Variable Sampling Interval Control Strategies For A Process Control Problem

by

Shanthi Sethuraman

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Statistics

APPROVED:

[illegible]

Marion R. Reynolds, Jr., Co-chair

[illegible]

Jesse C. Arnold, Co-chair

[illegible]

Klaus Hinkelmann

[illegible]

Raymond H. Myers

[illegible]

Robert V. Foutz

September 20, 1995

Blacksburg, Virginia

VARIABLE SAMPLING INTERVAL CONTROL STRATEGIES FOR A PROCESS CONTROL PROBLEM

by

Shanthi Sethuraman

Marion R. Reynolds, Jr., Co-chair

Jesse C. Arnold, Co-chair

Statistics

(ABSTRACT)

A process can be monitored for the purpose of detecting and eliminating special causes or for the purpose of adjusting the process to a target value. SPC (Statistical Process Control) methods are used for the purpose of locating and removing any unexpected changes in the quality characteristic. On the other hand, certain processes (manufacturing, chemical etc.) are monitored using APC (Automatic Process Control) methods which compensate for process variability and maintain the process as close as possible to a desired target value. The efficiency of control schemes can be increased by allowing the interval between the samples from the process to vary as a function of the process data. The following are developed for a process control problem using a variable sampling scheme: a model for the process mean, a performance criterion and an estimation technique. The process mean is a random walk model with a control variable. An observation for the process is the mean plus a random error. The Kalman filter estimation technique is used to estimate the time-varying process mean parameter of the process control problem. The objective is to determine the adjustment and sampling strategies that lead to a minimum expected loss. These adjustment and sampling limits address two questions namely, when to adjust and when to take the next sample. The performance of the VSI scheme is compared to the performance of the FSI scheme in terms of the percentage reduction in cost. Also, the effects of the cost combinations and the observation errors on the VSI and FSI are studied.

ACKNOWLEDGMENTS

I wish to express my sincere thanks to my mother Raji Sethuraman for her support and encouragement throughout my years of education and being the major factor in making it possible for me to obtain this degree. I dedicate this dissertation to my mother and in loving memory of my father Professor V. Sethuraman.

I wish to express my appreciation to my advisors Professors Marion R. Reynolds, Jr. and Jesse C. Arnold for their guidance in my research and their continued support, encouragement and patience. I would like to extend my appreciation to the members of my committee, Professors Raymond H. Myers, Klaus Hinkelmann and Robert V. Foutz for their advice and support in my work.

Special thanks to my best friend Mark R. Olin for his enduring support, patience and invaluable comments and suggestions on C-program implementation. Thanks are due to Ernest Crvich and Mark R. Olin for their competent tutelage of the C computer language. I would like to acknowledge Praveen Paripati, Saher Lahouar, Paul Wisnesky, John Sundelin, Scott Harper, and Mark R. Olin for their respective parts in making computer resources available to me.

Loving thanks to my family members Jayashree, Raghunath and Professors Esther and Harvey Olin for their continued support and encouragement throughout this course of study. Finally, thanks to my friend and peer Shankar Srinivasan for enjoyable and interesting discussions on various statistical analyses during the last four years.

Contents

List of Figures

List of Tables

Chapter 1 : Introduction

Many production processes, such as in the chemical and process industries, can be controlled using various control schemes. An important goal of process control is to reduce deviations of some quality characteristic from a desired target value. The goal of reducing deviations can be achieved by SPC (Statistical Process Control) or by APC (Automatic Process Control). APC and SPC have originated as separate and distinct methodologies to improve process productivity and product quality. SPC is principally concerned with the detection and elimination of any unexpected change in the quality characteristic. APC, also known as EPC (Engineering Process Control) is concerned with the compensatory continual adjustment of the process so as to maintain the process as close as possible to the desired target value.

Sometimes it may not be economically feasible or practical to eliminate factors that are beyond our control such as variation in the characteristics of feed stocks, changes in ambient temperature, etc. In such cases compensatory continual adjustments of the process become necessary. APC can be effective in controlling processes when the level of the quality characteristic tends to wander over time. APC methods require that there are control variables that affect the output quality characteristics. The process to be controlled must be known well enough so that the relationship of the output variables to a change in the control variables can be well described by a mathematical model. Compensatory adjustments are made to the control variables so that the deviations of the output quality characteristics from the desired target values are reduced. The adjustments could cause an immediate effect on the output(s) or it may take a while for the effect to be seen on the output(s). The idea is to compensate for the deviations of the system output from a desired target value rather than removal of special causes of variation that give rise to unexpected changes to the quality characteristics.

In traditional SPC methods the standard practice is to take samples at regular intervals of time during the process, say, every 1 time unit. A scheme of this type is called a *fixed sampling interval* (FSI) scheme. A modification of this scheme is to vary the sampling intervals or the time between samples depending on what is observed from the data. This is called a *variable sampling interval* (VSI) scheme. The basic idea of using a VSI scheme is that if there is an indication that the process is operating "in-control", then we may want to wait longer than 1 time unit to take the next sample, say, wait 2 time units. On the other hand if there is an indication of a process change, but the process change is not big enough that the procedure would signal, then we may want to take the next sample sooner than 1 time unit, say, after 0.5 time units. Sampling after 0.5 time units would enable us on average to detect any unexpected changes in the process quicker than usual. Considerable work on SPC control charts using VSI schemes has been done (see literature review).

So far, adjustment strategies have been developed for APC problems using an FSI scheme (see literature review). In this dissertation, we will develop the process control strategies for an APC problem using the VSI scheme. Intuitively it seems that varying the time between samples should lead to better process control than taking samples at regular fixed times. A brief description of the objectives of this dissertation are given below.

(a) <u>develop a model:</u>

The quality characteristic of interest is assumed to be the process mean. The times between samples are assumed to vary as a function of the current sample. Since different VSI schemes use different sampling intervals, a model for the process mean is developed. The process mean is a random walk model with a control variable. An observation for the process is the mean plus a random error.

(b) develop a performance criterion:

The performance criterion is a loss function that is a function of the following three costs:

(i) cost of making an appropriate adjustment (assuming negligible adjustment error)

(ii) cost of sampling

(iii) cost of the process functioning away from target.

(c) develop an estimation technique:

The Kalman filter estimation technique is used to estimate the time-varying process mean parameter when a VSI scheme is applied to the stochastic control problem. This is a recursive estimation procedure that updates the process mean estimate as and when a new observation becomes available. The Kalman filter estimator is a function of the previous process mean estimate, the error variances and the previous sampling interval.

(d) find optimal strategies

After having developed the model, performance criterion and an estimation technique, the process control strategies, namely the adjustment and the sampling strategies that lead to a minimum total expected loss will be determined. The adjustments and sampling strategies address the issues of when to adjust, how much to adjust, and when to take the next sample based on the observed data for the VSI process control problem. The dynamic programming approach is used to obtain the adjustment and sampling limits at each stage of the problem with the objective of minimizing the total expected loss from that stage onward. The adjustment limits will determine the decisions of whether to adjust or not at that stage and the sampling limits determine which sampling interval to use at the next stage. Since it is very difficult to obtain the adjustment and the sampling limits analytically, these limits and the expected loss are determined numerically.

(e) compare VSI vs. FSI

The performance of the VSI scheme is compared to the performance of the FSI scheme for certain cost combinations and observation errors. This comparison is based on the percentage reduction in cost when the VSI scheme is used instead of the FSI scheme. Also, the effects of the cost combinations and the observation errors on the VSI and FSI are studied.

Chapter 2 : Background and Literature Review

2.1 SPC, FSI and VSI Schemes

SPC methods usually employ control charts such as Shewhart charts, CUSUM (Cumulative Sum) charts, and EWMA (Exponentially Weighted Moving Average) charts for the purpose of locating and eliminating any unexpected process changes. Control charts help in determining when the change in the quality characteristic occurred. These charts are then used to bring the process to a basic state of statistical in control without continual adjustments. In traditional SPC techniques, the term in control is defined to be the situation when the observations are identically and independently distributed (i.i.d) with their mean equal to the target ($\mu = \mu_0$) and their variation due only to the common causes at work. When the process mean has shifted to $\mu = \mu_1$, then this is considered as the occurrence of a special cause and thus the process is defined to be out of control.

A control chart is usually maintained using a FSI scheme. A control statistic which is a function of the values from the samples is plotted on a control chart which is divided into regions of in-control and out-of-control by the control limits. When the statistic falls outside the control limits a signal is given indicating a change in the process parameter. This process change is then investigated and the cause if found is eliminated and hence the process is brought to its desired state. If the statistic falls within the control limits, then the process allowed to continue. As an example, consider a yarn manufacturing process. An important quality characteristic is the tenacity of the yarn defined to be the breaking strength with respect to the size of the yarn (measured in denier/meters). Control charts can be employed to monitor the tenacity by sampling at regular time intervals. The control statistic, a function of the observed value of tenacity, is then plotted on the control chart. When the statistic falls outside the control limits it would be taken as an indication of a special cause of variation (may be caused by the improper setup or functioning of the machine that measures the property, or an inexperienced operator). A search is made for the cause of the point outside the control limits, and if a cause can be found then an attempt is made to remove the cause. SPC methods can be viewed as a series of hypothesis testing problems where the null hypothesis states that the process is in a state of in-control and the alternative hypothesis states that the process is not in a state of in-control.

The performance of control charts using the FSI schemes is usually characterized by an important statistical property called the *average run length* (ARL). The ARL is the expected number of samples that is required for a procedure to signal given the present state of the process. It is desirable for the control chart to have a low rate of false alarms (an indication of out-of-control signal when the process is actually operating close to its target value) and to provide prompt signals when the process has shifted from target. When the process is in-control, the ARL should be large so that there is a low false alarm rate. On the other hand, when the process is out-of-control the ARL should be small so that the process disturbances can be detected quickly.

VSI control charts switch between different sampling intervals based on observed data and detect unexpected changes (if any) in the process faster than FSI control charts. Since the sampling rate in VSI control charts depends on the value of the parameter, it is important to look at both the number of samples to signal and the time to signal. The performance of control charts is characterized by the *average number of samples to signal* (ANSS) and *the average time to signal* (ATS). ANSS is the expected number of samples to signal. ATS is the expected time to signal, where the time to signal is the time from the beginning of the process until the chart signals. When the process is in a state of in-control, the ATS should be large and when the process has experienced a shift in its process mean, the ATS should be small.

Considerable work has been done on control charts using VSI schemes. Use of the VSI schemes in control charts has been shown to be more efficient than the FSI schemes in process monitoring. Studies on the properties of Shewhart charts using the VSI schemes and the improved efficiency of using the VSI schemes over the FSI schemes were investigated by Reynolds, Amin, Arnold, and Nachlas (1988), Amin (1987), Reynolds (1989), Runger and Pignatiello (1991), and Cui and Reynolds (1989). Reynolds, Amin, and Arnold (1992) examined the properties and the performance of CUSUM charts using the VSI schemes and it was shown that using the VSI schemes proves to be more efficient than using the FSI schemes. For independent and identically distributed (i.i.d) processes, Reynolds and Arnold (1989) showed that two sampling intervals are optimal for a Shewhart control chart to detect changes in the process parameter when the VSI scheme is used. Saccucci, Amin, and Lucas (1992) and Reynolds (1995) have studied the VSI schemes on EWMA charts. Chengular-Smith, Arnold, and Reynolds (1989) considered simultaneous monitoring of the process mean and the process variance using the VSI scheme and showed that the variable sampling procedures are generally more efficient than fixed sampling procedures. Additional work on variable sampling intervals can be found in papers by Arnold (1970), Crigler and Arnold (1979), Hui and Jensen (1980), Reynolds (1988a, 1988b), Vining and Reynolds (1989), Rendtel (1990), Amin and Letsinger (1991), Amin and Ncube (1991), Shamma et al. (1991), Amin and Hemasinha (1993), Amin and Miller (1993), Runger and Montgomery (1993), Vaughan (1993), Prabhu et al. (1994), and Reynolds (1995).

In SPC it is usually assumed that the process measurements are i.i.d with a distriburion that does not change over time when operating under common causes. But in the context of control charts, some studies on the effect of autocorrelated data in control charts have also been investigated. For example, Goldsmith and Whitfield (1961), Johnson and Bagshaw (1975), Bagshaw and Johnson (1976), and Kartha and Abraham (1979) have studied the effect of positive and negative correlation on the ARL of control charts using a time series model for the process. Fitting time series models to the process and using control charts for the one step ahead forecast residuals to detect a change in the process were examined by Abraham and Kartha (1979), Ermer (1980), Montgomery (1985), and Montgomery and Mastrangelo (1991). Alwan and Roberts (1988) suggested the use of a "special cause chart" to monitor time series residuals and a "common cause chart" to monitor 1-step ahead forecasts. Other studies examining control charts in the presence of correlation were considered by Reynolds et al. (to be published), VanBrackle (1991) and Lu (1994).

2.2 The Kalman Filter

Because the Kalman filter will be used as a tool in the APC problem, we will first give some background on the Kalman filter and then study its application to the optimal control strategies of the APC problem.

A filter is used to extract information about a parameter of interest, using data that is available up to and including a particular time. A filter is actually a system of equations (an estimator) that yields an estimate of the parameter of interest at the current time based upon all past measurements. In this dissertation the parameter of interest is the process mean. Let the process mean at time k be denoted by μ_k. At time k let μ_k be observed with an error ε_k. If observations $x_1, x_2, \ldots, x_k$ are available, where x_i is the observation at time i, then the question that arises is: "Based on these data, what can be inferred about the unknown parameter μ_t, when $t < k$, $t = k$, or $t > k$?". If $t < k$, then the estimation problem is called the *smoothing* problem; if $t = k$, it is called the *filtering* problem; if $t >$

k, it is called the *prediction* problem. Also, the term *state estimation* is applied to all the three problems where the parameter of interest is called the *state*.

Estimation of the process mean when an observation becomes available can be accomplished using the Kalman filter technique, which estimates models with time-varying parameters. Kalman filtering is a recursive estimation procedure that helps in drawing inferences about the unknown process mean parameter (i.e. the problem of estimation is an ongoing one). Many lengthy computations and storage requirements would be needed if one were to process all data at the end of each time period. To overcome these problems, recursive estimation is implemented. Kalman filter recursive estimation uses previous computations (previous estimates) and stores only those items required for the processing of future observations, thus requiring fewer computations and less computer storage than "starting from scratch". The general idea is to update the estimate of the process mean using the previous estimate and the new input data. Since Kalman filtering is a linear discrete-time system of equations, it can be easily implemented on a digital computer.

The Kalman filter theory was developed by Kalman (1960) and Kalman and Bucy (1961). Duncan and Horn (1972) derived the Kalman recursive results for the Kalman filter estimation problem from the regression standpoint. They have presented the relevant random-parameter regression theory as a natural extension of the conventional fixed-parameter regression theory and the derivation of the optimal recursive estimators in terms of the extended regression theory for a typical form of the recursive model. Using a Bayesian approach, Sarris (1973) described the problem of estimating time-varying regression coefficients when a Markov structure is imposed a priori on the coefficients. He explained how the time variation cannot be estimated unless some restrictions are placed on the infinite forms of possible time changes. He also discussed the limitations of some methods that attempt to identify the time-varying coefficients. Harrison and Stevens (1971,1976) derived the Kalman filter using a Bayesian approach to forecasting. The essential foundations of the method include the parametric structure of the state-space model, the probabilistic information on the parameters at any given time that is available in the form of a posterior distribution, and the sequential model definitions for the observation and the parameter. The sequential model definitions describe how the parameters change in time systematically and due to random shocks. For a special model that has time-varying regression coefficients, Sant (1977) applied the generalized least squares method to estimate the coefficients. He further showed the equivalence of Kalman filtering and smoothing techniques to generalized least squares. Ledolter (1979) discussed recursive estimation in linear regression theory using the Kalman filter approach. He also discussed estimation of the parameters recursively when the parameters vary according to an autoregressive integrated moving average (ARIMA) process. Sallas and Harville (1981) extended the Kalman filter recursive estimation technique for fixed and completely random models, to mixed linear models that have autoregressive random effects. Authors such as MacGregor (1973), MacGregor and Wong(1980), and Phadke (1981) have used the Kalman filter as an estimation technique in applications such as designing optimal (univariate and multivariate) stochastic control algorithms for chemical processes, nuclear material control, and quality auditing.

