroutine to initialize constants used to compute
      '   standard normal probabilities.
      '
      SHARED  p1#(), q1#(), p2#(), q2#(), p3#(), q3#()
      p1#(0)=2.42667955230531D2:            q1#(0)=2.15058875869861D2
      p1#(1)=2.19792616182941D1:            q1#(1)=9.11649054045149D1
      p1#(2)=6.99638348861913D0:            q1#(2)=1.50827976304078D1
      p1#(3)=-3.56098437018153D-2:          q1#(3)=1.00000000000000D0
      p2#(0)=3.004592610201616OD2:          q2#(0)=3.0045926095698329D2
      p2#(1)=4.5191895371187294D2:          q2#(1)=7.9095092532789803D2
      p2#(2)=3.3932081673434368D2:          q2#(2)=9.3135409485060962D2
      p2#(3)=1.529892850469404OD2:          q2#(3)=6.3898026446563117D2
      p2#(4)=4.3162227222056735D1:          q2#(4)=2.7758544474398764D2
      p2#(5)=7.2117582508830936D0:          q2#(5)=7.7000152935229473D1
      p2#(6)=5.6419551747897397D-1:         q2#(6)=1.2782727319629424D1
      p2#(7)=-1.3686485738271677D-7:        q2#(7)=1.0000000000000000D0
      p3#(0)=-2.996007077035421D-3:         q3#(0)=1.062092305284679D-2
      p3#(1)=-4.947309106232507D-2:         q3#(1)=1.913089261078298D-1
      p3#(2)=-2.269565935396869D-1:         q3#(2)=1.051675107067932D0
      p3#(3)=-2.786613086096478D-1:         q3#(3)=1.987332018171353D0
      p3#(4)=-2.231924597341847D-2:         q3#(4)=1.000000000000000D0
END SUB 'init_normal
'
'=============================================================================
'
DEF FN erf# (x#)
      '
      ' This function evaluate the error function, for 0 <= x < 0.50:
      '
      LOCAL   fctr#,   snum#,   sden#,   j%
      SHARED  p1#(),   q1#(),   p2#(),   q2#(),   p3#(),   q3#()
      '
      IF x# < 0  OR  x# >= 0.5 THEN
         CLS:                                                        BEEP
         PRINT "*** Error: Invalid argument in function erf(x).":    STOP
      END IF
      fctr# = 1
      snum# = 0
      sden# = 0
      FOR j% = 0 TO 3
         snum#    =  snum# + p1#(j%)*fctr#
         sden#    =  sden# + q1#(j%)*fctr#
         fctr#    =  fctr#*x#*x#
      NEXT j%
      FN erf#  =  x#*snum#/sden#
END DEF 'erf#
```

```
DEF FN erfc#(x#)
     '
     ' This function evaluate the complementary error function,
     ' for 0.46875 < =  x
     '
     LOCAL   fctr#,   snum#,   sden#,   j%,     y#
     SHARED  p1#(),   q1#(),   p2#(),   q2#(),  p3#(),  q3#()
     '
     IF x#<.46875 THEN
        CLS:   BEEP
        PRINT "*** Error: Invalid argument in function erfc(x)."
        STOP
     ELSEIF x#<4.000 THEN
        fctr# = 1:  snum# = 0:  sden# = 0
        FOR j% = 0 TO 7
           snum# = snum# + p2#(j%)*fctr#
           sden# = sden# + q2#(j%)*fctr#
           fctr# = fctr#*x#
        NEXT j%
        FN erfc#  = EXP(-x#*x#)*snum#/sden#
     ELSE
        y# = 1/x#/x#:  fctr# = 1
        snum# = 0:     sden# = 0
        FOR j% = 0 TO 4
           snum# = snum# + p3#(j%)*fctr#
           sden# = sden# + q3#(j%)*fctr#
           fctr# = fctr#/y#/y#
        NEXT j%
        FNerfc#=EXP(-x#*x#)/x#*(1/SQR(3.1415926#)+snum#/sden#/x#/x#)
     END IF
END DEF 'erfc#
'
'=====================================================================
'
DEF FN CDFnormal#(z#)
  '
  ' This function computes the CDF of a standard normal random variable
  '
 x# = Z#/SQR(2#)'x# is the argument used in erf(x) and erfc(x) routines
 IF x#<0              THEN x# = -x#:                  GOTO negative:
 IF x#>5.65           THEN FNCDFnormal# =  1#:        EXIT DEF
 IF x#>=0 AND x#<.5# THEN FNCDFnormal# = (1+FN erf#(x#))/2: EXIT DEF
 IF x#>=.5 AND x#<=4 THEN FNCDFnormal# = (2-FNerfc#(x#))/2: EXIT DEF
 IF x#>4 AND x#<5.65 THEN FNCDFnormal# = (2-FNerfc#(x#))/2: EXIT DEF
 negative:
 IF x#> 5.65          THEN  FN CDFnormal# = 0:                EXIT DEF
 IF x#>=0 AND x#<.5# THEN  FNCDFnormal# = (1-FNerf#(x#))/2: EXIT DEF
 IF x#>=.5 AND x#<=4 THEN  FNCDFnormal# = FN erfc#(x#)/2:   EXIT DEF
 IF x#>4 AND x#<5.65 THEN  FNCDFnormal# = FN erfc#(x#)/2:   EXIT DEF
END DEF 'FN CDFnormal#
```

# X. APPENDIX C: PROGRAM GENSTATE.BAS AND SUBROUTINES CHECKST.INC

```
' FILE: GENSTATE.BAS
' Noel Artiles.     August 1988.
'
' This program generates the possible states for the set of control rules
' given below and also the corresponding transition probabilities.
' NOTE: Only non-absorbing states are considered.
'
'            c
'            |       zone X: 1 out of 1 (+x, +oo)
'         +x +============
'            |       zone A: 2 out of 3 (+a, +oo)
'         +a +============
'            |       zone B: 3 out of 4 (+b, +oo)
'         +b +============
'            |       zone O:
'            |                  ok.
'         -b +============
'            |       zone C: 3 out of 4 (-oo, -b)
'         -a +============
'            |       zone D: 2 out of 3 (-oo, -a)
'         -x +============
'            |       zone X: 1 out of 1 (-oo, -x)
'
'
z$(1)="O":   z$(2)="A":      z$(3)="B"
z$(4)="C":   z$(5)="D"
nstate = 0:  CLS:            nzone = 5
flnm$="genstate.prn"
OPEN flnm$ FOR OUTPUT AS #1
DIM state$(100)
PRINT #1, "Output from GS342311.BAS"
PRINT #1, "Non-absorbing states ...":      PRINT #1,
FOR i = 1 TO nzone
   state1$=z$(I)
   FOR j=1 TO nzone
      state2$=state1$+z$(J)
      FOR k = 1 TO nzone
         state3$=state2$+z$(K)
         IF FNicheck34( state3$ ) + FNicheck23( state3$ ) = 0 THEN
'
'           Print non-absorbing state
'
            nstate         = nstate+1
            state$(nstate) = state3$
            PRINT #1,nstate;") ";state3$,
         END IF
      NEXT k
      PRINT #1,
   NEXT j
NEXT I
```

```
PRINT #1,
PRINT "Press any key to continue ..."
d$ = "" : WHILE d$ = "": D$=INKEY$: WEND
'
FORMAT1$="##### "
FORMAT2$="A(##\\##)=\ \: "
PRINT #1,:          PRINT #1, "Transition probabilities...": PRINT #1,
FOR i = 1 TO nstate
   FOR k=1 TO nzone
      ns$=RIGHT$(state$(i),2)+z$(k)
      IF FNicheck34( state$(i)+z$(k) ) = 0 THEN
        IF   FNicheck23( ns$ )         = 0 THEN
         FOR j = 1 TO nstate
             IF ns$ = state$(j) THEN
                PRINT #1,USING FORMAT2$ ;i;",";j;"-P"+z$(k),
                GOTO nextk:
             END IF
         NEXT j
        END IF
      END IF
   nextk:
   NEXT k
PRINT #1,
NEXT i
STOP
END
'
$INCLUDE "CHECKST.INC"
```

```
'
' Noel Artiles.    August 1988
' File: checkst.inc
'
' ========================================================================
'
DEF FNiscan( s$, c$, k )
'
' This function scans the string s$ and search for the occurrence of the
' character c$. If c$ appears AT LEAST k times in s$, the function returns
' the value of 1, otherwise it returns the value of 0.
'
  LOCAL t$, icounter, ipos
  IF LEN(c$)<>1 THEN
     PRINT "Error 1 in function FNiscan."
     STOP
  END IF
  IF LEN(s$) < LEN(c$) THEN
     PRINT "Error 2 in function FNiscan."
     STOP
  END IF
  IF K<1 or k> LEN(s$) THEN
     PRINT "Error 3 in function FNiscan."
     STOP
  END IF
'
  t$ = s$
  icounter = 0
  ipos = INSTR( t$, c$ )
  WHILE ipos>0
     icounter = icounter + 1
     IF icounter >= k THEN
        FNiscan = 1
        EXIT DEF
     ELSE
        MID$(t$, ipos, 1) = chr$(254)
     END IF
     ipos = INSTR( t$, c$ )
  WEND
  FNiscan = 0
END DEF 'iscan
'
```

```
'
DEF FNicheck23( s$ )
'
' This function checks whether 2 or more "A"s, or 2 or more "D"s appear in
' the string s$; in this case the function returns the value of 1,
' otherwise it returns 0.
'
  IF FNiscan (s$, "A", 2) = 1 THEN
     FNicheck23 = 1
     EXIT DEF
  END IF
  IF FNiscan (s$, "D", 2) = 1 THEN
     FNicheck23 = 1
     EXIT DEF
  END IF
  FNicheck23 = 0
END DEF 'FNicheck23
'
'=============================================================================
'
DEF FNicheck34( s$ )
'
' This function returns the value of 1 if:
'      there are 3 or more ("A"s or "B"s) in s$, or
'      there are 3 or more ("D"s or "C"s) in s$,
' otherwise it returns the value of 0.
'
IF FNiscan( s$, "A", 3) = 1                    THEN FNicheck34=1: EXIT DEF
IF FNiscan( s$, "B", 3) = 1                    THEN FNicheck34=1: EXIT DEF
IF ( FNiscan(s$,"A",2)+FNiscan(s$,"B",1) = 2 ) THEN FNicheck34=1: EXIT DEF
IF ( FNiscan(s$,"A",1)+FNiscan(s$,"B",2) = 2 ) THEN FNicheck34=1: EXIT DEF
'
IF FNiscan( s$, "D", 3) = 1                    THEN FNicheck34=1: EXIT DEF
IF FNiscan( s$, "C", 3) = 1                    THEN FNicheck34=1: EXIT DEF
IF ( FNiscan(s$,"D",2)+FNiscan(s$,"C",1) = 2 ) THEN FNicheck34=1: EXIT DEF
IF ( FNiscan(s$,"D",1)+FNiscan(s$,"C",2) = 2 ) THEN FNicheck34=1: EXIT DEF
'
FNicheck34 = 0
END DEF 'FNicheck34
```

# XI. APPENDIX D: PROGRAMS P3423A.BAS AND P3423B.BAS