2.3 Automatic Process Control

APC methods employ feedback, feedforward, or a combination of feedback and feedforward systems to adjust the control variables of the process. In feedback control, the deviations of the quality characteristics from the target values are used to calculate the appropriate compensations to be made to the process by inducing appropriate changes to the control variables. This has a feedback path between the observed deviations of the output from the target values and the control variables. For example, in a yarn manufacturing process a high or low viscosity y_t of the polymer can be compensated by adjusting the flow rate F_t of one of the ingredients at the input. The deviation of y_t from a desired target value T is used to decide the appropriate adjustments to be made on the flow rate F_t. In feedforward control, it is assumed that the effect of some measured but uncontrolled variable on the output variable is known. The control variable is altered based on this measured variable rather than on the deviations of the output from the target. For example, in a yarn manufacturing process the measured molecular weight M_t of one of the ingredients used to make the polymer is known to affect the viscosity of the polymer. Instead of the deviation of y_t from a desired target value T, M_t(molecular weight) can be used to determine the appropriate F_t (flow rate) which is the control variable that can be manipulated. The flow rate F_t is used to compensate for the deviations y_t-T, by compensating for the fluctuations in the molecular weight M_t.

Extensive work has been done on control procedures in control theory and one can find papers as far back as 1952 (Girshick and Rubin). Some of the other authors who have explored the field of feedback/feedforward control systems are Bishop (1957, 1960), Box and Jenkins (1962, 1963, 1970), Åström (1970), Åström and Wittenmark (1984), Box and MacGregor (1974), Abraham and Box (1979), Kelly et al. (1987), and MacGregor (1991).

Many studies have investigated control strategies for control theory problems with discrete time systems using the FSI scheme where samples are taken at equidistant time intervals. Åström (1970) gives optimal control strategies for a general model in control theory given by the

observation equation $$x_k = M_k\mu_k + \varepsilon_k \tag{2.3.1}$$

and

system equation $\mu_k = P_k\mu_{k-1}+A_k\mathbf{a}_{k-1}+\nu_k\,;\;\; k = 1, 2, 3, \ldots\ldots$ (2.3.2)

where

$\mathbf{x}_k$ is the r x 1 observation vector,

$\mathbf{a}_k$ is the p x 1 vector of control variables that may depend on the $\mathbf{x}_k$'s,

μ_k is the q x 1 state vector,

M_k is a r x q matrix,

P_k is a q x q matrix,

A_k is a q x p matrix,

and $\{\varepsilon_k, k = 1,2,3,...\}$ and $\{\nu_k, k = 1,2,3,...\}$ are sequences of independent multivariate normal random variables with mean zero and variance-covariance matrices E and N, respectively. The matrices M_k, P_k, A_k, E, and N may depend on time k and are assumed to be known. It is also assumed that the initial state μ_0 is multivariate normal with mean M_k and variance-covariance matrix **M**.

The performance of the system is characterized by the following expected loss function

$$EL(n) = E\left\{\sum_{k=1}^{n}\left(\mu_k^T Q_1 \mu_k + \mathbf{a}_{k-1}^T Q_2 \mathbf{a}_{k-1}\right)\right\} \tag{2.3.3}$$

The matrices Q_1 and Q_2 with dimensions (q x q) may also depend on time k. Q_1 is symmetric and nonnegative definite and Q_2 is positive definite.

The general control problem as described in Åström (1970) is to find a control strategy for the system specified by (2.3.1) and (2.3.2) such that (2.3.3) is minimized. The infinite stage process is considered as an approximation to the n-stage problem. The technique of dynamic programming is used to solve the control problem. The control problem can be thought of as consisting of a sequence of problems, each of which requires some variables to be determined. Solving a single n-dimension problem can turn out to be very cumbersome and time-consuming. On the other hand, dynamic programming uses the backward induction method to solve n single dimension problems, instead of solving a single n-dimension problem. The method of dynamic programming was developed by Bellman (1957, 1967), Bellman and Dreyfus (1962), and Bellman and Kalaba (1965). The components of a dynamic programming problem have been described by Cooper and Cooper (1981) as follows:

> Problems to which dynamic programming has been applied are usually stated in the following terms. A physical, operational, or conceptual system is considered to progress through a series of consecutive *stages*. At each stage the system can be described or characterized by a relatively small set of parameters called the *state variables* or *state vector* (state for short). At each stage, and no matter what state the system is in, one or more *decisions* must be made. These decisions may depend on either stage or state or both. It is also true that the past history of the system, i.e., how it got to the current stage and state, are of no importance. When a decision is made a *return* or reward is obtained and the system undergoes a *transformation* or *transition* to the next stage. The return is determined by a known single-valued function of the input state. Similarly, the transformation state results from a known single-valued function of the decision acting upon the current state. The overall purpose of the staged process is to maximize or minimize some function of the state and decision variables.

Bellman's *principle of optimality* is essential to be able to carry out the above mentioned transformation of the n-dimension problem to n single-dimension problems. As stated by Bellman (1957) the *principle of optimality* is:

> An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with respect to the state which results from the initial decision.

In addition to the stages, random state variables, decisions, transitions, and a cost function, a stochastic dynamic programming problem is further described by probability distributions and a probabilistic operator (which is the expected value in equation 2.3.3). The dynamic programming procedure consists of deriving a set of functional equations from the criterion function (for example, equation 2.3.3) and solving the equations to get the optimal decisions. This is stated in Cooper and Cooper (1981):

"The underlying similarity of all dynamic programming processes is the creation of a set of functional equations of a particular type called recurrence relations".

If the observation equation of the system (2.3.1) is specified under $M_k = I_k$ (unit matrix) and $\varepsilon_k = 0$, for k = 1,2,3, ..., then the observation $\mathbf{x}_k$ at any time k gives the exact value of the state vector μ_k. This is described as "the complete state information problem". On the other hand, when the observation equation of the system is specified by (2.3.1), in the case in which the state vector is not known exactly implying that ε_k has positive variance, it is described as "the incomplete state information problem".

A special case of (2.3.1) and (2.3.2) is a scalar linear system that is specified by the

observation equation $$x_k = \mu_k+\varepsilon_k \tag{2.3.4}$$

and the

process mean equation $$\mu_k = \mu_{k-1}+a_{k-1}+v_k \; ; \; k = 1, 2, 3, ..., n, \tag{2.3.5}$$

where x_k , μ_k , ε_k , v_k , and a_k are scalars. The sequences $\{\varepsilon_k\}$ and $\{v_k\}$ are assumed to be independent normal variables with mean zero and known variances σ_ε^2 and σ_v^2, respectively. The state variable is assumed to be the process mean. If $a_k \equiv 0$ then (2.3.5) is a random walk model. It is also assumed that the adjustments made to the process have an immediate effect on the process. Now, let c_1 denote the cost per squared deviation of the process from the zero target and c_2 denote the cost proportional to any size adjustment made to the process. Let $r = c_2/c_1$ be the cost ratio. Then the expected loss function (2.3.3) for the above n-stage problem becomes

$$EL(n) = c_1 E\left\{ \sum_{k=1}^{n} \left(\mu_k^2 + r a_{k-1}^2\right) \right\}. \tag{2.3.6}$$

The model is assumed to be initialized with the prior distribution $\mu_0 | x_0 \sim N(\hat{\mu}_0, q_0)$, where x_0 is the past history of the x_k process. Applying the results from Åström (1970) to the above special case and using the dynamic programming approach, the optimal adjustment that minimizes (2.3.6) is

$$a_k = -F_k \hat{\mu}_k \tag{2.3.7}$$

where $\hat{\mu}_k = E(\mu_k | x_0, x_1, ..., x_k)$ is the generalized Kalman filter estimate of μ_k given all the data through k, and

$$F_k = \frac{w(k+1)}{r+w(k+1)}.$$

The w(.) are defined recursively as follows:

$$w(n) = 1$$

and

$$w(k) = \frac{[1+r]w(k+1)+r}{r+w(k+1)}, \text{ for } k = 1, 2, 3, ..., n-1.$$

Here an adjustment of $-F_k \hat{\mu}_k$ is made at every time period k. We notice that F_k is less than or equal to 1.

Crowder (1986) studied the adjustment strategies for the model specified by (2.3.4) and (2.3.5) using the FSI scheme. The assumptions in Crowder's model are the same as those in Åström's model, but Crowder used a different performance criterion, given by

$$EL(n) = c_1 E\left\{ \sum_{k=1}^{n} \left(\mu_k^2 + r\delta(a_{k-1})\right) \right\}, \tag{2.3.8}$$

where

$$\delta(a) = \begin{cases} 0 & \text{if } a = 0 \\ 1 & \text{if } a \neq 0. \end{cases} \tag{2.3.9}$$

Any cost of sampling is considered irrelevant. The off-target cost and the adjustment cost are assumed to be the only costs that affect the decision regarding when to adjust and when not to adjust. While (2.3.6) assumes that an adjustment is made at every time period k, (2.3.8) assumes that an adjustment is made only when there is evidence that the process has wandered "far away" from the target. Crowder (1986) used the technique of dynamic programming to obtain the adjustment limits for the system specified by (2.3.4) and (2.3.5) such that (2.3.8) is minimal. He derived $\hat{\mu}_k$, the Kalman filter estimate which is given by

$$\hat{\mu}_k = \hat{\mu}_{k-1}+a_{k-1}+\left[\frac{q+\sigma_v^2}{q+\sigma_v^2+\sigma_\varepsilon^2}\right]\left(x_k-(\hat{\mu}_{k-1}+a_{k-1})\right), \tag{2.3.10}$$

where q is the limiting value of q_k, that is, q is the steady state variance of $\mu_k|x_0,x_1,\ldots,x_k$, and has been proved by Crowder to be equal to $\frac{-\sigma_v^2+\sqrt{\sigma_v^4+4\sigma_\varepsilon^2\sigma_v^2}}{2}$. Crowder then derived a set of two functional equations from the criterion function (2.3.8) and solved these functional equations to determine the optimal decisions. One functional equation is the minimum expected loss for a 1-period problem; the other equation is the minimum expected loss for a k-period problem obtained from a (k-1)-period problem. These minimums were taken over all possible actions that could be taken. The optimal adjustment limits that determine "when to adjust" and " when not to adjust" were obtained from the functional equations. The optimal adjustment limits were approximated numerically using the quadrature approximation approach. These limits depend on r the cost ratio, σ_v^2 the system variance, and n the length of the control problem. Crowder further showed that the adjustment limits have a convergent or limiting value as $k \to \infty$. The limiting optimal adjustment limits can be used for problems involving a large number of stages. He also computed the limiting adjustment limits for various values of r, the cost ratio of the adjustment cost to that of the off-target cost.

Jensen (1989) derived the control strategies based on a Kalman filter estimate of the process mean for the model specified in Crowder (1986) with the addition of a deterministic drift term and an adjustment error term such that a quadratic loss function is minimized. Here the adjustment error refers to the error in the automatic controller that monitors and adjusts the process. This error arises as a result of the inability of the controller to make perfect adjustments. The performance criterion used by Jensen is similar to that used by Crowder (1986). Jensen (1989) also used the technique of dynamic programming to determine the adjustment strategies. The adjustment limits were shown to depend on the deterministic drift, the various costs involved, and the number of stages in the control problem. For the case when there is no drift and no adjustment error, Jensen's results agreed with Crowder's results. She further discussed the effects of the various costs involved, the adjustment error variance, the system error variance, the observation error variance, and the drift on the adjustment limits. Using simulation, she concluded that the adjustment limits converge to a limiting value as the number of stages increases.

For the model described by the random walk model for the process mean, Adams and Woodall (1989) compared the work by Box and Jenkins (1963), Crowder (1986), Taguchi et. al. (1989), and Jensen (1989). By including a measurement cost in the economic model of Box and Jenkins (1963) another economic model is described in Adams and Woodall (1989). Approximate optimal control strategies and sampling intervals were obtained for the new economic model. Adams and Woodall (1989) discussed the limitations of Taguchi's model given in Taguchi et. al. (1989) and Taguchi (1985), which assumes no measurement error and includes a non zero cost of inspection of an item.

Optimal sampling frequency and adjustment limits when the cost of taking an observation is non negligible and when the process is sampled at some fixed multiple of the unit interval was investigated by Box and Kramer (1989). They used a time series model for the disturbances and one step ahead predicted deviation from target to decide when to make adjustments to the process in order to operate it as close as possible to the desired target value. Schemes designed to minimize the overall cost were developed. They studied the effect of the different values for the disturbance model parameter on the optimal adjustment limits. Alternatively, Box and Luceño (1994) investigated schemes that avoid the direct assignment of values to the costs of making an adjustment, of taking a sample and that of being off-target. These schemes are characterized by the mean squared deviation from target produced, the frequency with which the observations are required to be made, and the resulting overall length of time between adjustments. The schemes are calculated making specific assumptions concerning the dynamic and disturbance models affecting the process. Comparisons with Taguchi's (1984, 1986) results are also shown. Vander Wiel et. al. (1992) have shown that APC methods could be applied in conjunction with SPC methods leading to process optimization and process improvement. They developed an integrated approach of process adjustments and elimination of special causes of variability leading to quality improvement.

Chapter 3: Analysis of the Random Walk Case

3.1 Introduction

In this chapter we consider the problem of determining the process control strategies, namely the adjustment and the sampling strategies, for a process control problem model with a VSI scheme. In Section 3.2 the description of the process control model will be given. The model is specified by the observation and the process mean equations. The process mean is assumed to follow a random walk model with a control variable. Samples are assumed to be taken from the process based on a VSI scheme. The structure of a performance criterion will be given and based on the performance criterion the process control strategies will be derived. In Section 3.3, a Kalman filter estimator for the process mean will be derived using the Kalman filtering theory. The conditional distribution of the process mean given information available up to stage k and the conditional distribution of the Kalman filter estimator given information available up to stage k-1 will also be derived. These conditional distributions will be used in deriving the optimal adjustment and sampling strategies. In Section 3.4, the properties of the conditional variances of the above mentioned conditional distributions will be studied. In Section 3.5, an explanation of how the dynamic programming approach is used to determine a set of functional recursive expected loss equations for a VSI scheme with two sampling intervals, is given. Further, the set of functional equations, namely the expected loss functions based on the performance criterion will be derived. Also, the questions of how the adjustment and sampling limits are determined from the functional equations with the goal of minimizing the expected loss will be addressed. The properties in Section 3.4 will be used in the computation of the expected loss functions. The roles of the adjustment limits, sampling limits, and the Kalman filter estimator, as part of the control strategies will be explained.

3.2 Model and Optimization Criterion

Crowder (1986) used the process model consisting of the observation and process mean equations given by (2.3.4) and (2.3.5) respectively. He assumed that samples are taken from the process at each of the n possible times, namely $k = 1,2,\ldots,n$. These n times are assumed to be 1 unit apart. The n possible times correspond to the n stages in the problem. Throughout this thesis, we will refer to the model specified by (2.3.4) and (2.3.5) with the FSI scheme of 1 unit as Crowder's model.

Similar to that of Crowder's model, our model consists of two equations, an observation equation and a process mean equation. We shall refer to our model as the VSI process control model. In the VSI process control model, samples are not necessarily taken from the process at equidistant time intervals; the time to the next sample depends on the observed data. Let $d_1, d_2, \ldots, d_s$ be the various possible sampling interval lengths that can be used where $d_1 < d_2 < \ldots < d_s$. Let d_1 denote the base interval and d_i, $i = 2, 3, \ldots, s$ be an integer multiple of the base interval. In practice, d_1 can be taken to be the least amount of time required to take a sample and d_s can be taken to be the maximum amount of time the process is allowed to run before the next sample can be taken. Suppose that it is desired to implement a VSI scheme beginning immediately, for the next t' time units. Let the random times at which we sample be denoted by $t_1, t_2, \ldots, t_{n'}$. Let t_0 be the starting time for the VSI process control problem at which no sample is yet available and let $t_{n'+1}$ be equal to t'. Let us assume that no sample will be taken at the very last time, namely $t_{n'+1}$. The justification for this is that since there are no decisions to be made with regard to sampling/adjustment after time t', there is no point in taking a sample at the very end. At any time t_j, the next sampling time t_{j+1} is determined based on the observed data at time t_j. The distance between time t_j and time t_{j+1} could be any of the sampling intervals d_i, $i = 1, 2, 3, \ldots, s$.

The observation equation is given by

$$x_{t_{j+1}} = \mu_{t_{j+1}} + \varepsilon_{t_{j+1}}, \qquad (3.2.1)$$

where $x_{t_{j+1}}$ is the observation taken at time t_{j+1}, $\mu_{t_{j+1}}$ is the process mean at time t_{j+1} and $\varepsilon_{t_{j+1}}$ is the random error at time t_{j+1} for j = 0, 1, ..., n' and $x_{t_{n'+1}}$ is not observed.

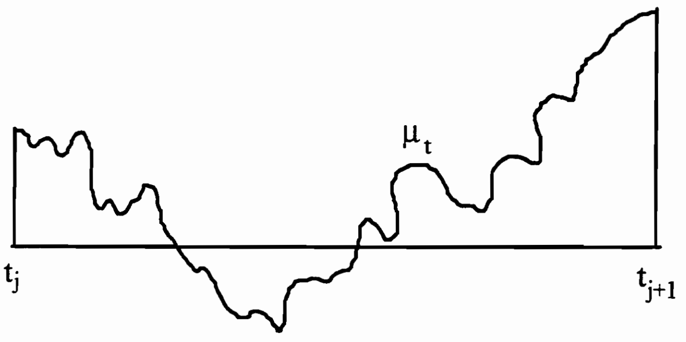


Figure 3.2.1

Equation (3.2.1) is similar to (2.3.4) except that (3.2.1) is expressed in terms of the sampling times. In the VSI process control problem we will generalize the process mean equation (2.3.5) in order to incorporate the varying sampling interval lengths. In Figure 3.2.1, let μ_t represent the process mean at time $t \in (t_j, t_{j+1}]$ where t_j and t_{j+1} represent any two times. Let the model for the process mean be given by

$$\mu_t = \mu_{t_j} + a_{t_j} + v(t - t_j) \qquad (3.2.2)$$

where a_{t_j} is the adjustment at time t_j and the system error $v(t-t_j)$ has zero mean and variance $(t-t_j)\sigma_v^2$ with σ_v^2 known. The process system error variance depends on the sampling interval length. The longer the sampling interval length, the larger the system error variance. We note that Crowder's process mean equation (2.3.5) is a special case of (3.2.2). If we let t_j and t_{j+1} be the times associated with stages k-1 and k respectively and use a FSI scheme of one unit which implies that the sampling times t_j, j = 0, 1, 2, ..., n' are each one unit apart, then we see that at stage k equation (3.2.2) is the same as that of

(2.3.5). Next, regarding the assumptions on VSI process control model we shall retain the assumption from Crowder (1986) that any adjustment to the process has an immediate effect on the process and also retain the assumptions on the model error terms. We will assume that the sample size = 1. For a sample size > 1, everything in this chapter follows through with σ_ε^2 replaced by (σ_ε^2/sample size) on the assumptions that the observations in the same sample are taken at the same time and have the same mean and that the observations are independent conditioned on the mean.

Since we will be referring to Crowder's performance criterion in order to explain the difference between Crowder's performance criterion and the performance criterion for the VSI process control model, we will rewrite Crowder's performance criterion (equation 2.3.8) as follows

$$EL(n) = E\left\{ \sum_{k=1}^{n} \left(c_1\mu_k^2 + c_2\delta(a_{k-1}) \right) \right\}. \tag{3.2.3}$$

The performance criterion for the VSI process control model is an extended version of Crowder's performance criterion. The loss function for the VSI process control model consists of three costs, namely the integrated off-target cost, an adjustment cost and a sampling cost. As in Crowder's performance criterion (3.2.3), let c_2 denote the fixed cost of making any size adjustment and $\delta(a)$ denote the indicator function given by equation (2.3.9) in our performance criterion. As stated earlier $\delta(a)$ takes the value of 1 if and only if a nonzero adjustment is made to the process.

There are two main differences between the structure of Crowder's performance criterion and the performance criterion for the VSI process control problem. One difference is with respect to the sampling cost. In Crowder's performance criterion (equation 3.2.3) there was no need to incorporate a sampling cost since samples were always taken at each possible sampling point and thus the sampling cost is not relevant to determining the optimal adjustment decisions. In the case of the VSI process control model, since different schemes will take different number of samples, there is a need to incorporate a sampling cost in determining the optimal sampling and adjustment strategies. Let c_3 denote the cost of taking a sample at any stage k.

The second difference is with regard to the way the off-target cost is defined. In the VSI process control problem, since different schemes will sample at different intervals we will define a more general measure of the off-target cost than the one given in (3.2.3). We will refer to this general measure of the off-target cost as the integrated off-target cost. The integrated off-target cost between any two sampled points will account for all deviations between the sampled points, unlike the off-target cost in (3.2.3) which evaluates the off-target costs for the process deviations only at the sampled points. Let $c_1 g(t_j, t_{j+1})$ denote the integrated off-target cost function from t_j to t_{j+1} in Figure 3.2.2 where $g(t_j, t_{j+1})$ be defined as

$$g(t_j, t_{j+1}) = \int_{t_j}^{t_{j+1}} \mu_t^2 dt \,.$$

Let c_1 denote the cost per squared unit deviation of the process mean μ_t at any time t from the zero target. The c_1 used in the performance criterion of the VSI process control

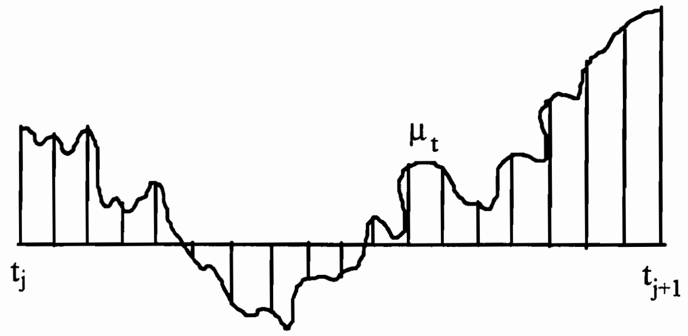


Figure 3.2.2

model is based on the integrated off-target cost. Let the vector $\mathbf{x}_{t_j}$ denote the past information up to time t_j. The past information up to time t_j represents all the observations available up to time t_j. Crowder's equation (3.2.3) evaluates the off-target costs of the deviations of the process from the zero target at the sampled points without directly accounting for the deviations between the sampled points. For example, let t_j and t_{j+1} in Figure 3.2.2 represent times associated with stages k-1 and k respectively. Then, writing Crowder's expected off-target cost in terms of the times t_j and t_{j+1}, we see that the expected off-target cost conditioned on the information at t_j, namely $E[c_1\mu^2_{t_{j+1}}|\mathbf{x}_{t_j}]$ approximates all deviations in the interval from t_j to t_{j+1} with the deviation at the end of the interval. On the other hand, the expected integrated off-target cost from t_j to t_{j+1} conditioned on the information at t_j, $E[c_1 g(t_j,t_{j+1})|\mathbf{x}_{t_j}]$ is given by

$$E[c_1 g(t_j,t_{j+1})|\mathbf{x}_{t_j}] = \int_{t_j}^{t_{j+1}} E\{c_1\mu_t^2|\mathbf{x}_{t_j}\}\,dt. \tag{3.2.4}$$

The above integral accounts for all the deviations of the process from the zero target between t_j to t_{j+1}. By accounting for all deviations between the ends of any two successive sampling points, one can determine the expected loss when the sampling interval lengths d_i ($i = 1,2,\ldots,s$) are not necessarily integers. For example, let's say that a VSI scheme with two sampling intervals d_1 and d_2 is used. Then using the integrated off-target cost one can determine the expected loss for say, $d_1 = 0.6$ and $d_2 = 1.8$. Therefore the expected loss calculations are not restricted to only integer values of d_1 and d_2 and to equidistant sampling interval lengths.

Let us now derive the expression for $E[c_1 g(t_j,t_{j+1})|\mathbf{x}_{t_j}]$. Let us assume that $\mu_{t_j}|x_{t_j} \sim N(\hat{\mu}_{t_j}, q)$, with q known. Then the conditional distribution of the process mean at any time $t \in (t_j, t_{j+1}]$ is given by

$$\mu_t|\mathbf{x}_{t_j} \sim N(\hat{\mu}_{t_j} + a_{t_j}, q + \sigma_v^2(t - t_j)).$$

Using the result $\mathrm{var}\{x(t)\} + [E\{x(t)\}]^2 = E\{x^2(t)\}$ in (3.2.4) we have,

$$\int_{t_j}^{t_{j+1}} E\{\mu_t^2 | \mathbf{x}_{t_j}\} dt = \int_{t_j}^{t_{j+1}} \mathrm{var}\{\mu_t | \mathbf{x}_{t_j}\} dt + \int_{t_j}^{t_{j+1}} [E\{\mu_t | \mathbf{x}_{t_j}\}]^2 dt$$
$$= \int_{t_j}^{t_{j+1}} [q + (t - t_j)\sigma_v^2] dt + \int_{t_j}^{t_{j+1}} [\hat{\mu}_{t_j} + a_{t_j}]^2 dt$$
$$= qd + \frac{\sigma_v^2 d^2}{2} + (\hat{\mu}_{t_j} + a_{t_j})^2 d, \tag{3.2.5}$$

where d is the distance between time t_j and time t_{j+1}. In the above expression that represents the integrated off-target loss from t_j to t_{j+1}, the first term on the right hand side represents the penalty due to the variability in the estimator $\hat{\mu}_{t_j}$, the second term represents the loss due to the inherent nature of the process to drift, and the last term is the estimated loss due to being away from target at time t_j. The integrated off-target cost (equation (3.2.5)) is a quadratic function in d unlike Crowder's off-target cost which is a linear function in d. Now, the performance criterion for the VSI process control problem is the following expected loss function

$$EL = E\left\{ \sum_{j=0}^{n'} \left[c_1 g(t_j, t_{j+1}) + c_2 \delta(a_{t_j}) \right] + c_3 n' \right\}. \tag{3.2.6}$$

Since we will be working on the VSI process control problem in terms of stages, we will define what we mean by stages and describe its relation to the sampling time t_j. Let the term stage be defined based on Cooper and Cooper's (1981) definition. Stages refer to a series of consecutive points through which the system progresses, starting from time t_0 up to time t'. Suppose the VSI process control problem has n+1 stages that consists of stage 0 which is associated with the start-up time t_0, stages 1, 2, ..., n-1 which are equidistant and where samples may or may not be taken, and stage n where no sample will be taken. Then the sampling time t_j is the time associated with a stage at which a sample is taken. Figure 3.2.3 shows the correspondence between stages and sampling times in the VSI process control problem. The stages are denoted by 0,1,2,...,n which are d_1 units

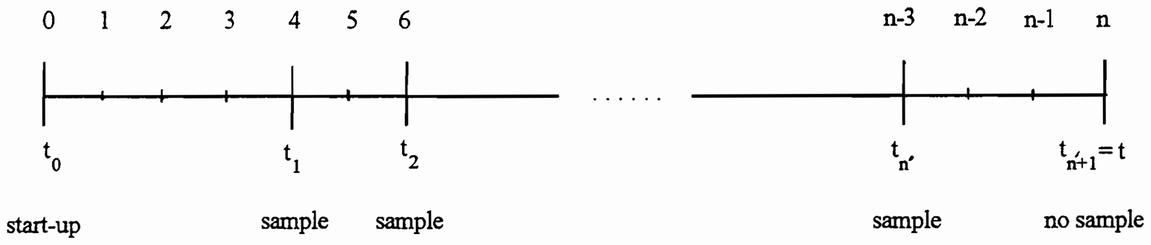


Figure 3.2.3

Relation between stages and sampling times

apart except for the last one which may or may not be d_1 units from stage n-1 and the random sampling times are denoted by $t_1, t_2, \ldots, t_{n'}$. In an n+1 stage VSI process control problem, n is determined by t'/d_1. We see that t'/d_1 may or may not be an integer. If t'/d_1 is not an integer, then n is taken to be the integer part of t'/d_1 plus one. This would lead to n-1 equal time lengths or periods and one unequal time length. For convenience let us assume that the last period represents the unequal period.

3.3 Derivation of a Kalman filter estimator

In this section, we will show that the process mean at stage k, μ_k of the VSI process control model can be estimated by a Kalman filter. The derivation given here uses a Bayesian approach to estimate μ_k given available data as stage k progresses. Crowder (1986) derived a Kalman filter estimator for the process mean in his model using a Bayesian approach with the assumption that at stage 0 prior to taking any sample, $\mu_0|x_0$ is normally distributed with mean $\hat{\mu}_0$ and variance known and x_0 denoting the past history of the process. He showed that the Kalman filter estimator is the mean of the posterior distribution of μ_k given information available up to and including stage k. The Kalman filter estimator is the Bayes estimator under squared error loss and thus is the minimum mean squared error estimate. Jensen(1989) derived a Kalman filter estimator for the process mean in Crowder's model with the addition of a deterministic drift and an adjustment error term. In both Crowder's and Jensen's model, observations are assumed to be available at each possible stage. So at any stage k there is only one expression to the Kalman filter estimator for the process mean based on observations available at every stage up to and including stage k. On the other hand observations at every stage may not be available when using a VSI scheme. As a result we will no longer have a single expression for the Kalman filter estimator at stage k. At any stage k the Kalman filter estimator will depend on how far back the last observation was available. For example, in the case of using the VSI scheme with d_1 and d_2 as the two sampling intervals, the Kalman filter estimator at stage k can take one of two expressions depending on whether the last observation available was d_1 or d_2 units back. In this case the two expressions for the Kalman filter estimator will have to be derived. Sorenson (1966) has shown that the Kalman filter estimator for the process mean when observations at every stage is available up to and including stage k is unbiased under the initial conditions, namely $E[\mu_0|x_0]=\hat{\mu}_0$ and variance$(\mu_0|x_0)=v_0$ with $\hat{\mu}_0$ and v_0 specified. One may easily extend Sorenson's proof for unbiasedness to show that our Kalman filter estimator for the process mean of the VSI process control problem, based on available observations up to and including stage k, is unbiased.

Given an arbitrary data sequence $(z_0, \ldots, z_k)$ up to stage k which means that this sequence of data need not be equally spaced, one can draw inference about μ_k by directly applying Bayes theorem to get

$$f(\mu_k|z_0, \ldots, z_k) \propto f(x_k|\mu_k, z_0, \ldots, z_t) * f(\mu_k|z_0, \ldots, z_t)$$

where z_t is the last observation available before stage k $(t < k)$, $f(\mu_k|z_0, \ldots, z_{k-1})$ is the p.d.f of μ_k prior to the observation z_k and thus denotes the prior distribution for μ_k, $f(z_k|\mu_k, z_0, \ldots, z_t)$ denotes the likelihood, and $f(\mu_k|z_0, \ldots, z_k)$ is the p.d.f of μ_k after the observation z_k has been taken and denotes the posterior of μ_k. The calculation of the posterior distribution of μ_k using Bayes theorem directly is very tedious. The derivation that is given in Theorem 1 of Appendix A is an easier approach to obtain the posterior distribution of μ_k and involves using the some well-known results from multivariate analysis:

As the Kalman filter technique is an iterative updating procedure, it is necessary to assume starting values in order to initialize this recursive procedure. There are various issues that affect the initialization of the procedure. These issues deal with how much past information is available at the start of the VSI process control problem and how far back this information was obtained. The past information has a significant influence upon the current and future calculations. Suppose that a FSI scheme using intervals of 1 unit has been in effect for a substantial amount of time on a process that follows Crowder's model. Let x_0 denote the past history of this process namely, a vector of observations up to stage 0. This past history is assumed to represent a substantial number of observations from the above process taken at stages of 1 unit distance up to and including stage 0. Assume μ_0 given x_0 to be normally distributed with given mean $\hat{\mu}_0$, the best estimate of μ_0 and variance v_0, the steady-state or the limiting variance of Crowder's process model that uses a FSI scheme of 1 unit. The distribution of μ_0 can be interpreted as summarizing one's knowledge about μ_0 based on the past history of the process. The distribution of μ_0 given x_0 which is the posterior for μ_0 will be used to obtain the prior for μ_1. The adjectives prior and posterior are relative terms relating to the observed sample. In general, if after observing a sample at stage k and obtaining the posterior distribution of μ_k one takes a sample at the next stage, the posterior of μ_k would be used to obtain the prior distribution for μ_{k+1}, which is the prior distribution relative to the new sample at stage k+1. Then this prior distribution is used to find the posterior of μ_{k+1}. Regarding the variance v_0, using results from Crowder (1986) it can be shown that

$$v_0 = 0.5*\left[-\sigma_v^2 + \sqrt{(\sigma_v^4 + 4\sigma_\varepsilon^2\sigma_v^2}\right], \tag{3.3.1}$$

where σ_ε^2 and σ_v^2 represent the observation and system error variances respectively.