```
'
'   Noel Artiles.   September 1988.
'   File: P3423A.BAS
'
'   AVERAGE-RUN-LENGTH CALCULATIONS
'
'   RULES CONSIDERED IN THIS PROGRAM:
'
'           Stop if 3/4 points fall in (-oo, -b), or
'           stop if 3/4 points fall in ( b, +oo), or
'           Stop if 2/3 points fall in (-oo, -x), or
'           stop if 2/3 points fall in ( x, +oo).
'
'   This program computes values for b and x that will result in a
'   given in-control ARL, ARL0, and that minimize the out-of-control
'   average run length (process mean = 1).
'
'=============================================================================
'
CLS
DEFINT i-n                                      ' Define integer variables.
DEFSNG a-h,o-z                                  ' Define single prec. vars.
DEF FNfmax(x,y) = (x+y+abs(x-y))/2              ' Define Max{} function
DEF FNfmin(x,y) = (x+y-abs(x-y))/2              ' Definr Min{} function
'
n = 91                  ' Size of matrix (I-Q)
n.max=10:               ' Maximum number of iterations for secant method
x.tol=1.0E-05           ' Tolerance in x.limit for secant method
CALL init.normal        ' Initialize constants for Normal.prob. routine
DIM a(0:91,0:91)        ' Matrix a contains (I-Q) matrix
DIM clim(3), arlout(3)
'
format1$ = "B = #.#####   X = #.#####  ARLzero = ####.####"
format2$ = format1$+"  ARLone = ####.####"
'
OPEN "P3423X1.PRN" FOR APPEND AS #1
'
'                           Optimize for ARLo = 100, 150, ..., 500
'
FOR iarl = 400 TO 400 STEP 50
   arl0  = iarl
   y.tol = arl0*(1.0E-6)
   pmean = 1.00
   PRINT USING "     arl0 = ###.##"; arl0
```

```
'
'                          Define an initial set of values for the
'                          inner limit (blimit)
blimit = 1.03 + .79*ar10/1000
FOR k = 1 TO 3
  clim(k)   = blimit
  arlout(k) = FNflagrange (blimit, pmean, ar10 )
  blimit    = blimit + 0.05
  print clim(k), arlout(k)
NEXT k
back:
icounter = 1
vlen = ABS( clim(3)-clim(1) )
WHILE (icounter < 500  AND vlen > .005 )
  CALL interp ( clim(), arlout(), flag, blimit)
  ysmall = arlout(1):  ismall = 1
  FOR i = 2 TO 3
     IF ysmall > arlout(i) THEN
        ysmall = arlout(i)
        ismall = i
     END IF
  NEXT i
  print using "###.######  "; clim(ismall), ysmall
  vlen = ABS( clim(3)-clim(1) )
  icounter = icounter + 1
  '
  IF flag = 1 THEN  '       If flag=1 the function is not convex
  '                         in current interval.
     PRINT
     PRINT " Warning: Function is not convex in current interval."
     PRINT "    Rerun the program with different starting points."
     PRINT
     FOR k = 1 TO 3
        print clim(k), arlout(k)
     NEXT k
     INPUT " Enter a new value for inner control limit: ";blimit
     arlout(1) = FNflagrange (blimit, pmean, ar10 )
     clim(1)   = blimit
     GOTO back:
                  '
  ELSE            ' If function is convex in current interval,
                  ' replace worst control limit by the approximation
                  ' computed by subroutine "interp".
                  '
     ar11     = FNflagrange (blimit, pmean, ar10 )
     print using "###.######  "; blimit, ar11: print
     arlworst = arlout(1)
     iworst   = 1
```

```
        FOR i = 2 TO 3
           IF arlworst < arlout(i) THEN
              arlworst = arlout(i)
              iworst   = i
           END IF
        NEXT i
        clim(iworst)   = blimit
        arlout(iworst) = arll
     END IF
  WEND
  '
  '                              Best inner control limit has been found.
  '                              Compute corresponding outer control limit.
  '
  bestarl = arlout(1):  ibest = 1
  FOR i = 2 TO 3
     IF bestarl > arlout(i) THEN
        bestarl = arlout(i)
        ibest   = i
     END IF
  NEXT i
  blimit = clim(ibest)
  CALL limit.app (arl0, blimit, x1, x2)
  CALL secant (x1, x2, n.max, x.tol, y.tol, arl0, arl, xlimit )
  PRINT #1, USING format2$; blimit, xlimit, arl, arlout(ibest)
  PRINT     USING format2$; blimit, xlimit, arl, arlout(ibest)
  PRINT
NEXT iarl
CLOSE #1
'
PRINT "Done !!"
END
'
```

```
'
SUB interp ( x(1), y(1), flag, xvalue )
  '
  ' This function uses quadratic interpolation to approximate the
  ' minimum of a function. If the set of points (x1,y1), (x2,y2),
  ' and (x3, y3) does not fall on a convex function the value of
  ' flag is set to 1, otherwise it is set to zero.
  '
  LOCAL d32, d13, d21, rnum, rden, xsmall, ismall, yvalue, i, j
  '
  FOR i = 1 TO 3
     xsmall = x(i)
     ismall = i
     FOR j = i TO 3
        IF xsmall > x(j) THEN
           xsmall = x(j)
           ismall = j
        END IF
     NEXT j
     SWAP x(i),x(ismall)
     SWAP y(i),y(ismall)
  NEXT i
  d32 = x(3)-x(2)
  d13 = x(1)-x(3)
  d21 = x(2)-x(1)
  yvalue = y(1) + d21*( y(1)-y(3) )/d13
  IF yvalue > y(2) THEN flag = 0 ELSE flag = 1
  rnum = y(1)*d32*(x(3)+x(2))+y(2)*d13*(x(1)+x(3))+y(3)*d21*(x(2)+x(1))
  rden = y(1)*d32 + y(2)*d13 + y(3)*d21
  xvalue = rnum/rden/2
  '
END SUB 'interp
'
```

```
'
DEF FNflagrange (blimit, pmean, arl0)
   '
   ' Given the value of the inner control limit, blimit, this function
   ' computes the value of the outer control limit that will result
   ' in-control ARL of ARLo and computes and returns the value of ARL
   ' at pmean. If a value for the outer limit cannot be found the
   ' function adds a penalty to the value of the ARL at pmean.
   '
  SHARED n.max, x.tol, y.tol
  LOCAL xlow, x1, x2, arlnull, arl
  CALL limit.app (arl0, blimit, x1, x2)
  CALL secant (x1, x2, n.max, x.tol, y.tol, arl0, arlnull, xlimit )
  '
  CALL init.matrix (blimit, xlimit, pmean)    ' Initialize (I-Q)
  CALL arl( a(), n, arl)                      ' Compute ARL for pmean
  IF ABS(arlnull-arl0)< 10*y.tol THEN
     FN flagrange = arl
  ELSE
     FN flagrange = arl + ABS(arlnull-arl0)
  END IF
END DEF 'flagrange
'
```

```
'
SUB secant ( x1, x2, n.max,  x.tol, y.tol, arl0, arl, xlimit )
    '
    ' Secant Method Routine
    '
    ' Input:   starting points:                x1 and x2
    '          maximum number of iterations :  n.max
    '          target average run length:      arl0 (for pmean=0)
    '          tolerances:                     x.tol, y.tol
    '          routine to initialize (I-Q):    init.matrix
    '          routine to compute ARL:         arl ( , , )
    '
    ' Output:  xlimit, and corresponding ARL, arl.
    '
   SHARED A(), n, blimit
   LOCAL  f1, f2, f3, x3, k!, psec, qsec, zero, fmt$
   zero = 1e-15
   fmt$="sec1: x1=###.### f1=#####.###  x2=###.### f2=#####.###"
   '
   CALL init.matrix (blimit, x1, 0)     ' Compute ARL for x1
   CALL arl( A(), n, arl)
   f1=arl-arl0
   CALL init.matrix (blimit, x2, 0)     ' Compute ARL for x2
   CALL arl( A(), n, arl)
   f2=arl-arl0
   IF ABS(f2)>ABS(f1) THEN
      SWAP x1,x2
      SWAP f1,f2
   END IF
   '
   FOR k! = 1 TO n.max
      'print using fmt$; x1,f1,x2,f2
      IF ABS(f2)<y.tol THEN
         xlimit=x2
         arl=f2+arl0
         EXIT SUB
      END IF
      s = f2/f1
      psec = (x1-x2)*s
      qsec = 1-s:
   '
      IF ABS(qsec) > zero THEN
        x3 = x2 -psec/qsec
        IF x3>6.0                       THEN    x3=6.0
        IF x3<blimit                    THEN    x3=blimit
        IF ABS(x2-x3)<x.tol*ABS(x2)     THEN
           xlimit = x2
           arl = f2 +arl0
           EXIT SUB
        END IF
```

```
        CALL init.matrix (blimit, x3, 0)     ' Compute ARL for x3
        CALL arl( A(), n, arl)
        f3=arl-arl0
        IF ABS(f3) > ABS(f2) THEN
           x1=x3:           f1=f3
        ELSE
           x1=x2:          f1=f2
           x2=x3:          f2=f3
        END IF
      END IF
   '
   NEXT k!
   PRINT "No convergence after";k!;" iterations"
   xlimit = x2
   arl = f2 +arl0
END SUB ' secant
'
```