Let s(k) denote the last stage before stage k at which a sample was taken. If no previous sample was taken at stage k, then s(k) will be zero. Let d(k) denote the sampling interval used before stage k. We see that d(k) must be one of the values $d_1, d_2, ..., d_s$. If k = 1, then d(k) must be d_1. Before we derive the general expression to the Kalman filter estimator of the process mean for the VSI process control problem, we will define the term $\gamma(h)x_h$ for h = 1, 2, ..., n. If the process is observed at stage h, then $\gamma(h)x_h = x_h$. If the process is not observed at stage h, then the term $\gamma(h)x_h$ is dropped from the sequence of data. For example, in (3.3.2) below, if the process was observed at stage 1, then the sequence would be $(x_0, x_1, \gamma(2)x_2, ..., x_{s(k)}, x_k)$. If the process is not observed at stage 1, then the sequence would be $(x_0, \gamma(2)x_2, ..., ..., x_{s(k)}, x_k)$. Now from Theorem 1 of Appendix A, the general expression of the Kalman filter estimator at stage k is given by the mean of the posterior distribution

$$\mu_k | x_0, \gamma(1)x_1, ..., \gamma(h)x_h, .., x_{s(k)}, x_k \sim N[\hat{\mu}_k, v_k], \tag{3.3.2}$$

where

$$\hat{\mu}_k = (\hat{\mu}_{s(k)} + a_{s(k)}) + k_{d(k),k}(x_k - (\hat{\mu}_{s(k)} + a_{s(k)})) \tag{3.3.3}$$

and

$$k_{d(k),k} = \frac{v_{s(k)} + d(k)\sigma_v^2}{v_{s(k)} + d(k)\sigma_v^2 + \sigma_\varepsilon^2},$$

$$v_k = k_{d(k),k}\,\sigma_\varepsilon^2. \tag{3.3.4}$$

Also, we see that from Theorem 1 of Appendix A, the conditional distribution of the Kalman filter $\hat{\mu}_k$ given information available up to stage s(k), which will be later used in deriving the optimal adjustment and sampling strategies, is given by

$$\hat{\mu}_k | x_0, \gamma(1)x_1, ..., \gamma(h)x_h, .., x_{s(k)} \sim N[M_k, u_k],$$

where $M_k = \hat{\mu}_{s(k)} + a_{s(k)}$ and the conditional variance u_k is given by

$$u_k = \frac{(v_{s(k)} + d(k)\sigma_v^2)^2}{v_{s(k)} + d(k)\sigma_v^2 + \sigma_\varepsilon^2} \tag{3.3.5}$$
$$= k_{d(k),k}(v_{s(k)} + d(k)\sigma_v^2).$$

As stated earlier, we have assumed that the sample size is 1. If the sample size is > 1, then the Kalman filter estimator is easily obtained by replacing x_k by $\bar{x}_k$ in its expression and replacing σ_ε^2 by (σ_ε^2/sample size) in the expressions of the conditional variances v_k and u_k. Since σ_ε^2 and σ_v^2 are assumed to be known, the posterior mean $\hat{\mu}_k$ depends on the data only through the sample means.

3.4 Properties of the conditional variance sequences - $\{v_k\}$ and $\{u_k\}$

In this section, let a sequence be denoted by curly braces { } and a set be denoted by square brackets [ ]. Let $[v_j]_k$ denote the set of v_j's for a given stage k, $[u_j]_k$ denote the set of u_j's for a given stage k, $\{v_k\}$ denote the sequence of v's for a given sampling pattern starting from stage 0 and $\{u_k\}$ denote the sequence of u's for a given sampling pattern starting from stage 0. Elements in the sets $[v_j]_k$ and $[u_j]_k$ are represented by v_{k,j,d_i} and u_{k,j,d_i}. The subscripts in v_{k,j,d_i} and u_{k,j,d_i}, namely k, j, d_i, i = 1,2,...,s represent the following: k is the stage, j is the jth element in the sets $[v_j]_k$ and $[u_j]_k$, and d_i is the last sampling interval length used at stage k. The properties of the set $[u_j]_k$ and sequence $\{u_k\}$ are similar to those of $[v_j]_k$ and $\{v_k\}$ respectively. The properties of $[v_j]_k$ and $\{v_k\}$ will be derived in detail and then the properties of $[u_j]_k$ and $\{u_k\}$ will be given without derivation. A detailed example represented by Figure 3.4.1 (next page) will help understand how $[v_j]_k$ and $\{v_k\}$ are formed. In this example let s=3 which means that a VSI scheme with d_1, d_2, and d_3 as the possible sampling intervals is used. Let $d_2 = 2*d_1$ and $d_3 = 3*d_1$. The starting variance is v_0. At stage 1 the set $[v_1]_1$ consists of 1 element, i.e., $[v_1]_1 = (v_{1,1,d_1})$. This element denotes the conditional variance obtained at stage 1 on using d_1 as the last sampling interval at stage 1. The expression for $v_{1,1,d_1}$ is given by

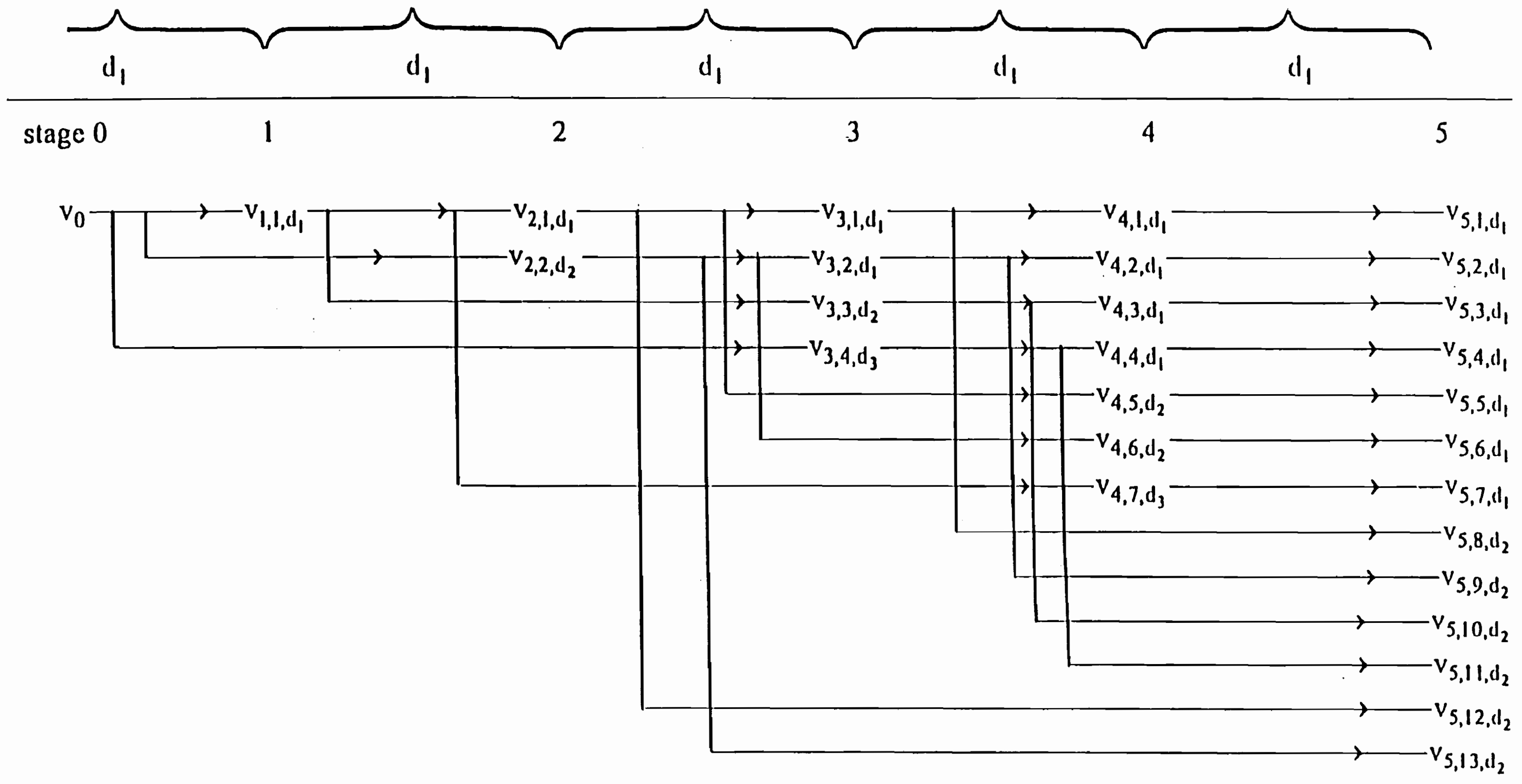


Figure 3.4.1

Structure of the conditional variance sets $[v_j]_k$ for $d_2 = 2*d_1$ and $d_3 = 3*d_1$

$$v_{1,1,d_1} = \left(\frac{v_0 + d_1\sigma_v^2}{v_0 + d_1\sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2.$$

At stage 2 the set $[v_j]_2$ consists of 2 elements as shown below:

$$[v_j]_2 = (v_{2,1,d_1}, v_{2,2,d_2}).$$

Since $d_2 = 2*d_1$ the last sampling interval used at stage 2 can be either d_1 or d_2 units. The 1st element in the set $[v_j]_2$ at stage 2 is obtained on using d_1 as the last sampling interval. The 2nd element is obtained on using d_2 as the last sampling interval. The expression for $v_{2,1,d_1}$ is given by

$$v_{2,1,d_1} = \left(\frac{v_{1,1,d_1} + d_1\sigma_v^2}{v_{1,1,d_1} + d_1\sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2,$$

and for $v_{2,2,d_2}$ is given by

$$v_{2,2,d_2} = \left(\frac{v_0 + d_2\sigma_v^2}{v_0 + d_2\sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2.$$

In Figure 3.4.1, at each stage the arrows mark the different sampling paths from stage 0 up to stage k that give rise to the set $[v_j]_k$. For example, at stage 5, it can be seen that there are 13 elements in the set $[v_j]_5$, 7 of which arise as a result of using d_1 as the last sampling interval denoted by v_{5,j,d_1} for j = 1, 2, ..., 7, and 4 of which arise on using d_2 as the last sampling interval denoted by v_{5,j,d_2} for j = 8, ..., 11, and 2 of which arise on using d_3 as the last sampling interval denoted by v_{5,j,d_3} for j = 12 and 13. As the number of stages k increases, the number of elements in the set $[v_j]_k$ also increase.

In general, let $d_i = m_i\, d_1$ for i=1,2,...,s with $m_1 = 1$ and $1 < m_2 < m_3 \ldots < m_s$ where m_i are all integers. Let $d_{max}(k)$ denote the longest possible sampling interval that can be used just before stage k and let $s_{max}(k)$ denote the previous stage before k at which $d_{max}(k)$ was used. At stage 1, the conditional variance set consists of only one element namely $(v_{1,1,d_1})$, at stage 2, the set consists of two elements $v_{2,1,d_1}$ and $v_{2,2,d_2}$. For any stage k, let the elements in the set $[v_j]_k$ be $(v_{k,1,d_1}, v_{k,2,d_1}, \ldots, v_{k,j,d_i}, \ldots, v_{k,p,d_{max}(k)})$. Let $v_{k,1,d_1}$

denote the element obtained when the shortest interval d_1 is used from stage 0 up to stage k and $v_{k,p,d_{max}(k)}$ denote the element obtained when the number of samples taken from stage 0 up to stage k is the least with $d_{max}(k)$ as the last sampling interval. For example, in Figure 3.4.1, at stage 5, $v_{5,1,d_1}$ is obtained when the shortest interval d_1 is used from stage 0 up to stage 5 and $v_{5,13,d_3}$ is obtained when the number of samples taken from stage 0 up to stage 5 is the least and the last sampling interval is d_3. We see that in this example, the element corresponding to $v_{k,p,d_{max}(k)}$ is $v_{5,13,d_3}$ where $d_{max}(5) = d_3$ and p = 13. The general expression for v_{k,j,d_i} in the set $[v_j]_k$ for j = 1, 2, ..., p is given by

$$v_{k,j,d_i} = \left(\frac{v_{k-m_i,j',d_{i'}} + d_i \sigma_v^2}{v_{k-m_i,j',d_{i'}} + d_i \sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2, \tag{3.4.1}$$

for some i, i′ and where $v_{k-m_i,j',d_{i'}}$ is an element in $[v_{j'}]_{k-m_i}$, $d_i = m_i\, d_1$ and $m_1 = 1$. The conditional variance set $[v_j]_k$ depends on the length of the last sampling interval. If we let $s(k) = k-m_i$, then $d(k) = d_i$. Note that if we drop the subscripts j, j′, d_i, $d_{i'}$, then (3.4.1) can be written as

$$v_k = \left(\frac{v_{s(k)} + d(k)\sigma_v^2}{v_{s(k)} + d(k)\sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2$$

which is equation (3.3.4). Also, the general expression for u_{k,j,d_i} in the set $[u_j]_k$ for j = 1, 2, ..., p is given by

$$u_{k,j,d_i} = \frac{(v_{k-m_i,j',d_{i'}} + d_i \sigma_v^2)^2}{v_{k-m_i,j',d_{i'}} + d_i \sigma_v^2 + \sigma_\varepsilon^2}. \tag{3.4.2}$$

The conditional variance set $[u_j]_k$ depends on the length of the last sampling interval. If we let $s(k) = k-m_i$, then $d(k) = d_i$. Note that if we drop the subscripts j, j′, d_i, $d_{i'}$, then (3.4.2) can be written as

$$u_k = \frac{(v_{s(k)} + d(k)\sigma_v^2)^2}{v_{s(k)} + d(k)\sigma_v^2 + \sigma_\varepsilon^2}$$

which is equation (3.3.5).

From Lemma 1 of Appendix A, for any stage k where $1 \le k \le n$, we see that $v_{k,1,d_1} < v_{k,p,d_{max}(k)}$. Lemma 1 of Appendix A is true for any number of sampling intervals with $d_i = m_i\, d_1$ and $m_1 = 1$. For the rest of this section, we will work with a specific VSI scheme that uses two sampling intervals d_1 and d_2 where d_2 is an integer multiple of d_1. More work needs to be done in order to generalize the material given in the rest of this section to any VSI scheme problem. Now in the 2 sampling interval VSI case, for $k \ge 2$, $d_{max}(k) = d_2$ and the set $[v_j]_k = (v_{k,1,d_1}, v_{k,2,d_1}, \ldots, v_{k,r',d_1}, v_{k,r'+1,d_2} \ldots, v_{k,p,d_2})$. Using Lemma 1 of Appendix A for the VSI process control problem with two sampling intervals, we see that $v_{k,1,d_1} < v_{k,p,d_2}$. Moreover extensive numerical computations have shown the following for the 2 sampling interval VSI scheme:

(a) for any $j = 2, 3, \ldots, r'$, $v_{k,1,d_1} < v_{k,j,d_1} < v_{k,p,d_2}$ and

(b) for any $j = r'+1, r'+2, \ldots, p-1$, $v_{k,1,d_1} < v_{k,j,d_2} < v_{k,p,d_2}$.

Now at each stage k, let min.[v_k] and max.[v_k] denote the minimum and maximum of the set $[v_j]_k$ respectively. Then from (a) and (b) above and Lemma 1 of Appendix A, we see that min.[v_k] = $v_{k,1,d_1}$ and max.[v_k] = v_{k,p,d_2}. This means that this minimum is obtained when samples are taken at lengths of d_1 units, i.e., when an FSI scheme using d_1 units is implemented. Furthermore, the maximum is obtained when the number of samples taken from start up to stage k is the least with $d_{max}(k) = d_2$.

The sequences $\{v_{k,1,d_1}\}$ and $\{u_{k,1,d_1}\}$ refer to the sequences $\{v_{1,1,d_1}, v_{2,1,d_1}, \ldots\}$ and $\{u_{1,1,d_1}, u_{2,1,d_1}, \ldots\}$ respectively. We notice that the sequence $\{v_{k,1,d_1}\} = \{\text{min.}[\ v_k]\}$. If we assume $d_1 = d_2 = 1$, then we have Crowder's process model. Crowder (1986) has shown that the sequence {min.[v_k]} for $d_1 = 1$ converges to a limit as $k \to \infty$. Crowder (1986) has proved that for $d_1 = 1$

$$\lim_{k\to\infty}\{\text{min.}[\ v_k]\} = 0.5 * \left[-\sigma_v^2 + \sqrt{(\sigma_v^4 + 4\sigma_\varepsilon^2\sigma_v^2}\right].$$

Properties (i) and (ii) listed below can be obtained as straightforward extensions to Crowder's proof.

Properties:

(i) $\lim_{k\to\infty}\{\min.[v_k]\} = v_{\min}$

and

$\lim_{k\to\infty}\{\max.[v_k]\} = v_{\max}$

where

$$v_{\min}=0.5*\left[-d_1\sigma_v^2+\sqrt{(d_1^2\sigma_v^4+4\sigma_\varepsilon^2 d_1\sigma_v^2}\right]$$

and

$$v_{\max}=0.5*\left[-d_2\sigma_v^2+\sqrt{(d_2^2\sigma_v^4+4\sigma_\varepsilon^2 d_2\sigma_v^2}\right].$$

(ii) $\lim_{k\to\infty}\{\min.[u_k]\} = u_{\min}$

and

$\lim_{k\to\infty}\{\max.[u_k]\} = u_{\max}$

where

$$u_{\min} = \frac{(v_{\min} + d_1\sigma_v^2)^2}{v_{\min} + d_1\sigma_v^2 + \sigma_\varepsilon^2}$$

and

$$u_{\max}=\frac{(v_{\max}+d_2\sigma_v^2)^2}{v_{\max}+d_2\sigma_v^2+\sigma_\varepsilon^2}.$$

Extensive numerical computational have shown that

(iii) If $v_s \in [v_{\min}, v_{\max}]$, then $v_t \in [v_{\min}, v_{\max}]$ for any $t \geq s$. Further, if $v_0 < v_{\min}$, then $v_t \leq v_{\max}$. If $v_0 > v_{\max}$, then $v_t \geq v_{\min}$, for any t.

The number of elements in the set $[v_j]_k$ and $[u_j]_k$ increase rapidly as the number of stages k increase, making it computationally very time consuming when the actual values for the conditional variance in the set $[v_j]_k$ and $[u_j]_k$ are used. As a result, one would need to approximate the set $[v_j]_k$ and $[u_j]_k$. The approximation depends on the integer multiple relating d_2 to d_1 and the observation error variance σ_ε^2. More on the issue of approximation will be given in Chapter 5.

3.5 Adjustment and Sampling Strategies

For the rest of this dissertation, we will work with the VSI process control problem using two sampling intervals. The two sampling intervals are d_1 and d_2 and $d_2 = m*d_1$ where m is an integer. A big part of this dissertation is to determine the adjustment and sampling strategies for the VSI process control problem specified by the observation and process mean equations given by (2.3.4) and (3.2.1) respectively with a VSI scheme that uses d_1 and d_2 as the two sampling intervals such that the expected loss function (3.2.6) is minimized. In the sections that follow a theoretical derivation of the adjustment and sampling strategies is given. In Section 3.5.1 a general outline of how the dynamic programming approach is used to determine these strategies is given. In Section 3.5.2 the derivation of a set of functional equations is given. Using the functional equations, the adjustment and sampling strategies are determined.

In order to understand the terminologies used hereforth, we will state the following: (1) in general, an n+1 stage problem implies that there are n decision points namely, 0, 1, ..., n-1 and n-1 possible sampling points namely, 1, 2, ..., n-1, (2) a period is defined as the length of time between any two successive stages and (3) given an n+1 stage problem, a (n-k)-period problem implies that starting from stage k there are n-k periods remaining in the problem.

3.5.1 Dynamic Programming Approach

The technique of dynamic programming (Cooper and Cooper (1981)) will be used to find the adjustment and sampling strategies that minimize the expected loss expression (3.2.5). The goal is to find the adjustment and the sampling limits at each one of the decision points 0, 1, ..., n-1. The decision, to adjust or not to adjust is based on the adjustment limits. The decision, use d_1 as the next sampling interval or use d_2 as the next sampling interval is based on the sampling limits. These limits are computed "sequentially" or as "multi-stage" decisions using the recurrence or functional equations derived in the following Section 3.5.2. The expected loss functional equations for an n+1 stage problem are derived starting from the last stage and moving backwards. With the help of two examples, given by Figures 3.5.1 and 3.5.2, the outline of the derivation of the functional equations will be explained in four steps. Figure 3.5.1 assumes that $m = 2$, i.e., $d_2 = 2*d_1$ and Figure 3.5.2 assumes that $m = 3$, i.e., $d_2 = 3*d_1$.

Step 1: Last adjustment strategy

In this step the last adjustment strategy is found for the 1-period problem. As shown in Figures 3.5.1 and 3.5.2, the 1-period problem is given by the period from stage (n-1) to stage n. Due to the assumption that no sample will be taken at stage n, at stage n-1 there are two decisions to be considered. These two decisions are to make an adjustment to the process and not to make an adjustment to the process. Given that a sample is taken at stage (n-1), the last adjustment limits are found such that the total expected loss for the 1-period problem is minimized. These adjustment limits decide whether an adjustment should be made at stage (n-1) or not. Opportunities to make a non-zero adjustment is available only at stages where the process mean has been observed.

Step 2: k^{th} adjustment strategy, $m > 2$, and $(n-m+1) \leq k \leq (n-2)$

At this step, we are at stage k with (n-k) periods remaining where $(n-m+1) \leq k \leq (n-2)$ and $m > 2$. As in Step 1, assuming that a sample is taken at stage k, there are two decisions to be considered at this step for the (n-k)-period problem. The two decisions are make an adjustment to the process and not to make an adjustment to the process. There are no decisions with respect to sampling at this step because when $d_2 = m*d_1$, the distance between the stages k and k+1 for $(n-m+1) \leq k \leq (n-2)$ and $m > 2$ is d_1 units.

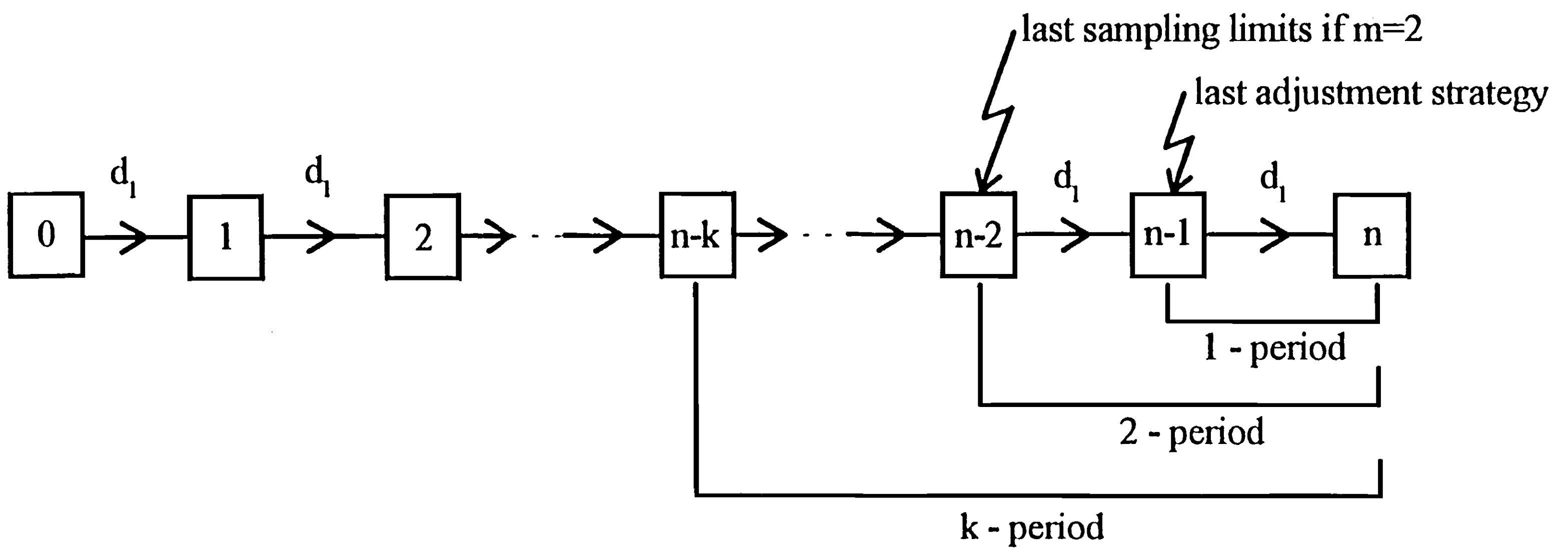


Figure 3.5.1

Example of a k-period problem $d_2 = 2*d_1$

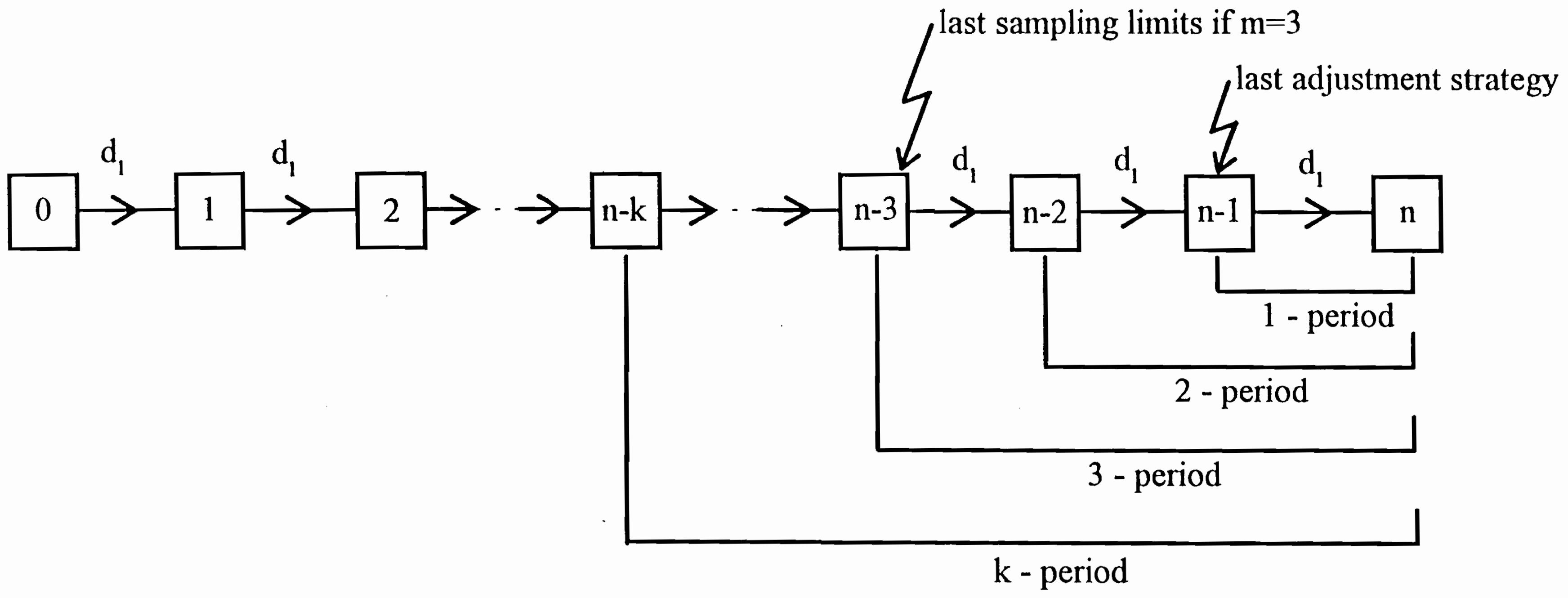


Figure 3.5.2

Example of a k-period problem $d_2 = 3 * d_1$

Moreover, if m = 2, then one will proceed to Step 3 from Step 1 and Step 2 will not be used. The total expected loss for a (n-k)-period problem is obtained using information from the (n-k-1)-period problem. The information from the (n-k-1)-period problem consists of the adjustment limits and the expected loss for a (n-k-1)-period problem. Given that a sample is taken at stage k, the kth adjustment limits are found such that the total expected loss function for the (n-k)-period problem is minimized. For example, in Figure 3.5.2 when m = 3, this step corresponds to finding the adjustment limits to the 2-period problem.

Step 3: kth adjustment and sampling strategies, $m \geq 2$, k = (n-m)

When k = (n-m), we are actually referring to the m-period problem which means that there are m remaining periods starting from stage k. At this step, given that a sample is taken at stage k, we shall obtain not only the adjustment strategies, but also the sampling strategies to the m-period problem. As stated in the previous two steps the adjustment strategies refer to two decisions, namely make an adjustment to the process and do not make an adjustment to the process. The sampling strategies refer to two decisions, namely use d_2 as the next sampling interval and use d_1 as the next sampling interval. As a result of the two sampling and adjustment decisions, four possible decisions can be considered at this step. These are

(i) adjust and use d_1 as the next sampling interval,

(ii) do not adjust and use d_1 as the next sampling interval,

(iii) adjust and use d_2 as the next sampling interval,

and

(iv) do not adjust and use d_2 as the next sampling interval.

The adjustment and sampling strategies at stage k specify when to use each of the above four decisions. These strategies are found using information from the (n-k-1)-

period problem and the expected loss obtained on using d_2 for the first time. In Figure 3.5.1 for m = 2, this step corresponds to finding the adjustment and sampling strategies for the 2-period problem. This is represented by the segment from stage (n-2) to stage n. We notice from the figure that the sampling limits found at this step are the first set of sampling limits, obtained moving backward. In Figure 3.5.2 for m = 3, this step corresponds to the 3-period problem. This is represented by the segment from stage (n-3) to stage n. The set of sampling limits found for this example is again the first set of sampling limits found moving backward.

Step 4: kth adjustment and sampling strategies, $m \geq 2$, $0 \leq k \leq (n\text{-}m\text{-}1)$

Now for a (n-k)-period problem where $0 \leq k \leq (n\text{-}m\text{-}1)$, as in Step 3 given that a sample is taken at stage k both the adjustment and sampling strategies will be determined. As stated in Step 3, four possible decisions can be considered in this step. The four decisions are listed in Step 3. In this step the kth adjustment and sampling limits are found using information from the (n-k-1)-period problem and the (n-k+m)-period problem.

In the following section we will derive the set of functional recursive equations based on the 4 steps given above. The functional recursive equations will be used to obtain the adjustment and the sampling strategies to our VSI process control problem.

3.5.2 Functional recursive equations

The following notation is used in the development of the functional recursive equations:

δ_i = decisions regarding sampling and adjustments at stage i

$EL_{i,n}(\hat{\mu}_i, v_i, \delta_i)$ = the total expected loss computed at stage i for the remaining (n-i) periods in an n period problem expressed as a function of $\hat{\mu}_i$,,v_i and the decisions at stage i, given information up to stage i

$\Re_{i,n}(\hat{\mu}_i, v_i)$ = the minimum expected loss computed at stage i for the remaining (n-i) periods in an n period problem given information up to stage i, also called the risk function for a (n-i)-period problem

$A_i S_{d_1}$ = the decision of adjusting at stage i and using d_1 as the next sampling interval

$\overline{A}_i S_{d_1}$ = the decision of not adjusting at stage i and using d_1 as the next sampling interval

$A_i S_{d_2}$ = the decision of adjusting at stage i and using d_2 as the next sampling interval

$\overline{A}_i S_{d_2}$ = the decision of not adjusting at stage i and using d_2 as the next sampling interval

$f(x{:}\ \mu, \sigma^2)$ = the normal density function with mean μ and variance σ^2.