```
'
SUB arl ( array(2), n, arl.value )
      '
      ' This routine computes the average run length given the array (I-Q)
      ' Input:   square array "array( , )" containing (I-Q)
      '          array size, n
      ' Output:  average run length, arl.value
      '          lower triangular array equivalent to "array(,)"
      '
      LOCAL zero, irow, index, big, nrow, jcol, fctr, frmt$
      zero  = 1E-10
      frmt$ = "Largest possible pivot element #.####### in row ###"
      '
      FOR irow = 1 TO N
         array(irow,0) = 1
      NEXT irow
      '
      FOR irow = N TO 1 STEP -1
         index = 1
         big   = ABS(array(index,irow))
         FOR nrow = 2 TO irow
            IF big   < ABS(array(nrow,irow)) THEN
               big   = ABS(array(nrow,irow))
               index = nrow
            END IF
         NEXT nrow
         IF ABS(big) <= zero      THEN
            PRINT
            PRINT "Error in SUB arl [...]: Pivot element is zero !!"
            PRINT USING frmt$; big, index
            STOP
         END IF
         FOR jcol = 0 TO irow
             SWAP array(index,jcol), array(irow,jcol)
         NEXT jcol
         FOR jcol = 0 TO irow-1
             array(irow,jcol) = array(irow,jcol)/array(irow,irow)
         NEXT jcol
         array(irow,irow) = 1
         FOR nrow = 1 TO irow-1
           fctr   = array(nrow,irow)
           IF ABS(fctr) > zero    THEN
             FOR jcol = 0 TO irow
               array(nrow,jcol) = array(nrow,jcol)-fctr*array(irow,jcol)
             NEXT jcol
           END IF
         NEXT nrow
      NEXT irow
      arl.value = array(1,0)
END SUB 'arl
```

```
'
' Note: File "NORMAL.INC" contains a subroutine to compute cumulative
'       standard normal probabilities.
'
$INCLUDE "NORMAL.INC"
'
'=============================================================================
'
SUB init.matrix (blimit, xlimit, pmean)
    '
    '   Initialize A = (I-Q) matrix
    '
    ' Input:  control limits:              xlimit, blimit
    '         process mean:                pmean
    '
    ' Output: A ( matrix [I-Q] )
    '
    '                       Output from GS342311.BAS
    '                        Non-absorbing states
    '                                                        Notation:
    '  1) OOO    2) OOA    3) OOB    4) OOC    5) OOD
    '  6) OAO    7) OAB    8) OAC    9) OAD   10) OBO      ¢
    ' 11) OBA   12) OBB   13) OBC   14) OBD   15) OCO        |  zone "A" (2/3)
    ' 16) OCA   17) OCB   18) OCC   19) OCD   20) ODO     +x+--------------
    ' 21) ODA   22) ODB   23) ODC   24) AOO   25) AOB        |
    ' 26) AOC   27) AOD   28) ABO   29) ABC   30) ABD        |  zone "B" (3/4)
    ' 31) ACO   32) ACB   33) ACC   34) ACD   35) ADO     +b+--------------
    ' 36) ADB   37) ADC   38) BOO   39) BOA   40) BOB        |
    ' 41) BOC   42) BOD   43) BAO   44) BAC   45) BAD        |  zone "O"
    ' 46) BBO   47) BBC   48) BBD   49) BCO   50) BCA        |
    ' 51) BCB   52) BCC   53) BCD   54) BDO   55) BDA   -b+--------------
    ' 56) BDB   57) BDC   58) COO   59) COA   60) COB        |
    ' 61) COC   62) COD   63) CAO   64) CAB   65) CAC        |  zone "C" (3/4)
    ' 66) CAD   67) CBO   68) CBA   69) CBB   70) CBC   -x+--------------
    ' 71) CBD   72) CCO   73) CCA   74) CCB   75) CDO        |
    ' 76) CDA   77) CDB   78) DOO   79) DOA   80) DOB        |  zone "D" (2/3)
    ' 81) DOC   82) DAO   83) DAB   84) DAC   85) DBO
    ' 86) DBA   87) DBB   88) DBC   89) DCO   90) DCA
    ' 91) DCB
    '
    SHARED A(), n
    LOCAL  i, j, nsize, z#, p1#, p2#, p3#, p4#, p5#, p6#
    LOCAL  pb, po, pc
    '
    z# = ( xlimit-pmean):  p1# = FNCDFnormal#( z#) :  pa=1-p1#
    z# = ( blimit-pmean):  p2# = FNCDFnormal#( z#) :  pb=p1#-p2#
    z# = (-blimit-pmean):  p3# = FNCDFnormal#( z#) :  po=p2#-p3#
    z# = (-xlimit-pmean):  p4# = FNCDFnormal#( z#) :  pc=p3#-p4# : pd=p4#
    nsize = 91
```

```
IF nsize<>n THEN
   PRINT "Error in SUB init.matrix ( , , ): incorrect matrix size."
   STOP
END IF
FOR i = 1 TO nsize
   FOR j = 1 TO nsize:    a(i,j) = 0:                    NEXT j
NEXT i

a(1, 1)=-po: a(1, 2)=-pa: a(1, 3)=-pb: a(1, 4)=-pc: a(1, 5)=-pd:
a(2, 6)=-po: a(2, 7)=-pb: a(2, 8)=-pc: a(2, 9)=-pd:
a(3,10)=-po: a(3,11)=-pa: a(3,12)=-pb: a(3,13)=-pc: a(3,14)=-pd:
a(4,15)=-po: a(4,16)=-pa: a(4,17)=-pb: a(4,18)=-pc: a(4,19)=-pd:
a(5,20)=-po: a(5,21)=-pa: a(5,22)=-pb: a(5,23)=-pc:
a(6,24)=-po: a(6,25)=-pb: a(6,26)=-pc: a(6,27)=-pd:
a(7,28)=-po: a(7,29)=-pc: a(7,30)=-pd:
a(8,31)=-po: a(8,32)=-pb: a(8,33)=-pc: a(8,34)=-pd:
a(9,35)=-po: a(9,36)=-pb: a(9,37)=-pc:
a(10,38)=-po: a(10,39)=-pa: a(10,40)=-pb: a(10,41)=-pc: a(10,42)=-pd:
a(11,43)=-po: a(11,44)=-pc: a(11,45)=-pd:
a(12,46)=-po: a(12,47)=-pc: a(12,48)=-pd:
a(13,49)=-po: a(13,50)=-pa: a(13,51)=-pb: a(13,52)=-pc: a(13,53)=-pd:
a(14,54)=-po: a(14,55)=-pa: a(14,56)=-pb: a(14,57)=-pc:
a(15,58)=-po: a(15,59)=-pa: a(15,60)=-pb: a(15,61)=-pc: a(15,62)=-pd:
a(16,63)=-po: a(16,64)=-pb: a(16,65)=-pc: a(16,66)=-pd:
a(17,67)=-po: a(17,68)=-pa: a(17,69)=-pb: a(17,70)=-pc: a(17,71)=-pd:
a(18,72)=-po: a(18,73)=-pa: a(18,74)=-pb:
a(19,75)=-po: a(19,76)=-pa: a(19,77)=-pb:
a(20,78)=-po: a(20,79)=-pa: a(20,80)=-pb: a(20,81)=-pc:
a(21,82)=-po: a(21,83)=-pb: a(21,84)=-pc:
a(22,85)=-po: a(22,86)=-pa: a(22,87)=-pb: a(22,88)=-pc:
a(23,89)=-po: a(23,90)=-pa: a(23,91)=-pb:
a(24, 1)=-po: a(24, 2)=-pa: a(24, 3)=-pb: a(24, 4)=-pc: a(24, 5)=-pd:
a(25,10)=-po: a(25,13)=-pc: a(25,14)=-pd:
a(26,15)=-po: a(26,16)=-pa: a(26,17)=-pb: a(26,18)=-pc: a(26,19)=-pd:
a(27,20)=-po: a(27,21)=-pa: a(27,22)=-pb: a(27,23)=-pc:
a(28,38)=-po: a(28,41)=-pc: a(28,42)=-pd:
a(29,49)=-po: a(29,52)=-pc: a(29,53)=-pd:
a(30,54)=-po: a(30,57)=-pc:
a(31,58)=-po: a(31,59)=-pa: a(31,60)=-pb: a(31,61)=-pc: a(31,62)=-pd:
a(32,67)=-po: a(32,70)=-pc: a(32,71)=-pd:
a(33,72)=-po: a(33,73)=-pa: a(33,74)=-pb:
a(34,75)=-po: a(34,76)=-pa: a(34,77)=-pb:
a(35,78)=-po: a(35,79)=-pa: a(35,80)=-pb: a(35,81)=-pc:
a(36,85)=-po: a(36,88)=-pc:
a(37,89)=-po: a(37,90)=-pa: a(37,91)=-pb:
a(38, 1)=-po: a(38, 2)=-pa: a(38, 3)=-pb: a(38, 4)=-pc: a(38, 5)=-pd:
a(39, 6)=-po: a(39, 8)=-pc: a(39, 9)=-pd:
a(40,10)=-po: a(40,13)=-pc: a(40,14)=-pd:
a(41,15)=-po: a(41,16)=-pa: a(41,17)=-pb: a(41,18)=-pc: a(41,19)=-pd:
a(42,20)=-po: a(42,21)=-pa: a(42,22)=-pb: a(42,23)=-pc:
```