<u>Step 1: Last adjustment strategy</u>

As stated in Section 3.5.1 the 1-period problem is specified by the segment from stage n-1 to stage n for an n+1 stage problem. When a sample is taken at stage n-1, two possible decisions can be made at this point. These are either adjust at stage n-1 or do not adjust at stage n-1. Since here we assume that a sample is taken at stage n-1, the sampling times $t_{n'}$ and $t_{n'+1}$ are the times associated with stages n-1 and n respectively and the distance between stages n-1 and n is d_1 units. Now using (3.2.5) one can find the integrated off-target cost for the last period. Then using this integrated off-target cost from stage n-1 to stage n and the performance criterion (3.2.6) conditional on data up to stage n-1, the expected loss for the 1-period problem, $EL_{n-1,n}(\hat{\mu}_{n-1}, v_{n-1}, \delta_{n-1})$ is given by

$$EL_{n-1,n}(\hat{\mu}_{n-1}, v_{n-1}, \delta_{n-1}) = \begin{cases} c_1 v_{n-1} d_1 + \dfrac{c_1 \sigma_v^2 d_1^2}{2} + c_1 \hat{\mu}_{n-1}^2 d_1 + c_3 & \text{if } \overline{A}_{n-1} S_{d_1} \\ c_1 v_{n-1} d_1 + \dfrac{c_1 \sigma_v^2 d_1^2}{2} + c_2 + c_3 & \text{if } A_{n-1} S_{d_1} \end{cases}$$

where the set $\{\overline{A}_{n-1} S_{d_1}, A_{n-1} S_{d_1}\}$ is the range of the decision rule δ_{n-1} at stage n-1. Recall that c_1, c_2, and c_3 are the costs associated with the process being off-target, making an adjustment and taking a sample respectively.

Let $\mathfrak{R}_{n\text{-}1,n}(\hat{\mu}_{n-1},v_{n\text{-}1})$ denote the minimum expected loss for the 1-period problem given by

$$\mathfrak{R}_{n\text{-}1,n}(\hat{\mu}_{n-1},v_{n\text{-}1}) = \min_{a_{n-1}}[EL_{n-1,n}(\hat{\mu}_{n-1},v_{n\text{-}1},\delta_{n-1})]$$
$$=\min_{a_{n-1}}[c_1 v_{n\text{-}1}d_1+\frac{c_1\sigma_v^2 d_1^2}{2}+c_1(\hat{\mu}_{n-1}+a_{n-1})^2 d_1+c_2\delta(a_{n-1})+c_3]$$
$$=c_1 v_{n\text{-}1}d_1+\frac{c_1\sigma_v^2 d_1^2}{2}+\min(c_1\hat{\mu}_{n-1}^2 d_1,c_2)+c_3.$$

(3.5.1)

It can be seen that $c_1\hat{\mu}_{n-1}^2 d_1$ and c_2 are both nondecreasing in $|\hat{\mu}_{n-1}|$ and so by Lemma 2 of Appendix A, $\min(c_1\hat{\mu}_{n-1}^2 d_1,c_2)$ is nondecreasing in $|\hat{\mu}_{n-1}|$. Clearly, $\mathfrak{R}_{n\text{-}1,n}(\hat{\mu}_{n-1},v_{n\text{-}1})$ is also nondecreasing in $|\hat{\mu}_{n-1}|$, nonnegative and symmetric about the origin. Then the solution to the above minimization problem is given by the solution to $c_1\hat{\mu}_{n-1}^2 d_1 = c_2$, namely

$$a_{n-1}=\begin{cases} -\hat{\mu}_{n-1} & \text{if } |\hat{\mu}_{n-1}|\geq\sqrt{c_2/c_1 d_1} \\ 0 & \text{if } |\hat{\mu}_{n-1}|<\sqrt{c_2/c_1 d_1}. \end{cases} \qquad (3.5.2)$$

The above equation (3.5.2) has the following interpretation. At stage n-1 if the computed value of the Kalman filter estimator $\hat{\mu}_{n-1}$ falls outside the band represented by $\left[-\sqrt{c_2/c_1 d_1},\ +\sqrt{c_2/c_1 d_1}\right]$, then an adjustment equivalent to $-\hat{\mu}_{n-1}$ is indicated. This band is called the adjustment band. On the other hand, if the Kalman filter estimate falls inside the adjustment band, then no adjustment is called for. The points $\pm\sqrt{c_2/c_1 d_1}$ are called the adjustment limits for the 1-period problem and are denoted by $\pm AL_{n-1}$.

<u>Step 2: kth adjustment strategy, m > 2, and (n-m+1) ≤ k ≤ (n-2)</u>

As stated in Step 2 of Section 3.5.1, assuming that a sample is taken at stage k, we shall use the information from the (n-k-1)-period problem in the expected loss expression at this stage. The expected loss for the (n-k)-period problem is obtained as a sum of two quantities, the first is the expected loss from stage k to stage k+1 and the second is the expected value of the optimum risk suffered thereafter.

Let t_j and t_{j+1} in (3.2.5) represent times associated with stages k and k+1 respectively. Then, using the expression for the integrated off-target cost from stage k to stage k+1, the minimum expected loss for the (n-k)-period problem is given by

$$\mathfrak{R}_{k,n}(\hat{\mu}_k, v_k) = \min_{a_k}\Bigg[c_1 v_k d_1 + \frac{c_1\sigma_v^2 d_1^2}{2} + c_1(\hat{\mu}_k + a_k)^2 d_1 + c_2\delta(a_k) + c_3 + E\left[\mathfrak{R}_{k+1,n}(\hat{\mu}_{k+1}, v_{k+1}) | \hat{\mu}_k, v_k\right] \Bigg],$$

where $c_1 v_k d_1 + \frac{c_1\sigma_v^2 d_1^2}{2} + c_1(\hat{\mu}_k + a_k)^2 d_1 + c_2\delta(a_k) + c_3$ is the expected loss from stage k to stage k+1. On using the conditional distribution of $\hat{\mu}_{k+1}$ given past information up to stage k, namely $\hat{\mu}_{k+1} | x_0, \gamma(1)x_1, \dots, \gamma(j)x_j, \dots, \gamma(k-m)x_{k-m}, \gamma(k-m+1)x_{k-m+1}, \dots, x_k$, the above minimum expected loss is given by

$$\mathfrak{R}_{k,n}(\hat{\mu}_k, v_k) = \min_{a_k}\Bigg[c_1 v_k d_1 + \frac{c_1\sigma_v^2 d_1^2}{2} + c_1(\hat{\mu}_k + a_k)^2 d_1 + c_2\delta(a_k) + c_3 + \int \mathfrak{R}_{k+1,n}(x, v_{k+1}) f(x : \hat{\mu}_k + a_k, u_{k+1}) dx \Bigg], \tag{3.5.3}$$

where f(x: $\hat{\mu}_k + a_k, u_{k+1}$) denotes the normal density with mean $\hat{\mu}_k + a_k$ and variance u_{k+1}, the density of $\hat{\mu}_{k+1} | x_0, \gamma(1)x_1, \dots, \gamma(j)x_j, \dots, \gamma(k-m)x_{k-m}, \gamma(k-m+1)x_{k-m+1}, \dots, x_k$. Assuming certain properties for $\mathfrak{R}_{k+1,n}(x, v_{k+1})$, we shall show that $\mathfrak{R}_{k,n}(\hat{\mu}_k, v_k)$ has the same properties. Let the minimum expected loss $\mathfrak{R}_{k+1,n}(x, v_{k+1})$ be nonnegative, symmetric about the origin, and nondecreasing in $|x|$. We first notice that on using Lemma 3 of Appendix A, $\int \mathfrak{R}_{k+1,n}(x, v_{k+1}) f(x : \hat{\mu}_k + a_k, u_{k+1}) dx$ is nondecreasing in $|\hat{\mu}_k + a_k|$ for any fixed u_{k+1}. Therefore the nonzero value of 'a' that minimizes (3.5.3) is $-\hat{\mu}_k$. Then (3.5.3) can be written as

$$\Re_{k,n}(\hat{\mu}_k, v_k) = c_1 v_k d_1 + \frac{c_1 \sigma_v^2 d_1^2}{2} + c_3 + \min\left[c_1 \hat{\mu}_k^2 d_1 + \int \Re_{k+1,n}(x, v_{k+1}) f(x{:}\hat{\mu}_k, u_{k+1}) dx; \right.$$
$$\left. c_2 + \int \Re_{k+1,n}(x, v_{k+1}) f(x{:}0, u_{k+1}) dx \right].$$

(3.5.4)

Then, on using Theorem 2 of Appendix A, we see that $\Re_{k,n}(\hat{\mu}_k, v_k)$ is nonnegative, symmetric about the origin and nondecreasing in $|\hat{\mu}_k|$.

The expected loss expressions for the two decisions at this stage are given by

$$EL_{k,n}(\hat{\mu}_k, v_k, \delta_k)$$

$$= \begin{cases} c_1 v_k d_1 + \frac{c_1 \sigma_v^2 d_1^2}{2} + c_3 + c_1 \hat{\mu}_k^2 d_1 + \int \Re_{k+1,n}(x, v_{k+1}) f(x{:}\hat{\mu}_k, u_{k+1}) dx & \text{if } \overline{A}_k S_{d_1} \\ c_1 v_k d_1 + \frac{c_1 \sigma_v^2 d_1^2}{2} + c_3 + c_2 + \int \Re_{k+1,n}(x, v_{k+1}) f(x{:}0, u_{k+1}) dx & \text{if } A_k S_{d_1} \end{cases}$$

(3.5.5)

where the set $\{\overline{A}_k S_{d_1}, A_k S_{d_1}\}$ is the range of the decision rule δ_k at stage k. From (3.5.5), we see that $EL_{k,n}(\hat{\mu}_k, v_k, \overline{A}_k S_{d_1})$ is strictly increasing in $|\hat{\mu}_k|$ since $\int \Re_{k+1,n}(x, v_{k+1}) f(x{:}\hat{\mu}_k, u_{k+1}) dx$ is nonnegative, symmetric about the origin and nondecreasing in $|\hat{\mu}_k|$ for a given u_{k+1}, $EL_{k,n}(\hat{\mu}_k, v_k, A_k S_{d_1})$ is a constant and $EL_{k,n}(0, v_k, \overline{A}_k S_{d_1}) < EL_{k,n}(0, v_k, A_k S_{d_1})$. Thus, we can conclude that there is a point of intersection for the two functions given by the two decisions $\overline{A}_k S_{d_1}$ and $A_k S_{d_1}$ in (3.5.5). This point of intersection of the two functions gives the adjustment limits at stage k. Thus the k th adjustment strategy is given by

$$a_k = \begin{cases} -\hat{\mu}_k & \text{if } |\hat{\mu}_k| \geq AL_k \\ 0 & \text{if } |\hat{\mu}_k| < AL_k, \end{cases} \qquad (3.5.6)$$

where AL_k denotes the adjustment limit at stage k.

Step 3: kth adjustment and sampling strategies, $m \geq 2$, $k = (n-m)$

As explained in Step 3 of Section 3.5.1, at this step four possible decisions can be considered, obtained as a result of two adjustment decisions and two sampling decisions.

The minimum expected loss for the m-period problem is given by

$$\mathfrak{R}_{k,n}(\hat{\mu}_k, v_k) = \min_{a_k}\left\{ \left(c_1 v_k d_1 + \frac{c_1\sigma_v^2 d_1^2}{2} + c_3 + \left[c_2\delta(a_k) + c_1(\hat{\mu}_k + a_k)^2 d_1 + \int \mathfrak{R}_{k+1,n}(x, v_{k+1}) f(x{:}\hat{\mu}_k + a_k, u_{k+1})dx \right] \right); \left(c_1 v_k d_2 + \frac{c_1\sigma_v^2 d_2^2}{2} + c_3 + \left[c_2\delta(a_k) + c_1(\hat{\mu}_k + a_k)^2 d_2 \right] \right) \right\}. \tag{3.5.7}$$

In (3.5.7) above, the first expression in min.{(.);(.)} corresponds to the expected loss obtained on using d_1 as the next sampling interval, which is the sum of two quantities; one being the expected loss from stage k to stage k+1, and the other being the expected value of the optimum risk suffered thereafter. This first expression is similar to (3.5.3) and we have seen that it is nonnegative, symmetric about the origin and nondecreasing in $|\hat{\mu}_k + a_k|$. The second expression of min.{(.);(.)} corresponds to the expected loss obtained on using d_2 as the next sampling interval for the first time when using backward induction. It is clearly seen that this second expression is nonnegative, symmetric about the origin, and nondecreasing in $|\hat{\mu}_k + a_k|$. Now applying Lemma 2 of Appendix A to min.{(.);(.)}, we see that the risk function $\mathfrak{R}_{k,n}(\hat{\mu}_k, v_k)$ is nonnegative, symmetric about the origin, and nondecreasing in $|\hat{\mu}_k + a_k|$. Therefore the nonzero minimizing choice for 'a' is $-\hat{\mu}_k$.

The following are the expected loss expressions for the four possible decisions at this stage:

$$EL_{k,n}(\hat{\mu}_k, v_k, \delta_k)$$

$$= \begin{cases} c_1 v_k d_1 + \dfrac{c_1 \sigma_v^2 d_1^2}{2} + c_3 + c_1 \hat{\mu}_k^2 d_1 + \int \Re_{k+1,n}(x, v_{k+1}) f(x{:}\hat{\mu}_k, u_{k+1})dx & \text{if } \overline{A}_k S_{d_1} \\ c_1 v_k d_1 + \dfrac{c_1 \sigma_v^2 d_1^2}{2} + c_3 + c_2 + \int \Re_{k+1,n}(x, v_{k+1}) f(x{:}0, u_{k+1})dx & \text{if } A_k S_{d_1} \\ c_1 v_k d_2 + \dfrac{c_1 \sigma_v^2 d_2^2}{2} + c_3 + c_1 \hat{\mu}_k^2 d_2 & \text{if } \overline{A}_k S_{d_2} \\ c_1 v_k d_2 + \dfrac{c_1 \sigma_v^2 d_2^2}{2} + c_3 + c_2 & \text{if } A_k S_{d_2} \end{cases}$$

(3.5.8)

where the set $\{\overline{A}_k S_{d_1}, A_k S_{d_1}, \overline{A}_k S_{d_2}, A_k S_{d_2}\}$ is the range of the decision rule δ_k at stage k.

In (3.5.8) the expressions corresponding to the decisions of $A_k S_{d_1}$ and $A_k S_{d_2}$ are independent of $\hat{\mu}_k$. The expressions corresponding to the decisions $\overline{A}_k S_{d_1}$ and $\overline{A}_k S_{d_2}$ are strictly increasing in $|\hat{\mu}_k|$. Moreover, the expressions corresponding to all the four decisions are nonnegative and symmetric about the origin. Therefore using Lemma 4 of Appendix A we see that min.$EL_{k,n}(\hat{\mu}_k, v_k)$ is also nonnegative, symmetric about the origin and nondecreasing in $|\hat{\mu}_k|$. In (3.5.8), since both functions $EL_{k,n}(\hat{\mu}_k, v_k, \overline{A}_k S_{d_1})$ and $EL_{k,n}(\hat{\mu}_k, v_k, \overline{A}_k S_{d_2})$ are strictly increasing in $|\hat{\mu}_k|$, $EL_{k,n}(0, v_k, \overline{A}_k S_{d_1}) < EL_{k,n}(0, v_k, A_k S_{d_1})$ and $EL_{k,n}(0, v_k, \overline{A}_k S_{d_2}) < EL_{k,n}(0, v_k, A_k S_{d_2})$, we can conclude that there are at most two sets and at least one set of points of intersection for the four functions in (3.5.8) that give the minimum expected loss. If there are two sets of points of intersection, then there are called the sampling limits, denoted by $\pm$ SL_k and the adjustment limits denoted by $\pm$ AL_k. The sampling limits determine whether the shorter interval d_1 or the longer interval d_2 is to be used as the next sampling interval. If there is only one set of points of intersection, then these are called the adjustment limits. The adjustment limits determine whether the process requires an adjustment or not.