```
a(43,24)=-po: a(43,26)=-pc: a(43,27)=-pd:
a(44,31)=-po: a(44,33)=-pc: a(44,34)=-pd:
a(45,35)=-po: a(45,37)=-pc:
a(46,38)=-po: a(46,41)=-pc: a(46,42)=-pd:
a(47,49)=-po: a(47,52)=-pc: a(47,53)=-pd:
a(48,54)=-po: a(48,57)=-pc:
a(49,58)=-po: a(49,59)=-pa: a(49,60)=-pb: a(49,61)=-pc: a(49,62)=-pd:
a(50,63)=-po: a(50,65)=-pc: a(50,66)=-pd:
a(51,67)=-po: a(51,70)=-pc: a(51,71)=-pd:
a(52,72)=-po: a(52,73)=-pa: a(52,74)=-pb:
a(53,75)=-po: a(53,76)=-pa: a(53,77)=-pb:
a(54,78)=-po: a(54,79)=-pa: a(54,80)=-pb: a(54,81)=-pc:
a(55,82)=-po: a(55,84)=-pc:
a(56,85)=-po: a(56,88)=-pc:
a(57,89)=-po: a(57,90)=-pa: a(57,91)=-pb:
a(58, 1)=-po: a(58, 2)=-pa: a(58, 3)=-pb: a(58, 4)=-pc: a(58, 5)=-pd:
a(59, 6)=-po: a(59, 7)=-pb: a(59, 8)=-pc: a(59, 9)=-pd:
a(60,10)=-po: a(60,11)=-pa: a(60,12)=-pb: a(60,13)=-pc: a(60,14)=-pd:
a(61,15)=-po: a(61,16)=-pa: a(61,17)=-pb:
a(62,20)=-po: a(62,21)=-pa: a(62,22)=-pb:
a(63,24)=-po: a(63,25)=-pb: a(63,26)=-pc: a(63,27)=-pd:
a(64,28)=-po: a(64,29)=-pc: a(64,30)=-pd:
a(65,31)=-po: a(65,32)=-pb:
a(66,35)=-po: a(66,36)=-pb:
a(67,38)=-po: a(67,39)=-pa: a(67,40)=-pb: a(67,41)=-pc: a(67,42)=-pd:
a(68,43)=-po: a(68,44)=-pc: a(68,45)=-pd:
a(69,46)=-po: a(69,47)=-pc: a(69,48)=-pd:
a(70,49)=-po: a(70,50)=-pa: a(70,51)=-pb:
a(71,54)=-po: a(71,55)=-pa: a(71,56)=-pb:
a(72,58)=-po: a(72,59)=-pa: a(72,60)=-pb:
a(73,63)=-po: a(73,64)=-pb:
a(74,67)=-po: a(74,68)=-pa: a(74,69)=-pb:
a(75,78)=-po: a(75,79)=-pa: a(75,80)=-pb:
a(76,82)=-po: a(76,83)=-pb:
a(77,85)=-po: a(77,86)=-pa: a(77,87)=-pb:
a(78, 1)=-po: a(78, 2)=-pa: a(78, 3)=-pb: a(78, 4)=-pc: a(78, 5)=-pd:
a(79, 6)=-po: a(79, 7)=-pb: a(79, 8)=-pc: a(79, 9)=-pd:
a(80,10)=-po: a(80,11)=-pa: a(80,12)=-pb: a(80,13)=-pc: a(80,14)=-pd:
a(81,15)=-po: a(81,16)=-pa: a(81,17)=-pb:
a(82,24)=-po: a(82,25)=-pb: a(82,26)=-pc: a(82,27)=-pd:
a(83,28)=-po: a(83,29)=-pc: a(83,30)=-pd:
a(84,31)=-po: a(84,32)=-pb:
a(85,38)=-po: a(85,39)=-pa: a(85,40)=-pb: a(85,41)=-pc: a(85,42)=-pd:
a(86,43)=-po: a(86,44)=-pc: a(86,45)=-pd:
a(87,46)=-po: a(87,47)=-pc: a(87,48)=-pd:
a(88,49)=-po: a(88,50)=-pa: a(88,51)=-pb:
a(89,58)=-po: a(89,59)=-pa: a(89,60)=-pb:
a(90,63)=-po: a(90,64)=-pb:
a(91,67)=-po: a(91,68)=-pa: a(91,69)=-pb:
```

```
    '
    FOR i=1 TO nsize
        a(i,i) = a(i,i) + 1
    NEXT i
    '
END SUB 'init.matrix
'
'=====================================================================
'
SUB limit.app( arl0,blimit,a1,a2 )
'
' This subroutine return approx. values for Alimit,a1 and a2,given the
' desired in-control ARL,arl0,and a value for Blimit.
'
   LOCAL arlln,xvalue,denom
   arlln  = LOG(arl0)
   denom  = (arl0/blimit¢6) - 55
   avalue = 0.6177 + 0.191*arlln + 1.2898/blimit¢5 - 4.6912/denom
   a1     = FNfmax( avalue-0.15,blimit )
   a2     = FNfmax( xvalue,blimit ) + 0.15
'
END SUB ' limit.app
'
'=====================================================================
'  END OF FILE   END OF FILE   END OF FILE   END OF FILE   END OF FILE
```

```
'
'   Noel Artiles.   September 1988.
'   File: P3423B.BAS
'
'   AVERAGE-RUN-LENGTH CALCULATIONS
'
'   RULES CONSIDERED IN THIS PROGRAM:
'
'           Stop if 3/4 points fall in (-oo, -b), or
'           stop if 3/4 points fall in ( b, +oo), or
'           Stop if 2/3 points fall in (-oo, -x), or
'           stop if 2/3 points fall in ( x, +oo).
'
'   This program computes values for b and x that will result in a
'   given in-control ARL, ARL0, and that minimize the out-of-control
'   average run length (process mean = 1).
'
'=============================================================================
'
CLS
DEFINT i-n                              ' Define integer variables.
DEFSNG a-h,o-z                          ' Define single prec. vars.
DEF FNfmax(x,y) = (x+y+abs(x-y))/2      ' Define Max{} function
DEF FNfmin(x,y) = (x+y-abs(x-y))/2      ' Define Min{} function
'
n = 91              ' Size of matrix (I-Q)
n.max=10:           ' Maximum number of iterations for secant method
x.tol=1.0E-05       ' Tolerance in x.limit for secant method
CALL init.normal    ' Initialize constants for Normal.prob. routine
DIM a(0:91,0:91)    ' Matrix a contains (I-Q) matrix
DIM clim(3), arlout(3)
OPEN "P3423X2.PRN" FOR OUTPUT AS #1
FOR iarl = 100 TO 500 STEP 100
   arl0  = iarl
   y.tol = arl0*(1.0E-6)
   blimit = -.0319887527+.3182770169*LOG(arl0)-.011882544*LOG(arl0)¢2
   CALL limit.app (arl0, blimit, xl, x2)
   CALL secant (xl, x2, n.max, x.tol, y.tol, arl0, arl, xlimit )
   PRINT:    PRINT #1,
   PRINT     USING " b = ##.####  x = ##.#### "; blimit, xlimit
   PRINT #1, USING " b = ##.####  x = ##.#### "; blimit, xlimit
   FOR pmean = 0.00 to 2.01 STEP .05
      CALL init.matrix( blimit, xlimit, pmean)
      CALL arl ( a(), n, arlval)
      PRINT     USING " ###.#####  "; pmean, arlval
      PRINT #1, USING " ###.#####  "; pmean, arlval
   NEXT pmean:              PRINT
NEXT iarl
CLOSE #1:                   PRINT "Done !!"
END
```

```
'
' Note: File "NORMAL.INC" contains a subroutine to compute cumulative
'       standard normal probabilities.
'
$INCLUDE "NORMAL.INC"
'
'=============================================================================
'
SUB limit.app( ar10,blimit,a1,a2 )
'
' This subroutine return approx. values for Alimit,a1 and a2,given the
' desired in-control ARL,ar10,and a value for Blimit.
'
   LOCAL ar11n,xvalue,denom
   avalue = 0.269*LOG(ar10) + 0.624
   a1     = FNfmax( avalue-0.02,blimit )
   a2     = FNfmax( xvalue,blimit ) + 0.02
'
END SUB ' limit.app
'
'=============================================================================
'  END OF FILE   END OF FILE   END OF FILE   END OF FILE   END OF FILE
```

# XII. APPENDIX E: PROGRAMS P342311A.BAS AND P342311B.BAS

```
'
'   Noel Artiles.   September 1988.
'   File: P342311A.BAS
'
'   AVERAGE-RUN-LENGTH CALCULATIONS
'
'   RULES CONSIDERED IN THIS PROGRAM:
'
'           Stop if 3/4 points fall in (-oo, -b), or
'           stop if 3/4 points fall in ( b, +oo), or
'           stop if 2/3 points fall in ( a, +oo), or
'           stop if 2/3 points fall in (-oo, -a), or
'           stop if 1/1 point  fall in (-oo,-x) U (x, +oo).
'
'   This program finds values for a, b and x that will result in a
'   given in-control ARL, ARL0, and that give a minimum value for the
'   out-of-control average run length.
'
'=============================================================================
'
CLS
DEFINT i-n                                  ' Define integer variables.
DEFSNG a-h,o-z                              ' Define single prec. vars.
DEF FNfmax(x,y) = (x+y+abs(x-y))/2          ' Define Max{} function
DEF FNfmin(x,y) = (x+y-abs(x-y))/2          ' Definr Min{} function
'
n = 91:                 ' Size of matrix (I-Q)
n.max=10:               ' Maximum number of iterations for secant method
x.tol=1.0E-5            ' Tolerance in x.limit for secant method
CALL init.normal        ' Initialize constants for Normal.prob. routine
DIM a(0:91,0:91)        ' Matrix a contains (I-Q) matrix
'
format1$="B = #.####  A = #.####  X = #.##### ARLzero = ####.####"
format2$=format1$+"  ARLone = ####.####"
format3$="X1 = ##.#### F(X1)= ####.##   X2 = ##.#### F(X2)= ####.##"
format4$=">> ARLzero = ###.# BLIMIT = ##.### ALIMIT = ##.### << "
'
for iarl = 450 to 500 step 50
'
flnm$ = "F34XB" + RIGHT$( STR$(iarl),3 ) + ".PRN"
OPEN flnm$ FOR APPEND AS #1
PRINT "Writting to file ";flnm$
arl0   = iarl
y.tol  = arl0*.00001            ' Tolerance in ARL for secant method
bvalue = 0.19*LOG(arl0)+0.33    ' Initial value for blimit
b.lo   = bvalue - 0.07
b.hi   = bvalue + 0.05
avalue = 0.23*LOG(arl0)+0.94    ' Initial value for a limit
a.hi   = avalue + 0.10
a.lo   = avalue - 0.12
```

```
FOR blimit = b.lo TO b.hi STEP 0.01
   arllast = 1e+30
   index   = 1
   FOR alimit = a.hi TO a.lo step -0.01
      pmean = 0                ' Process mean
      PRINT
      PRINT USING format4$; ar10, blimit, alimit
      PRINT
      CALL limit.app( ar10, alimit, blimit, x1, x2 )
      CALL secant   (x1, x2, n.max, x.tol, y.tol, ar10, ar1, xlimit )
      IF ABS(ar1-ar10)<10*y.tol THEN
         PRINT "Exact control limits ..."
         ar1.temp = ar1
         pmean = 1!
         CALL init.matrix (alimit, blimit, xlimit, pmean)
         CALL ar1( a(), n, ar1)
         PRINT    USING format2$; blimit,alimit,xlimit,ar1.temp, ar1
         PRINT #1,USING format2$; blimit,alimit,xlimit,ar10    , ar1
         PRINT
         IF ar1 > arllast THEN
            IF index >= 3 THEN GOTO nextblimit:
            index = index + 1
         END IF
         arllast = ar1
      ELSE
         GOTO nextblimit:
      END IF
'
   NEXT alimit
   nextblimit:
   PRINT
   PRINT #1,
'
NEXT blimit
CLOSE #1
NEXT iar1
'
PRINT "Done !!"
END
'
```

```
SUB secant ( x1, x2, n.max,  x.tol, y.tol, arl0, arl, xlimit )
   '
   ' Secant Method Routine
   '
   ' Input:   starting points:                x1 and x2
   '          maximum number of iterations :  n.max
   '          target average run length:      arl0 (for pmean=0)
   '          tolerances:                     x.tol, y.tol
   '          routine to initialize (I-Q):    init.matrix
   '          routine to compute ARL:         arl ( , , )
   '
   ' Output:  xlimit, and corresponding ARL, arl.
   '
   SHARED A(), n, alimit, blimit, format3$
   LOCAL  f1, f2, f3, x3, k!, psec, qsec, zero
   zero = 1.0E-15
   '
   CALL init.matrix (alimit, blimit, x1, 0)     ' Compute ARL for x1
   CALL arl( A(), n, arl)
   f1=arl-arl0
   CALL init.matrix (alimit, blimit, x2, 0)     ' Compute ARL for x2
   CALL arl( A(), n, arl)
   f2=arl-arl0
   IF ABS(f2)>ABS(f1) THEN
      SWAP x1,x2
      SWAP f1,f2
   END IF
   FOR k! = 1 TO n.max
      IF ABS(f2)<y.tol THEN
         xlimit=x2
         arl=f2+arl0
         PRINT "Tolerance for ARL satisfied ..."
         EXIT SUB
      END IF
      PRINT USING format3$; x1,f1,x2,f2
      s = f2/f1
      psec = (x1-x2)*s
      qsec = 1-s:
   '
      IF ABS(qsec) > zero THEN
        x3 = x2 -psec/qsec
        IF x3>8.0                       THEN    x3=8.0
        IF x3<alimit                    THEN    x3=alimit
        CALL init.matrix (alimit, blimit, x3, 0)
        CALL arl( A(), n, arl)
        IF ABS(x2-x3)<x.tol*ABS(x2)     THEN
           xlimit = x3
           PRINT "Tolerance for XLIMIT satisfied ..."
           EXIT SUB
        END IF
```

```
        f3=ar1-ar10
        IF ABS(f3) > ABS(f2) THEN
           x1=x3:           f1=f3
        ELSE
           x1=x2:          f1=f2
           x2=x3:          f2=f3
        END IF
      END IF
   '
   NEXT k!
   'PRINT "No convergence after";k!;" iterations"
END SUB ' secant
'
```

```
'
SUB limit.app ( ar10, alimit, blimit, x1, x2 )
'
' Subroutine to find starting points for secant routine
'
' INPUT:  desired ARL:                              ARL0
'         control limits:                           ALIMIT, BLIMIT
' OUTPUT: starting points for secant method:        X1, X2
'
LOCAL xvalue, ratio
ratio = FNfmin( 900, (-25 + ar10/blimit¢6)¢2 )
xvalue= 2.8384 - 2.1885*LOG(blimit) + 0.6271*LOG(ar10) - 0.2491*alimit
xvalue= xvalue - 0.059644*SQR( 900 - ratio )
x1    = xvalue - 0.10
x2    = xvalue + 0.10
IF x2 > 5 THEN x1 = 4.00: x2 = 5.00
'
END SUB 'limit.app
'
'=================================================================
'
' Notes: [1] File "P342311.INC" contains a subroutine to initialize
'            the matrix (I-Q) given the control limits.
'        [2] File "NORMAL.INC" contains a subroutine to compute
'            cumulative standard normal probabilities.
'
$INCLUDE "P342311.INC"
$INCLUDE "NORMAL.INC"
'
'=================================================================
' END OF FILE ** END OF FILE ** END OF FILE ** END OF FILE
```

```
'
'  Noel Artiles.   September 1988.
'  File: P342311B.BAS
'
'   AVERAGE-RUN-LENGTH CALCULATIONS
'
'   RULES CONSIDERED IN THIS PROGRAM:
'
'           Stop if 3/4 points fall in (-oo, -b), or
'           stop if 3/4 points fall in ( b, +oo), or
'           stop if 2/3 points fall in ( a, +oo), or
'           stop if 2/3 points fall in (-oo, -a), or
'           stop if 1/1 point  fall in (-oo,-x) U (x, +oo).
'
'   This program computes the optimal ARL curve (as a function of
'   shift in the process mean) for given in-control ARL values of
'   100, 200, ..., 500.
'
'=====================================================================
'
CLS
DEFINT i-n                                  ' Define integer variables.
DEFSNG a-h,o-z                              ' Define single prec. vars.
DEF FNfmax(x,y) = (x+y+abs(x-y))/2          ' Define Max{} function
DEF FNfmin(x,y) = (x+y-abs(x-y))/2          ' Definr Min{} function
'
n = 91:                ' Size of matrix (I-Q)
n.max=10:              ' Maximum number of iterations for secant method
x.tol=1.0E-5           ' Tolerance in x.limit for secant method
CALL init.normAl       ' Initialize constants for Normal.prob. routine
DIM a(0:91,0:91)       ' Matrix a contains (I-Q) matrix
'
format1$="B = #.####  A = #.####  X = #.##### ARLzero = ####.####"
format2$=format1$+"  ARLone = ####.####"
format3$="X1 = ##.#### F(X1)= ####.##   X2 = ##.#### F(X2)= ####.##"
format4$="BLIMIT = ##.###   ALIMIT = ##.###   XLIMIT = ##.###"
format5$="       #.###        ####.###"
'
OPEN "F34XOXX1.PRN" FOR APPEND AS #1
for iarl = 100 to 500 step 100
   arl0   = iarl
   y.tol  = arl0*.00001       ' Tolerance in ARL for secant method
   blimit = 0.3511 + 0.1839*LOG(arl0)
   alimit = 0.8779 + 0.2404*LOG(arl0)
   x1     = 0.9012 + 0.5276*LOG(arl0)
   x2     = x1 + 0.01
   pmean = 0                  ' Process mean
   PRINT
```

```
    CALL secant   (x1, x2, n.max, x.tol, y.tol, ar10, ar1, xlimit )
    PRINT
    PRINT #1,
    IF ABS(ar1-ar10)<5*y.tol THEN
       PRINT #1, USING format4$; blimit, alimit, xlimit
       PRINT #1,
       PRINT #1, "  Process Mean      ARL "
       PRINT #1, USING format5$; pmean, ar1
       PRINT     USING format4$; blimit, alimit, xlimit
       PRINT
       PRINT     "  Process Mean      ARL "
       PRINT     USING format5$; pmean, ar1
       '
       FOR pmean = 0.05 TO 2.5 STEP 0.10
          CALL init.matrix (alimit, blimit, xlimit, pmean)
          CALL ar1( a(), n, ar1)
          PRINT    USING format5$; pmean, ar1
          PRINT #1,USING format5$; pmean, ar1
       NEXT pmean
       '
    ELSE
       PRINT " Xlimit cannot be found !"
       CLOSE #1
       STOP
    END IF
NEXT iar1
PRINT
PRINT " Done !!"
CLOSE #1
END
'
'====================================================================
'
' Notes: [1] File "P342311.INC" contains a subroutine to initialize
'            the matrix (I-Q) given the control limits.
'        [2] File "NORMAL.INC" contains a subroutine to compute
'            cumulative standard normal probabilities.
'
$INCLUDE "P342311.INC"
$INCLUDE "NORMAL.INC"
'
'====================================================================
' END OF FILE ** END OF FILE ** END OF FILE ** END OF FILE
```

**FILE: P342311.INC**

```
'
SUB init.matrix (alimit, blimit, xlimit, pmean)
  '
  '   Initialize A = (I-Q) matrix
  '
  ' Input:  control limits:              alimit, blimit, xlimit
  '         process mean:               pmean
  ' Output: A ( matrix [I-Q] )
  '
  '                       Output from GS342311.BAS
  '                        Non-absorbing states
  '                                            Notation:
  '                                               c
  '                                               |
  '  1)OOO    2)OOA    3)OOB    4)OOC    5)OOD      |  zone "X" (1/1)
  '  6)OAO    7)OAB    8)OAC    9)OAD   10)OBO    +x+-------------
  ' 11)OBA   12)OBB   13)OBC   14)OBD   15)OCO      |  zone "A" (2/3)
  ' 16)OCA   17)OCB   18)OCC   19)OCD   20)ODO    +a+--------------
  ' 21)ODA   22)ODB   23)ODC   24)AOO   25)AOB      |
  ' 26)AOC   27)AOD   28)ABO   29)ABC   30)ABD      |  zone "B" (3/4)
  ' 31)ACO   32)ACB   33)ACC   34)ACD   35)ADO    +b+--------------
  ' 36)ADB   37)ADC   38)BOO   39)BOA   40)BOB      |
  ' 41)BOC   42)BOD   43)BAO   44)BAC   45)BAD      |  zone "O"
  ' 46)BBO   47)BBC   48)BBD   49)BCO   50)BCA      |
  ' 51)BCB   52)BCC   53)BCD   54)BDO   55)BDA    -b+--------------
  ' 56)BDB   57)BDC   58)COO   59)COA   60)COB      |
  ' 61)COC   62)COD   63)CAO   64)CAB   65)CAC      |  zone "C" (3/4)
  ' 66)CAD   67)CBO   68)CBA   69)CBB   70)CBC    -a+--------------
  ' 71)CBD   72)CCO   73)CCA   74)CCB   75)CDO      |
  ' 76)CDA   77)CDB   78)DOO   79)DOA   80)DOB      |  zone "D" (2/3)
  ' 81)DOC   82)DAO   83)DAB   84)DAC   85)DBO    -x+--------------
  ' 86)DBA   87)DBB   88)DBC   89)DCO   90)DCA      |
  ' 91)DCB                                          |  zone "X" (1/1)
  '
  SHARED A(), n
  LOCAL  i, j, nsize, z#, p1#, p2#, p3#, p4#, p5#, p6#
  LOCAL  pb, po, pc
  '
  z# = ( xlimit-pmean):  p0# = FNCDFnormal1#( z#)
  z# = ( alimit-pmean):  p1# = FNCDFnormal1#( z#) :  pa=p0#-p1#
  z# = ( blimit-pmean):  p2# = FNCDFnormal1#( z#) :  pb=p1#-p2#
  z# = (-blimit-pmean):  p3# = FNCDFnormal1#( z#) :  po=p2#-p3#
  z# = (-alimit-pmean):  p4# = FNCDFnormal1#( z#) :  pc=p3#-p4#
  z# = (-xlimit-pmean):  p5# = FNCDFnormal1#( z#) :  pd=p4#-p5#
  nsize = 91
  IF nsize<>n THEN
     PRINT "Error in SUB init.matrix ( , , ): incorrect matrix size."
     STOP
  END IF
```