Step 4: kth adjustment and sampling strategies, $m \geq 2$, $0 \leq k \leq (n-m-1)$

As explained in Step 4 of Section 3.5.1 at this step we will determine the sampling and adjustment strategies for the (n-k)-period problem using information from the (n-k-1)-period and (n-k-m)-period problems. At this step, the expressions for the integrated off-target costs from stage k to stage k+1 and from stage k to stage k+m will be used. Using these integrated off-target costs and the conditional distributions $\hat{\mu}_{k+1}|x_0,\gamma(1)x_1,\ldots,\gamma(j)x_j,\ldots,\gamma(k-m)x_{k-m},\gamma(k-m+1)x_{k-m+1},\ldots,x_k$ and $\hat{\mu}_{k+m}|x_0,\gamma(1)x_1,\ldots,\gamma(j)x_j,\ldots,\gamma(k-m)x_{k-m},\gamma(k-m+1)x_{k-m+1},\ldots,x_k$, the recursive minimum expected loss function is given by

$$\begin{aligned}\Re_{k,n}(\hat{\mu}_k,v_k) = \min_{a_k}\Bigg\{ &\Bigg(c_1v_kd_1+\frac{c_1\sigma_v^2d_1^2}{2}+c_3+\\ &\left[c_2\delta(a_k)+c_1(\hat{\mu}_k+a_k)^2d_1+\int\Re_{k+1,n}(x,v_{k+1})f(x:\hat{\mu}_k+a_k,u_{k+1})dx\right]\Bigg);\\ &\Bigg(c_1v_kd_2+\frac{c_1\sigma_v^2d_2^2}{2}+c_3+\\ &\left[c_2\delta(a_k)+c_1(\hat{\mu}_k+a_k)^2d_1+\int\Re_{k+m,n}(x,v_{k+m})f(x:\hat{\mu}_k+a_k,u'_{k+m})dx\right]\Bigg)\Bigg\}.\end{aligned}$$

(3.5.9)

The functions $f(x\colon \hat{\mu}_k+a_k,u_{k+1})$ and $f(x\colon \hat{\mu}_k+a_k,u'_{k+1})$ denote the normal densities of $\hat{\mu}_{k+1}|x_0,\gamma(1)x_1,\ldots,\gamma(j)x_j,\ldots,\gamma(k-m)x_{k-m},\gamma(k-m+1)x_{k-m+1},\ldots,x_k$ and $\hat{\mu}_{k+m}|x_0,\gamma(1)x_1,\ldots,\gamma(j)x_j,\ldots,\gamma(k-m)x_{k-m},\gamma(k-m+1)x_{k-m+1},\ldots,x_k$ respectively. In (3.5.9) the first expression in min.{(.), (.)} represents the expected loss on using d_1 as the next sampling interval, which is obtained as a sum of two quantities, one being the expected loss from stage k to stage k+1 and the other being the expected value of the optimum risk suffered thereafter. The second expression in min.{(.), (.)} represents the expected loss on using d_2 as the next sampling interval, which is obtained as a sum of two quantities, one being the expected loss from stage k to stage k+m and the other being the expected value of the optimum risk suffered thereafter. Applying Lemma 2 and Lemma 3 of Appendix A to the first and the second expressions of min.{(.), (.)} in (3.5.9), it is seen that both these expressions are nondecreasing in $|\hat{\mu}_k+a_k|$. Therefore, the nonzero minimizing choice of 'a' is given by $-\hat{\mu}_k$.

The expected loss functions for the four possible decisions namely, $A_kS_{d_1}$, $A_kS_{d_2}$, $\overline{A}_kS_{d_1}$ and $\overline{A}_kS_{d_2}$ as follows:

$$EL_{k,n}(\hat{\mu}_k,v_k,\delta_k)$$
$$=\begin{cases} c_1v_kd_1+\dfrac{c_1\sigma_v^2d_1^2}{2}+c_3+c_1\hat{\mu}_k^2d_1+\displaystyle\int\Re_{k+1,n}(x,v_{k+1})f(x:\hat{\mu}_k,u_{k+1})dx & \text{if } \overline{A}_kS_{d_1} \\ c_1v_kd_1+\dfrac{c_1\sigma_v^2d_1^2}{2}+c_3+c_2+\displaystyle\int\Re_{k+1,n}(x,v_{k+1})f(x:0,u_{k+1})dx & \text{if } A_kS_{d_1} \\ c_1v_kd_2+\dfrac{c_1\sigma_v^2d_2^2}{2}+c_3+c_1\hat{\mu}_k^2d_2+\displaystyle\int\Re_{k+m,n}(x,v_{k+m})f(x:\hat{\mu}_k,u'_{k+m})dx & \text{if } \overline{A}_kS_{d_2} \\ c_1v_kd_2+\dfrac{c_1\sigma_v^2d_2^2}{2}+c_3+c_2+\displaystyle\int\Re_{k+m,n}(x,v_{k+m})f(x:0,u'_{k+m})dx & \text{if } A_kS_{d_2} \end{cases}$$

(3.5.10)

where the set $\{\overline{A}_kS_{d_1}, A_kS_{d_1}, \overline{A}_kS_{d_2}, A_kS_{d_2}\}$ is the range of the decision rule δ_k at stage k.

For a fixed u_{k+1} and u'_{k+1}, on applying Lemma 3 of Appendix A it is seen that the expressions in (3.5.10) for the decisions $\overline{A}_kS_{d_1}$ and $\overline{A}_kS_{d_2}$ are strictly increasing in $|\hat{\mu}_k|$ and those for the decisions $A_kS_{d_1}$ and $A_kS_{d_2}$ are constants. Also, each of these expressions are nonnegative and symmetric about the origin. Thus, using Lemma 4 of Appendix A it is seen that $\min.EL_{k,n}(\hat{\mu}_k,v_k)$ is nondecreasing in $|\hat{\mu}_k|$. It is also nonnegative and symmetric about the origin. As in Step 3, since $EL_{k,n}(0,v_k,\overline{A}_kS_{d_1}) < EL_{k,n}(0,v_k,A_kS_{d_1})$ and $EL_{k,n}(0,v_k,\overline{A}_kS_{d_2}) < EL_{k,n}(0,v_k,A_kS_{d_2})$, we can conclude that there are at most two sets and at least one set of points of intersection for the four functions in (3.5.10) that give the minimum expected loss. Thus, the sampling and the adjustment limits are found as the points of intersection of the four expressions in (3.5.10) that give rise to the minimum expected loss at stage k. In Chapter 4 the types of behavior of the expected loss functions for the possible decisions at the 4 stages are displayed.

Chapter 4: Numerical Methods

4.1 Introduction

In the last chapter a theoretical derivation of the adjustment and sampling strategies was given. In the present chapter a graphical explanation of the strategies and numerical examples comparing the performance of the VSI scheme with that of the FSI scheme are given. An overview of the computational method used to determine the adjustment limits, sampling limits and the minimum expected loss for the VSI process control problem described by the observation equation (2.3.4) and the process mean equation (3.2.1) is given. Further the types of behavior of the expected loss functions are described graphically.

4.2 Overview of the computing method

In this section a computational method which yields the adjustment limits, sampling limits, and the expected loss for the theory explained in Chapter 3 is described. A computer program in C language (Appendix B) has been written that computes the above mentioned limits and the expected loss at each stage.

4.2.1 Initial calculations

As stated in Chapter 3 let us suppose that the FSI scheme using intervals of 1 unit has been in effect on a process that follows Crowder's model for a substantial amount of time. Now, it is desired to implement a VSI scheme beginning immediately for the next t time units. Let us assume that the VSI process model uses the VSI scheme consisting of two sampling intervals d_1 and d_2 units where $d_2 = m*d_1$. Given t' and d_1 the total number of stages, namely n can be calculated. The total number of stages n represents the n possible sampling points in the problem. For example, if we let $t' = 40$ hours and $d_1 = 0.8$ hours, then $n = 40/0.8 = 50$ which implies that there are 50 possible sampling points 0.8

hours apart. Next from Section 3.4 the starting variance v_0 for the VSI process model is given by

$$v_0 = 0.5 * \left[-\sigma_v^2 + \sqrt{(\sigma_v^4 + 4\sigma_\varepsilon^2 \sigma_v^2)} \right].$$

In Section 3.4, we saw that at any stage k, the conditional variance sets $[v_j]_k$ and $[u_j]_k$ can be calculated given v_0, d_1 and d_2. Both the conditional variance sets depend on the lengths of the last sampling interval used at stage k. We also noticed that the number of elements in these sets increase rapidly as the number of stages increase resulting in very time consuming computations of the expected loss and the limits. We could overcome the problem of the time consuming computations by using an approximation to the number of elements in the sets. From the properties of the conditional variances mentioned in Section 3.4 and extensive numerical experimentation, in most cases, one can find an integer k′, where $k' \leq k \leq n$ such that $[v_j]_{k'}$ and $[u_j]_{k'}$ are good approximations to $[v_j]_k$ and $[u_j]_k$ respectively. The value of k′ depends on the integer multiple m that relates the longer sampling interval d_2 to the shorter sampling interval d_1 and the observation error σ_ε^2. On the other hand, for any $k \leq k'$ the actual sets $[v_j]_k$ and $[u_j]_k$ would be used in the calculation of the total expected loss and the limits at each stage. At what stage is one justified in using the approximation? To answer this question let us consider an example. Let n = 30, m = 2, and $\sigma_\varepsilon^2 = 1.0$. Table 4.2.1-(a) gives the number of elements in the sets $[v_j]_{s'}$ and $[u_j]_{s'}$ for s′= 1,2,...,14. For example, when s′ = 10, the number of elements in set $[v_j]_{10}$ is 55 and the subscript j ranges from 1 to 55. Now given a cost combination, the total expected loss for the 30 stage problem was calculated using the elements in $[v_j]_{s'}$ and $[u_j]_{s'}$ for s′= 2 as approximations to the elements in $[v_j]_k$ and $[u_j]_k$ for $2 \leq k \leq 30$. Next the same procedure was repeated for s′= 3,4,...,14. It was seen that the change in the total expected loss was negligible for $s' \geq 9$. In the calculations of the adjustment and the sampling strategies to the 30 stage problem, one can thus use the

Table 4.2.1-(a)

Number of elements in the conditional variance set - Example for $\sigma_\varepsilon^2 \leq 5.0$ and $m = 2$

s' Stage	Number of elements in $[v_j]_{s'}$
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
11	89
12	144
13	233
14	377

Table 4.2.1-(b)

Approximation to the conditional variance sets
for $\sigma_\varepsilon^2 \leq 5.0$

m (the multiple)	v' (number of elements in $[v_j]_{k'}$)	k' (the stage at which the approximation starts)
2	34	9
3	60	13
4	95	17
5	106	20
6	119	23
7	105	25
8	119	28
9	134	31
10	150	34
11	167	37
12	185	40
13	238	44

elements in the sets $[v_j]_9$ and $[u_j]_9$ as good approximations to $[v_j]_k$ and $[u_j]_k$ for $9 \le k \le 30$. For the example above, $k' = 9$ and the number of elements in $[v_j]_9$ and $[u_j]_9$ is 34 each. For $\sigma_\varepsilon^2 \le 5.0$ and certain values m, Table 4.2.1- (b) gives the number of elements in the sets $[v_j]_{k'}$ and $[u_j]_{k'}$ (denoted by v′) that can be used as good approximations to $[v_j]_k$ and $[u_j]_k$ for $k' \le k \le n$. These approximations are used in the computations which yields the optimal strategies to the VSI process control problem.

For the same values of m, Table (4.2.1)-(b) that is given for $\sigma_\varepsilon^2 \le 5.0$ may not be the same for $\sigma_\varepsilon^2 > 5.0$ since the value of k′ depends on the multiple m and the observation error σ_ε^2. The number of elements in the sets $[v_j]_{k'}$ and $[u_j]_{k'}$ which are used as good approximations to the sets $[v_j]_k$ and $[u_j]_k$ for $k' \le k \le n$ increases as m and σ_ε^2 increase and thus the computations of the expected losses become very time consuming. In this dissertation the calculations have been restricted to two values of the observation error, namely $\sigma_\varepsilon^2 = 1.0$ and $\sigma_\varepsilon^2 = 5.0$.

The input parameters used in the computer program are:

- c_1, the cost associated with the process being off-target,
- c_2, the cost associated with making a non zero adjustment,
- c_3, the cost associated with taking a sample,
- d_1, the shorter sampling interval length,
- m, the integer multiple that relates d_1 the shorter sampling interval length to d_2 the longer sampling interval length,
- σ_ε^2, the observation error variance component,
- t', the total production run time (in months, days, hours, minutes etc.), and
- v′, the number of elements in the two conditional variance sets $[v_j]_{k'}$ and $[u_j]_{k'}$ that can be used as good approximations to the sets $[v_j]_k$ and $[u_j]_k$ for $k' \le k \le n$, where k′ is the number of the stage at which the approximation begins.

The output of the computer program consists of the adjustment limits and the sampling limits at each stage and the total expected loss to an n+1 stage problem.

4.2.2 Iterative procedure

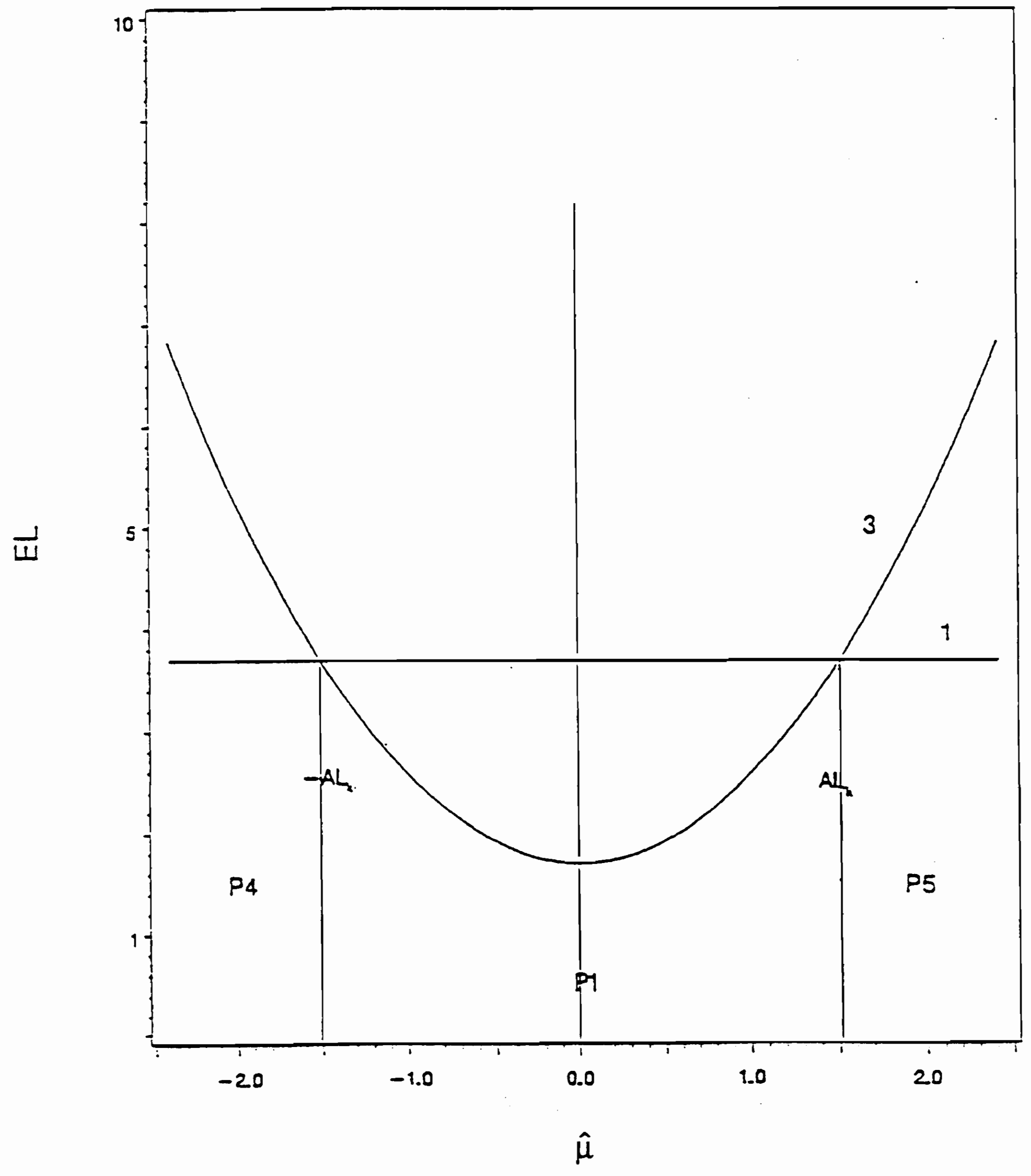
As described in Section 3.5 the technique of dynamic programming is used to calculate the adjustment limits, sampling limits, and the expected loss for an n+1 stage problem. The computational method consists of 4 steps corresponding to the four steps described in Section 3.5. Of the four steps, Step 2 and Step 4 are iterative steps. If the integer multiple m that relates d_1 and d_2 is 2, that is, $d_2 = 2*d_1$, then Step 2 will not be used .