```
FOR i = 1 TO nsize
   FOR j = 1 TO nsize:    a(i,j) = 0:                    NEXT j
NEXT i

a( 1, 1)=-po: a( 1, 2)=-pa: a( 1, 3)=-pb: a( 1, 4)=-pc: a( 1, 5)=-pd
a( 2, 6)=-po: a( 2, 7)=-pb: a( 2, 8)=-pc: a( 2, 9)=-pd
a( 3,10)=-po: a( 3,11)=-pa: a( 3,12)=-pb: a( 3,13)=-pc: a( 3,14)=-pd
a( 4,15)=-po: a( 4,16)=-pa: a( 4,17)=-pb: a( 4,18)=-pc: a( 4,19)=-pd
a( 5,20)=-po: a( 5,21)=-pa: a( 5,22)=-pb: a( 5,23)=-pc:
a( 6,24)=-po: a( 6,25)=-pb: a( 6,26)=-pc: a( 6,27)=-pd
a( 7,28)=-po: a( 7,29)=-pc: a( 7,30)=-pd
a( 8,31)=-po: a( 8,32)=-pb: a( 8,33)=-pc: a( 8,34)=-pd
a( 9,35)=-po: a( 9,36)=-pb: a( 9,37)=-pc:
a(10,38)=-po: a(10,39)=-pa: a(10,40)=-pb: a(10,41)=-pc: a(10,42)=-pd
a(11,43)=-po: a(11,44)=-pc: a(11,45)=-pd
a(12,46)=-po: a(12,47)=-pc: a(12,48)=-pd
a(13,49)=-po: a(13,50)=-pa: a(13,51)=-pb: a(13,52)=-pc: a(13,53)=-pd
a(14,54)=-po: a(14,55)=-pa: a(14,56)=-pb: a(14,57)=-pc:
a(15,58)=-po: a(15,59)=-pa: a(15,60)=-pb: a(15,61)=-pc: a(15,62)=-pd
a(16,63)=-po: a(16,64)=-pb: a(16,65)=-pc: a(16,66)=-pd
a(17,67)=-po: a(17,68)=-pa: a(17,69)=-pb: a(17,70)=-pc: a(17,71)=-pd
a(18,72)=-po: a(18,73)=-pa: a(18,74)=-pb:
a(19,75)=-po: a(19,76)=-pa: a(19,77)=-pb:
a(20,78)=-po: a(20,79)=-pa: a(20,80)=-pb: a(20,81)=-pc:
a(21,82)=-po: a(21,83)=-pb: a(21,84)=-pc:
a(22,85)=-po: a(22,86)=-pa: a(22,87)=-pb: a(22,88)=-pc:
a(23,89)=-po: a(23,90)=-pa: a(23,91)=-pb:
a(24, 1)=-po: a(24, 3)=-pb: a(24, 4)=-pc: a(24, 5)=-pd
a(25,10)=-po: a(25,13)=-pc: a(25,14)=-pd
a(26,15)=-po: a(26,17)=-pb: a(26,18)=-pc: a(26,19)=-pd
a(27,20)=-po: a(27,22)=-pb: a(27,23)=-pc:
a(28,38)=-po: a(28,41)=-pc: a(28,42)=-pd
a(29,49)=-po: a(29,52)=-pc: a(29,53)=-pd
a(30,54)=-po: a(30,57)=-pc:
a(31,58)=-po: a(31,60)=-pb: a(31,61)=-pc: a(31,62)=-pd
a(32,67)=-po: a(32,70)=-pc: a(32,71)=-pd
a(33,72)=-po: a(33,74)=-pb:
a(34,75)=-po: a(34,77)=-pb:
a(35,78)=-po: a(35,80)=-pb: a(35,81)=-pc:
a(36,85)=-po: a(36,88)=-pc:
a(37,89)=-po: a(37,91)=-pb:
a(38, 1)=-po: a(38, 2)=-pa: a(38, 3)=-pb: a(38, 4)=-pc: a(38, 5)=-pd
a(39, 6)=-po: a(39, 8)=-pc: a(39, 9)=-pd
a(40,10)=-po: a(40,13)=-pc: a(40,14)=-pd
a(41,15)=-po: a(41,16)=-pa: a(41,17)=-pb: a(41,18)=-pc: a(41,19)=-pd
a(42,20)=-po: a(42,21)=-pa: a(42,22)=-pb: a(42,23)=-pc:
a(43,24)=-po: a(43,26)=-pc: a(43,27)=-pd
a(44,31)=-po: a(44,33)=-pc: a(44,34)=-pd
a(45,35)=-po: a(45,37)=-pc:
```

```
    a(46,38)=-po: a(46,41)=-pc: a(46,42)=-pd
    a(47,49)=-po: a(47,52)=-pc: a(47,53)=-pd
    a(48,54)=-po: a(48,57)=-pc:
    a(49,58)=-po: a(49,59)=-pa: a(49,60)=-pb: a(49,61)=-pc: a(49,62)=-pd
    a(50,63)=-po: a(50,65)=-pc: a(50,66)=-pd
    a(51,67)=-po: a(51,70)=-pc: a(51,71)=-pd
    a(52,72)=-po: a(52,73)=-pa: a(52,74)=-pb:
    a(53,75)=-po: a(53,76)=-pa: a(53,77)=-pb:
    a(54,78)=-po: a(54,79)=-pa: a(54,80)=-pb: a(54,81)=-pc:
    a(55,82)=-po: a(55,84)=-pc:
    a(56,85)=-po: a(56,88)=-pc:
    a(57,89)=-po: a(57,90)=-pa: a(57,91)=-pb:
    a(58, 1)=-po: a(58, 2)=-pa: a(58, 3)=-pb: a(58, 4)=-pc: a(58, 5)=-pd
    a(59, 6)=-po: a(59, 7)=-pb: a(59, 8)=-pc: a(59, 9)=-pd
    a(60,10)=-po: a(60,11)=-pa: a(60,12)=-pb: a(60,13)=-pc: a(60,14)=-pd
    a(61,15)=-po: a(61,16)=-pa: a(61,17)=-pb:
    a(62,20)=-po: a(62,21)=-pa: a(62,22)=-pb:
    a(63,24)=-po: a(63,25)=-pb: a(63,26)=-pc: a(63,27)=-pd
    a(64,28)=-po: a(64,29)=-pc: a(64,30)=-pd
    a(65,31)=-po: a(65,32)=-pb:
    a(66,35)=-po: a(66,36)=-pb:
    a(67,38)=-po: a(67,39)=-pa: a(67,40)=-pb: a(67,41)=-pc: a(67,42)=-pd
    a(68,43)=-po: a(68,44)=-pc: a(68,45)=-pd
    a(69,46)=-po: a(69,47)=-pc: a(69,48)=-pd
    a(70,49)=-po: a(70,50)=-pa: a(70,51)=-pb:
    a(71,54)=-po: a(71,55)=-pa: a(71,56)=-pb:
    a(72,58)=-po: a(72,59)=-pa: a(72,60)=-pb:
    a(73,63)=-po: a(73,64)=-pb:
    a(74,67)=-po: a(74,68)=-pa: a(74,69)=-pb:
    a(75,78)=-po: a(75,79)=-pa: a(75,80)=-pb:
    a(76,82)=-po: a(76,83)=-pb:
    a(77,85)=-po: a(77,86)=-pa: a(77,87)=-pb:
    a(78, 1)=-po: a(78, 2)=-pa: a(78, 3)=-pb: a(78, 4)=-pc:
    a(79, 6)=-po: a(79, 7)=-pb: a(79, 8)=-pc:
    a(80,10)=-po: a(80,11)=-pa: a(80,12)=-pb: a(80,13)=-pc:
    a(81,15)=-po: a(81,16)=-pa: a(81,17)=-pb:
    a(82,24)=-po:  a(82,25)=-pb:  a(82,26)=-pc:
    a(83,28)=-po:  a(83,29)=-pc:
    a(84,31)=-po:  a(84,32)=-pb:
    a(85,38)=-po:  a(85,39)=-pa:  a(85,40)=-pb:  a(85,41)=-pc:
    a(86,43)=-po:  a(86,44)=-pc:
    a(87,46)=-po:  a(87,47)=-pc:
    a(88,49)=-po:  a(88,50)=-pa:  a(88,51)=-pb:
    a(89,58)=-po:  a(89,59)=-pa:  a(89,60)=-pb:
    a(90,63)=-po:  a(90,64)=-pb:
    a(91,67)=-po:  a(91,68)=-pa:  a(91,69)=-pb:
    '
    FOR i=1 TO nsize:
        a(i,i) = a(i,i) + 1:        NEXT i
END SUB 'init.matrix
```

# XIII. APPENDIX F: SIMULATION PROGRAMS AND SUBROUTINES

```
'
' Noel Artiles-Leon.    September 1988.
' FILE: DBLEXP1.BAS
'
' PROGRAM   TO   SIMULATE   THE   CONTROL  SCHEME
' [1,1,3,+oo],[2,3,2,+oo],[4,5,1,+oo],[8,8,0,+oo]"
' (      DOUBLE  EXPONENTIAL  DISTRIBUTION      )
' ============================================
'
DEF FN fmax(x,y) = (x+y + ABS(X-Y)) /2
DIM samples(8), par(5), ar1(10000)
CLS
PI = 3.1415926
nvctr  = 8              ' Keep track of the last "nvctr" samples
ptype  = 2
ptype$ = "Double Exponential"
rtype$ = "R1[1,1,3,+oo],[2,3,2,+oo],[4,5,1,+oo],[8,8,0,+oo]"
par(2) = SQR(2)         ' Set rate = SQR(2) (standard deviation = 1)
nruns  = 1500           ' Set number of simulation runs = 1500
OPEN "OUTSIM.PRN" FOR APPEND AS #1
par(1) = 0.5
FOR kmean = 1 to 2
 par(1) = 2*par(1)
 PRINT #1,
 PRINT #1, ptype$, rtype$
 PRINT #1, USING "Parameters : ####.####   ####.####"; par(1), par(2)
 PRINT #1, USING "No. of runs: #####";nruns
 PRINT #1, "Average   Std. Dev. LoLimit   UpperLim."
 FOR i = 1 to nruns
   ar1(i) = 0
   flag   = 0
   FOR j = 1 to nvctr
      samples(j) = 0
   NEXT j
   WHILE flag = 0
      CALL generate (ptype, par(), 2,  xbar)
      CALL update   (xbar, samples(), nvctr)
      CALL check11  (samples(), 3.0, flag1)
      CALL check23  (samples(), 2.0, flag2)
      CALL check45  (samples(), 1.0, flag3)
      CALL check88  (samples(), 0.0, flag4)
      ar1(i) = ar1(i) + 1
      flag = flag1+flag2+flag3+flag4
   WEND
NEXT i
```

```
 CALL bastat ( ar1(), nruns, average, stddev)
 lolimit = average - 1.96*stddev/SQR(nruns)
 uplimit = average + 1.96*stddev/SQR(nruns)
 PRINT #1, USING "###.###  "; average, stddev, lolimit, uplimit
'
NEXT kmean
PRINT "Done !"
CLOSE #1
END
$INCLUDE "SIM.INC"
```

```
' Noel Artiles-Leon.    September 1988.
' FILE: DBLEXP2.BAS
'
' PROGRAM   TO   SIMULATE   THE   CONTROL  SCHEME
' [1,1,3.216,oo], [2,3,1.962,oo], [3,4,1.181,oo]
' (      DOUBLE  EXPONENTIAL  DISTRIBUTION      )
' ================================================
'
DEF FN fmax(x,y) = (x+y + ABS(X-Y)) /2
DIM samples(8), par(5), arl(10000)
CLS
PI = 3.1415926
nvctr  = 4              ' Keep track of the last "nvctr" samples
ptype  = 2
ptype$ = "Double Exponential"
rtype$ = "R[1,1,3.216,oo], [2,3,1.962,oo], [3,4,1.181,oo] "
par(2) = SQR(2)         ' Set rate = SQR(2) (standard deviation = 1)
nruns  = 1500           ' Set number of simulation runs = 1500
OPEN "OUTSIM.PRN" FOR APPEND AS #1
par(1) = 0.5
FOR kmean = 1 to 2
 par(1) = 2*par(1)
 PRINT #1,
 PRINT #1, ptype$, rtype$
 PRINT #1, USING "Parameters : ####.####   ####.####"; par(1), par(2)
 PRINT #1, USING "No. of runs: #####";nruns
 PRINT #1, "Average   Std. Dev. LoLimit   UpperLim."
 FOR i = 1 to nruns
   arl(i) = 0
   flag   = 0
   FOR j = 1 to nvctr
      samples(j) = 0
   NEXT j
   WHILE flag = 0
      CALL generate (ptype, par(), 2,  xbar)
      CALL update   (xbar, samples(), nvctr)
      CALL check11  (samples(), 3.126, flag1)
      CALL check23  (samples(), 1.962, flag2)
      CALL check34  (samples(), 1.181, flag3)
      arl(i) = arl(i) + 1
      flag = flag1 + flag2 + flag3
   WEND
NEXT i
```

```
 CALL bastat ( ar1(), nruns, average, stddev)
 lolimit = average - 1.96*stddev/SQR(nruns)
 uplimit = average + 1.96*stddev/SQR(nruns)
 PRINT #1, USING "###.###  "; average, stddev, lolimit, uplimit
'
NEXT kmean
PRINT "Done !"
CLOSE #1
END
$INCLUDE "SIM.INC"
```

```
'
' Noel Artiles-Leon.    September 1988.
' FILE: CAUCHY1.BAS
'
' PROGRAM   TO   SIMULATE   THE   CONTROL  SCHEME
' [1,1,3,+oo],[2,3,2,+oo],[4,5,1,+oo],[8,8,0,+oo]"
' (    C A U C H Y    D I S T R I B U T I O N    )
' ================================================
'
DEF FN fmax(x,y) = (x+y + ABS(X-Y)) /2
DIM samples(8), par(5), arl(10000)
CLS
PI = 3.1415926
nvctr  = 8              ' Keep track of the last "nvctr" samples
ptype  = 3
ptype$ = "Cauchy"
rtype$ = "R1[1,1,3,+oo],[2,3,2,+oo],[4,5,1,+oo],[8,8,0,+oo]"
par(1) = 0.50           ' Set mean
par(2) = 0.5011         ' Set scale parameter
nruns  = 1500           ' Set number of simulation runs = 1500
OPEN "OUTSIMXX.PRN" FOR APPEND AS #1
 PRINT #1,
 PRINT #1, ptype$, rtype$
 PRINT #1, USING "Parameters : ####.####   ####.####"; par(1), par(2)
 PRINT #1, USING "No. of runs: #####";nruns
 PRINT #1, "Average   Std. Dev. LoLimit   UpperLim."
 FOR i = 1 to nruns
   arl(i) = 0
   flag   = 0
   FOR j = 1 to nvctr
      samples(j) = 0
   NEXT j
   WHILE flag = 0
      CALL generate (ptype, par(), 2,  xbar)
      CALL update   (xbar, samples(), nvctr)
      CALL check11  (samples(), 3.0, flag1)
      CALL check23  (samples(), 2.0, flag2)
      CALL check45  (samples(), 1.0, flag3)
      CALL check88  (samples(), 0.0, flag4)
      arl(i) = arl(i) + 1
      flag = flag1+flag2+flag3+flag4
   WEND
 NEXT i
```

```
 CALL bastat ( ar1(), nruns, average, stddev)
 lolimit = average - 1.96*stddev/SQR(nruns)
 uplimit = average + 1.96*stddev/SQR(nruns)
 PRINT #1, USING "###.###  "; average, stddev, lolimit, uplimit
PRINT "Done !"
CLOSE #1
END
$INCLUDE "SIM.INC"
```

```
' Noel Artiles-Leon.    September 1988.
' FILE: CAUCHY2.BAS
'
' PROGRAM   TO   SIMULATE   THE   CONTROL  SCHEME
'  [1,1,3.216,oo], [2,3,1.962,oo], [3,4,1.181,oo]
' (    C A U C H Y    D I S T R I B U T I O N    )
' ================================================
'
DEF FN fmax(x,y) = (x+y + ABS(X-Y)) /2
DIM samples(8), par(5), arl(10000)
CLS
PI = 3.1415926
nvctr  = 4              ' Keep track of the last "nvctr" samples
ptype  = 3
ptype$ = "Cauchy"
rtype$ = "R[1,1,3.216,oo], [2,3,1.962,oo], [3,4,1.181,oo]"
par(1) = 0.50           ' Set mean
par(2) = 0.5011         ' Set scale parameter
nruns  = 1500           ' Set number of simulation runs = 1500
OPEN "OUTSIMCA.PRN" FOR APPEND AS #1
'
 PRINT #1,
 PRINT #1, ptype$, rtype$
 PRINT #1, USING "Parameters : ####.####   ####.####"; par(1), par(2)
 PRINT #1, USING "No. of runs: #####";nruns
 PRINT #1, "Average   Std. Dev. LoLimit   UpperLim."
 FOR i = 1 to nruns
   arl(i) = 0
   flag   = 0
   FOR j = 1 to nvctr
      samples(j) = 0
   NEXT j
   WHILE flag = 0
      CALL generate (ptype, par(), 2,  xbar)
      CALL update   (xbar, samples(), nvctr)
      CALL check11  (samples(), 3.216, flag1)
      CALL check23  (samples(), 1.962, flag2)
      CALL check34  (samples(), 1.181, flag3)
      arl(i) = arl(i) + 1
      flag = flag1 + flag2 + flag3
   WEND
   LOCATE 12,1: PRINT par(1),i
NEXT i
```

```
 CALL bastat ( ar1(), nruns, average, stddev)
 lolimit = average - 1.96*stddev/SQR(nruns)
 uplimit = average + 1.96*stddev/SQR(nruns)
 PRINT #1, USING "###.###  "; average, stddev, lolimit, uplimit
'
PRINT "Done !"
CLOSE #1
END
$INCLUDE "SIM.INC"
```

```
'
' Noel Artiles-Leon.    October 1988.
' FILE: ARMA1.BAS
'
' PROGRAM  TO  SIMULATE  THE  CONTROL  SCHEME:
' [1,1,3,+oo],[2,3,2,+oo],[4,5,1,+oo],[8,8,0,+oo]
' ( MIXED AUTOREGRESSIVE-MOVING AVERAGE PROCESS )
' ================================================
'
DEF FN fmax(x,y) = (x+y + ABS(X-Y)) /2
DIM samples(8), par(5), arl(10000)
CLS
PI = 3.1415926
nvctr  = 8              ' Keep track of the last "nvctr" samples
ptype$ = "A.R.M.A. (1)"
rtype$ = "R1[1,1,3,+oo],[2,3,2,+oo],[4,5,1,+oo],[8,8,0,+oo]"
DATA -0.938, 0.100, 0.10
DATA -0.515, 1.200, 0.20
DATA -0.024, 1.200, 0.40
nruns  = 1500           ' Set number of simulation runs = 1500
OPEN "OUTARMA.PRN" FOR OUTPUT AS #1
'
FOR k = 1 TO 3
 READ para, parb, sigma
 sigma = sqr(sigma)
 PRINT #1,
 PRINT #1, ptype$, rtype$
 PRINT #1, USING "Parameters: ##.####  ##.####  ##.###"; para,parb,sigma
 PRINT #1, USING "No. of runs: #####";nruns
 PRINT #1, "Average   Std. Dev. LoLimit   UpperLim."
 FOR i = 1 to nruns
   arl(i) = 0
   flag   = 0
   FOR j = 1 to nvctr
      samples(j) = 0
   NEXT j
   WHILE flag = 0
      oldxbar = samples(1)
      CALL arma1  (xbar, oldxbar, enm1, para, parb, sigma)
      CALL update   (xbar, samples(), nvctr)
      CALL check11  (samples(), 3.0, flag1)
      CALL check23  (samples(), 2.0, flag2)
      CALL check45  (samples(), 1.0, flag3)
      CALL check88  (samples(), 0.0, flag4)
      arl(i) = arl(i) + 1
      flag = flag1+flag2+flag3+flag4
   WEND
```

```
 NEXT i
 CALL bastat ( arl(), nruns, average, stddev)
 lolimit = average - 1.96*stddev/SQR(nruns)
 uplimit = average + 1.96*stddev/SQR(nruns)
 PRINT #1, USING "###.###  "; average, stddev, lolimit, uplimit
'
NEXT k
PRINT "Done !"
CLOSE #1
END
$INCLUDE "SIM.INC"
```

```
' Noel Artiles-Leon.    October 1988.
' FILE: ARMA2.BAS
'
' PROGRAM TO SIMULATE THE CONTROL SCHEME:
' [1,1,3.216,oo],[2,3,1.962,oo],[3,4,1.181,oo]
' ( MIXED AUTOREGRESSIVE-MOVING AVERAGE PROCESS )
' ================================================
'
DEF FN fmax(x,y) = (x+y + ABS(X-Y)) /2
DIM samples(8), par(5), arl(10000)
CLS
PI = 3.1415926
nvctr  = 4              ' Keep track of the last "nvctr" samples
ptype$ = "A.R.M.A. (1)"
rtype$ = "R1[1,1,3.216,oo],[2,3,1.962,oo],[3,4,1.181,oo]"
'
DATA -0.938, 0.100, 0.10
DATA -0.515, 1.200, 0.20
DATA -0.024, 1.200, 0.40
nruns  = 1500           ' Set number of simulation runs = 1500
OPEN "OUTARMA4.PRN" FOR OUTPUT AS #1
'
FOR k = 1 TO 3
 READ para, parb, sigma
 sigma = sqr(sigma)
 PRINT #1,
 PRINT #1, ptype$, rtype$
 PRINT #1, USING "Parameters: ##.####  ##.####  ##.###"; para,parb,sigma
 PRINT #1, USING "No. of runs: #####";nruns
 PRINT #1, "Average   Std. Dev. LoLimit   UpperLim."
 FOR i = 1 to nruns
   arl(i) = 0
   flag   = 0
   FOR j = 1 to nvctr
      samples(j) = 0
   NEXT j
   WHILE flag = 0
      oldxbar = samples(1)
      CALL arma1  (xbar, oldxbar, enm1, para, parb, sigma)
      CALL update   (xbar, samples(), nvctr)
      CALL check11  (samples(), 3.216, flag1)
      CALL check23  (samples(), 1.962, flag2)
      CALL check34  (samples(), 1.181, flag3)
      arl(i) = arl(i) + 1
      flag = flag1+flag2+flag3
   WEND
```

```
 NEXT i
 CALL bastat ( ar1(), nruns, average, stddev)
 lolimit = average - 1.96*stddev/SQR(nruns)
 uplimit = average + 1.96*stddev/SQR(nruns)
 PRINT #1, USING "###.###  "; average, stddev, lolimit, uplimit
'
NEXT k
PRINT "Done !"
CLOSE #1
END
$INCLUDE "SIM.INC"
```

```
'
' Noel Artiles-Leon.    September 1988.
' FILE: SIM.INC
'
' SUBROUTINES TO SIMULATE A CONTROL SCHEME
' UNDER  DIFFERENT  ALTERNATIVE HYPOTHESES
' ========================================
'
SUB generate (type, p(1), np, rndm)
   '
   ' This subroutine generates a random number according to a process
   ' specified by the variable "type". p() is an array containing "np"
   ' parameters necessary for the generation of the random number.
   ' The random number is returned in the variable "rndm".
   '
   LOCAL z, prob, t
   SHARED PI
   SELECT CASE type
      CASE 1   ' Normal distribution; p(1)=mean, p(2)=variance
         '
         z = SQR(-2*LOG(RND))*COS(2*PI*RND)
         rndm = SQR(p(2))*z+p(1)
         '
      CASE 2   ' Double exponential distribution, p(1)=mean, p(2)=rate
         '
         prob = RND
         IF prob <=0.50 THEN
            t = LOG(2*prob)/p(2)
         ELSE
            t = -LOG(2*(1-prob))/p(2)
         END IF
         rndm = t + p(1)
         '
      CASE 3  ' Cauchy distribution, p(1)=mean, p(2)=shape parameter
         '
         prob = RND
         rndm = p(1) + p(2)*TAN(PI*(prob-1/2))
         '
      CASE ELSE
         PRINT "*** Error in FNgenerate: invalid argument"
         STOP
   END SELECT
END SUB 'generate
'
```

```
'
SUB update (x, vector(1), n)
   LOCAL i
   FOR i = n TO 2 STEP -1
      vector(i) = vector(i-1)
   NEXT i
   vector(1) = x
END SUB 'update
'
'=====================================================================
'
SUB bastat ( array(1), n, fmean, stddev )
   '
   ' This subroutine computes the mean and the standard deviation of the
   ' n numbers stored in the array "array".
   '
   LOCAL i, sum
   IF n < 2 THEN PRINT "*** Error in SUB bastat: n < 2": STOP
   sum = 0.0
   FOR i = 1 to n
      sum = sum + array(i)
   NEXT I
   fmean = sum/n
   sum = 0.0
   FOR i = 1 to n
      sum = sum + (array(i) - fmean)¢2
   NEXT i
   stddev = SQR( sum / (n-1) )
END SUB 'bastat
'
'=====================================================================
'
SUB check (vector(1), n, k, a, b, flag)
   '
   ' This subroutine sets the value of "flag" to 1 if there are k or
   ' more elements in the array "vector()" in the interval (a,b).
   ' Otherwise, the value of "flag" is set to 0.
   '
   LOCAL counter, i
   counter = 0
   flag    = 0
   IF k > n THEN  PRINT "*** Error in SUB check: k > n." : STOP
   IF a > b THEN  PRINT "*** Error in SUB check: a > b." : STOP
   FOR i = 1 to n
      IF ( vector(i) > a AND vector(i) < b ) THEN counter = counter + 1
   NEXT i
   IF counter >= k THEN flag = 1
END SUB' check
'
```

```
'
SUB check11 (vector(1), a, flag)
    '
    ' This subroutine sets the value of "flag" to 1 if there is at
    ' least one element in the array "vector" in the interval
    ' (-oo,-a)U(+a, +oo).  Otherwise the value of "flag" is set to 0.
    '
    LOCAL big
    big  = 1.0E+30
    flag = 0
    IF a < 0 THEN PRINT "*** Error in SUB check11: a is negative." : STOP
    CALL check (vector(), 1, 1, a, big, flag)
    IF flag = 1 THEN EXIT SUB
    CALL check (vector(), 1, 1,-big,-a, flag)
END SUB ' check11
'
'=================================================================
'
SUB check23 (vector(1), a, flag)
    '
    ' This subroutine sets the value of "flag" to 1 if
    ' i)  2 out of the first 3 elements of the array "vector" fall in the
    '      interval (-oo, -a), or
    ' ii) 2 out of the first 3 elements of the array "vector" fall in the
    '      interval (+a, +oo).
    '
    LOCAL big
    big  = 1.0E+30
    flag = 0
    IF a < 0 THEN PRINT "*** Error in SUB check23: a is negative." : STOP
    CALL check (vector(), 3, 2, a, big, flag)
    IF flag = 1 THEN EXIT SUB
    CALL check (vector(), 3, 2,-big,-a, flag)
END SUB ' check23
'
```

```
'
SUB check34 (vector(1), a, flag)
    '
    ' This subroutine sets the value of "flag" to 1 if
    ' i)  3 out of the first 4 elements of the array "vector" fall in the
    '      interval (-oo, -a), or
    ' ii) 3 out of the first 4 elements of the array "vector" fall in the
    '      interval (+a, +oo).
    '
   LOCAL big
   big  = 1.0E+30
   flag = 0
   IF a < 0 THEN PRINT "*** Error in SUB check23: a is negative." : STOP
   CALL check (vector(), 4, 3, a, big, flag)
   IF flag = 1 THEN EXIT SUB
   CALL check (vector(), 4, 3,-big,-a, flag)
END SUB ' check34
'
'=====================================================================
'
SUB check45 (vector(1), a, flag)
    '
   ' This subroutine sets the value of "flag" to 1 if
   ' i)  4 out of the first 5 elements of the array "vector" fall in the
   '      interval (-oo, -a), or
   ' ii) 4 out of the first 5 elements of the array "vector" fall in the
   '      interval (+a, +oo).
    '
   LOCAL big
   big  = 1.0E+30
   flag = 0
   IF a < 0 THEN PRINT "*** Error in SUB check45: a is negative." : STOP
   CALL check (vector(), 5, 4, a, big, flag)
   IF flag = 1 THEN EXIT SUB
   CALL check (vector(), 5, 4,-big,-a, flag)
END SUB ' check45
'
```

```
'
SUB check88 (vector(1), a, flag)
    '
    ' This subroutine sets the value of "flag" to 1 if
    ' i)  8 out of the first 8 elements of the array "vector" fall in the
    '     interval (-oo, -a), or
    ' ii) 8 out of the first 8 elements of the array "vector" fall in the
    '     interval (+a, +oo).
    '
    LOCAL big
    big  = 1.0E+30
    flag = 0
    IF a < 0 THEN PRINT "*** Error in SUB check88: a is negative." : STOP
    CALL check (vector(), 8, 8, a, big, flag)
    IF flag = 1 THEN EXIT SUB
    CALL check (vector(), 8, 8,-big,-a, flag)
END SUB ' check88
'
'=====================================================================
'
SUB arma1( xn, xnm1, enm1, a, b, stddev)
'
' This subroutine generates random observations from a mixed auto-
' regressive-moving average process of order (1,1), that is,
'                        x[n] = e[n] + b e[n-1] - a x[n-1],
' where the e[i], i=1,2,... are idependently normally distributed
' with mean zero and standard deviation = std.dev.
'
  SHARED PI
  en = SQR(-2*LOG(RND))*SIN(2*PI*RND)*stddev
  xn = en + b*enm1 - a*xnm1
  enm1 = en
END SUB ' arma1
'
```