Step 1: Last adjustment strategy

The last adjustment limits to an n+1 stage problem are the adjustment limits to the 1-period problem. The conditional variance v_{n-1} in (3.5.1) represents an element of the set $[v_j]_{k'}$. For each element of $[v_j]_{k'}$ the positive adjustment limit to the 1-period problem is given by

$$AL_{n-1} = \sqrt{c_2 / c_1 d_1}\,.$$

For each element of $[v_j]_{k'}$ the general shape of the expected loss equations corresponding to the two decisions for the 1-period problem is shown in Figure 4.2.1. The general shape of the function for the decision $\overline{A}_{n-1}S_{d_1}$ is given by curve 3 and that for the decision $A_{n-1}S_{d_1}$ is given by curve 1. Let the points of intersection of the two functions in Figure 4.2.1 be denoted by $\pm AL_{n-1}$. The lower envelope in the figure determined by the adjustment limits denotes the general shape of $\Re_{n-1,n}$, the minimum expected loss function for the 1-period problem. At this step the minimum expected loss function $\Re_{n-1,n}$ consists of two constant portions and a parabolic portion. The constant portions P4, P5, and the parabolic portion P1 are depicted in Figure 4.2.1. The value of the function in



1 - adjust and use the shorter sampling interval

3 - do not adjust and use the shorter sampling interval

Figure 4.2.1 General Shape of the Expected Loss Functions

(n-k)-period problem, $n-m+1 \leq k \leq n-1$

the constant portion which we will refer to as the constant functional value, is stored for use in the calculations that follow. The constant functional value is calculated at AL_{n-1}. For the parabolic portion, based on 48 Gaussian quadrature points, a grid of values for $\hat{\mu}_{n-1}$ are obtained between $-AL_{n-1}$ and AL_{n-1}. Next, the functional values at each of the grid of values are calculated and stored. These initial functional values will be used in the iterative computations that follow.

Step 2: kth adjustment strategy, $m > 2$ and $(n-m+1) \leq k \leq (n-2)$

For $m > 2$ and $(n-m+1) \leq k \leq (n-2)$ the minimum expected loss $\Re_{k,n}$ is computed from $\Re_{k+1,n}$ and AL_{k+1}. As stated earlier in Section 3.5, similar to Step 1 there are two decisions of interest at this step namely, $\overline{A}_k S_{d_1}$ and $A_k S_{d_1}$. The conditional variance v_k in equations (3.5.3), (3.5.4) and (3.5.5) of Sectiom 3.5 represent an element of $[v_j]_{k'}$ if $k > k'$ and represent an element of the actual set $[v_j]_k$ if $k \leq k'$. Using the Gaussian quadrature approximation method and the functional values stored at stage k+1, the expected loss functions are recursively calculated for decisions $\overline{A}_k S_{d_1}$ and $A_k S_{d_1}$ for each of the elements in $[v_j]_{k'}$ or $[v_j]_k$. Also, the adjustment limits computed at stage k are denoted by $\pm AL_k$. These adjustment limits are given by the points of intersection of the expected loss functions for the decisions $\overline{A}_k S_{d_1}$ and $A_k S_{d_1}$. In the computer program, the adjustment limits $\pm AL_k$ are computed using the bisection algorithm (Mathews (1992)).

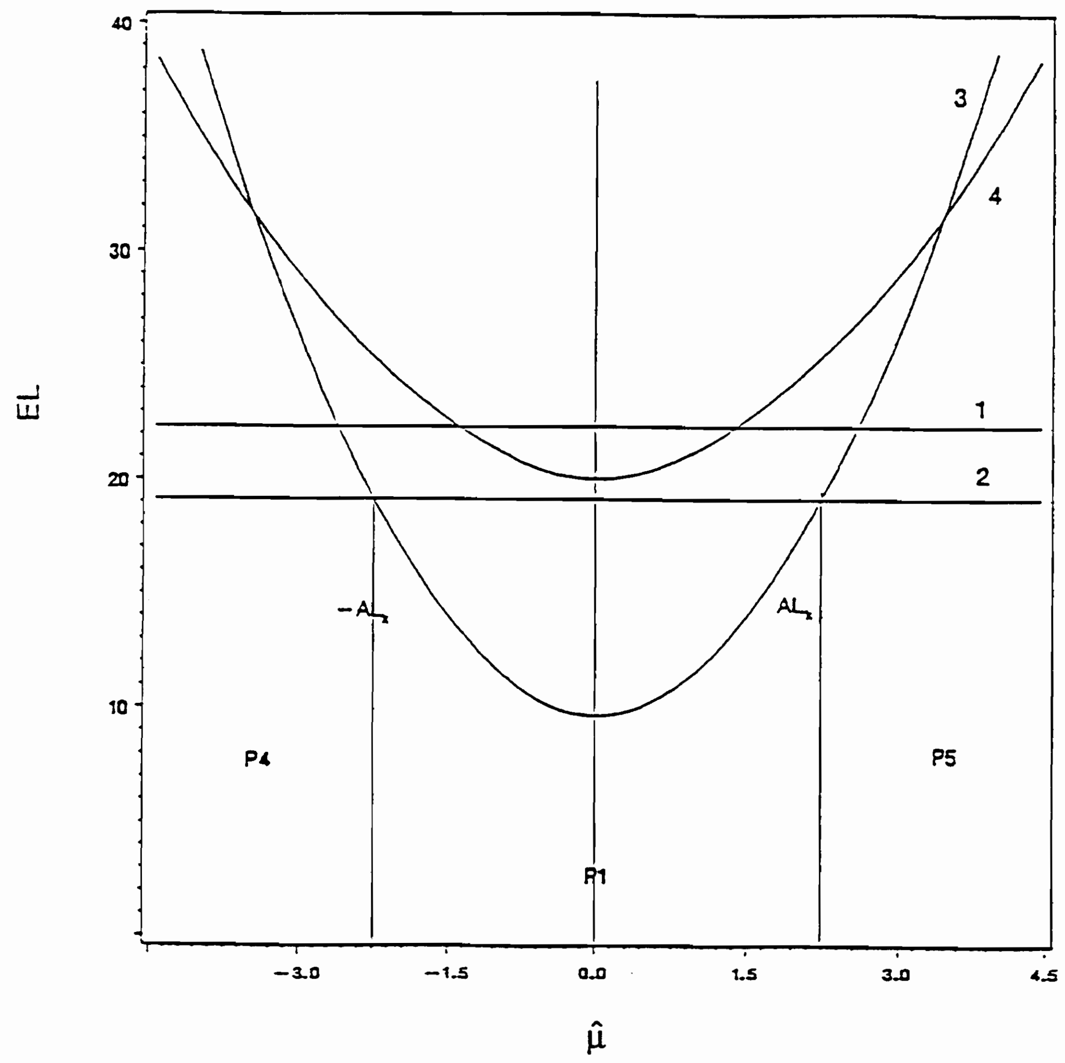
The general shape of the expected loss equations for the two decisions look similar to that shown in Figure 4.2.1. The curves corresponding to the decisions $\overline{A}_k S_{d_1}$ and $A_k S_{d_1}$ look similar to curve 3 and curve 1 of Figure 4.2.1 respectively. Also, the shape of $\Re_{k,n}$, the minimum expected loss function for the (n-k)-period problem where $m > 2$ and $(n-m+1) \leq k \leq (n-2)$ resembles the lower envelope in the figure determined by the appropriate adjustment limits. At this step the minimum expected loss function $\Re_{k,n}$ consists of two constant portions and a nonconstant portion. The constant portions P4 and P5, and the nonconstant portion P1 are depicted in Figure 4.2.1. The constant functional value is stored for the next iterative calculation. Since the adjustment limit AL_k is obtained as an approximate value, the constant functional value calculated at AL_k would be approximate too. In order to obtain the exact constant functional value, it will have to be calculated at some point > AL_k, which has been taken to be $AL_k+5.0$ in this program. For the middle nonconstant portion, based on the 48 quadrature points, a grid of values for $\hat{\mu}_k$ from $-AL_k$ to AL_k are obtained . The functional values at each of the grid of values are calculated and stored. These functional values will be used in the iterative computations that follow. When m = 2 Step 2 will not be used and Step 3 will follow Step 1.

Step 3: kth adjustment and sampling strategies, $m \geq 2$ and $k = (n\text{-}m)$

As stated in Section 3.5, the very last sampling limits are computed for the m-period problem, where $m \geq 2$. We saw earlier in Section 3.5 that there are four expected loss functions corresponding to the four possible decisions, namely and $\overline{A}_k S_{d_1}$, $A_k S_{d_1}$, $\overline{A}_k S_{d_2}$, and $A_k S_{d_2}$. The expected loss functions $\overline{A}_k S_{d_1}$ and $A_k S_{d_1}$ are computed using the functional values stored at stage k+1 for each element in $[v_j]_{k'}$ if $k > k'$ and each element of the actual set $[v_j]_k$ if $k \leq k'$. The initialization for the expected loss functions for the decisions $\overline{A}_k S_{d_2}$ and $A_k S_{d_2}$ start at Step 3.

Since there are four decisions to be considered at this step, we shall explore the types of behavior of the four expected loss functions with respect to each other. First we shall study the different types of possible relevant behavior and then proceed to look at the types of impossible behavior of the four expected loss functions. Figures 4.2.2-(i) through 4.2.2-(vi) portray the different types of possible relevant behavior of the four expected loss functions. In the Figures 4.2.2-(i) through (vi), curves 1, 2, 3 and 4 represent the decisions $A_k S_{d_1}$, $A_k S_{d_2}$, $\overline{A}_k S_{d_1}$, and $\overline{A}_k S_{d_2}$ respectively. In (i) of Figure

4.2.2 the nonconstant portion P1 and the constant portions P4 and P5 are determined by certain points of intersection of the expected loss functions; these points of intersection define the lower envelope to the expected loss functions. The lower envelope denotes $\mathfrak{R}_{k,n}$, the minimum expected loss function to the m-period problem. The nonconstant portion P1 represents the decision $\overline{A}_k S_{d_1}$ and the two constant portions P4 and P5 represent the decision $A_k S_{d_2}$. In Figure 4.2.2-(i) the sampling limit is 0. We also note that the points of intersection that define $\mathfrak{R}_{k,n}$, the lower envelope of the minimum expected loss function does not change if curve 4 does not intersect curve 1. Next, in (ii) of Figure 4.2.2 the nonconstant portion P1 represents the decision $\overline{A}_k S_{d_1}$ and the two constant portions P4 and P5 represent the decision $A_k S_{d_1}$. Here the sampling limit is 0 and d_1 is used as the next sampling interval. Moreover, the points of intersection that define the lower envelope to the minimum expected loss function $\mathfrak{R}_{k,n}$, does not change if curve 4 does not intersect curve 2. In (iii) of Figure 4.2.2 the nonconstant portion P1 represents the decision $\overline{A}_k S_{d_2}$ and the two constant portions P4 and P5 represent the decision $A_k S_{d_2}$. Here the sampling limit is ∞ and d_2 will be used as the next sampling interval. Furthermore, the points of intersection that define $\mathfrak{R}_{k,n}$, the lower envelope to the minimum expected loss function does not change if curve 3 does not intersect curve 1. In (iv) of Figure 4.2.) the nonconstant portion P1 represents the decision $\overline{A}_k S_{d_2}$ and the two constant portions P4 and P5 represent the decision $A_k S_{d_1}$. Once again the points of intersection that define $\mathfrak{R}_{k,n}$, the lower envelope to the minimum expected loss function does not change if curve 3 does not intersect curve 2. Next, unlike (i) through (iv) of Figure 4.2.2 each of which have only one nonconstant portion, (v) and (vi) of Figure 4.2.2 have 3 nonconstant portions represented by P1, P2 and P3. At stage k the 3 nonconstant portions are defined by two sets of limits, namely the sampling and



1 - adjust and use the shorter sampling interval

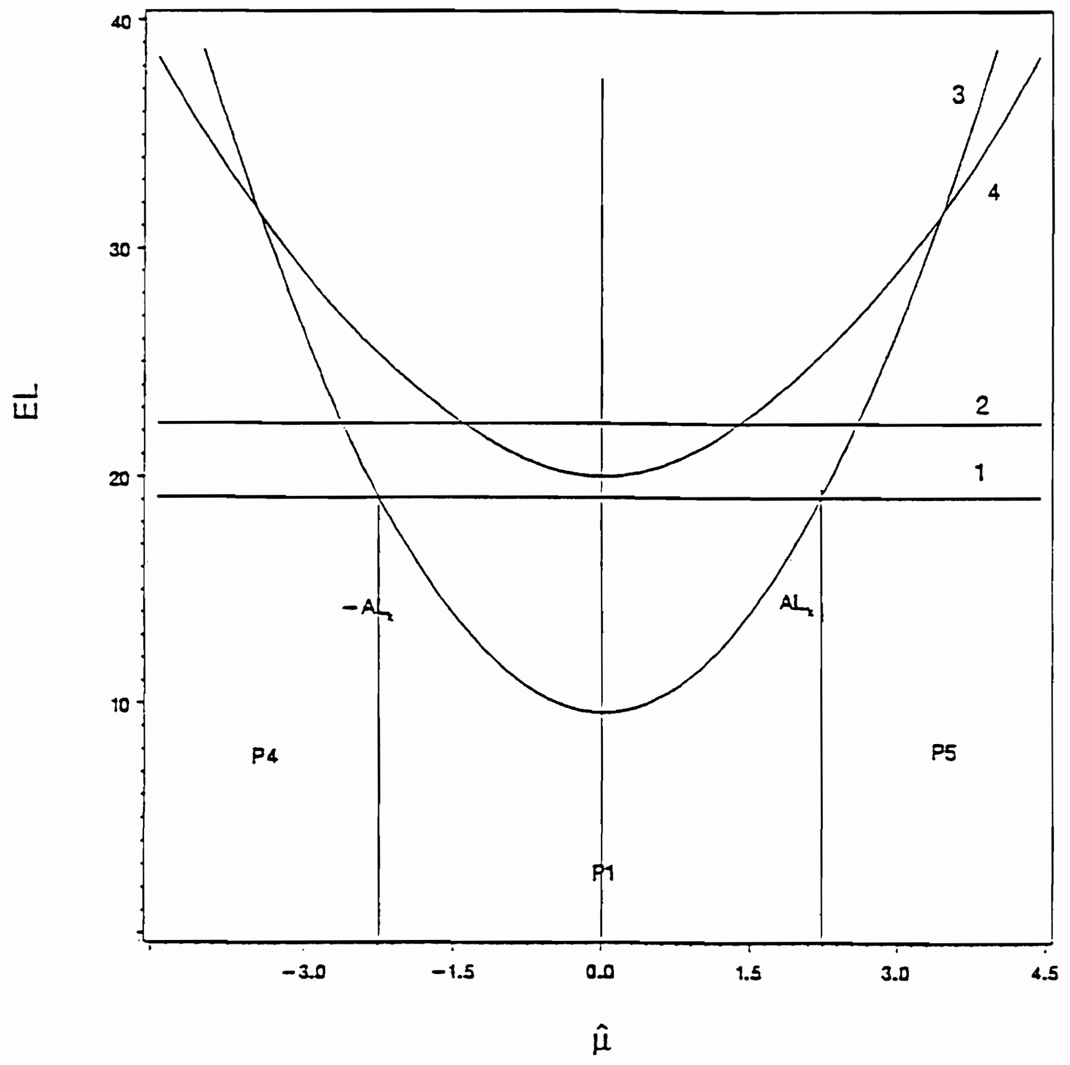
2 - adjust and use the longer sampling interval

3 - do not adjust and use the shorter sampling interval

4 - do not adjust and use the longer sampling interval

Figure 4.2.2 - (i) General Shape of the Expected Loss Functions

(n-k)-period problem, $0 \leq k \leq$ n-m



1 - adjust and use the shorter sampling interval

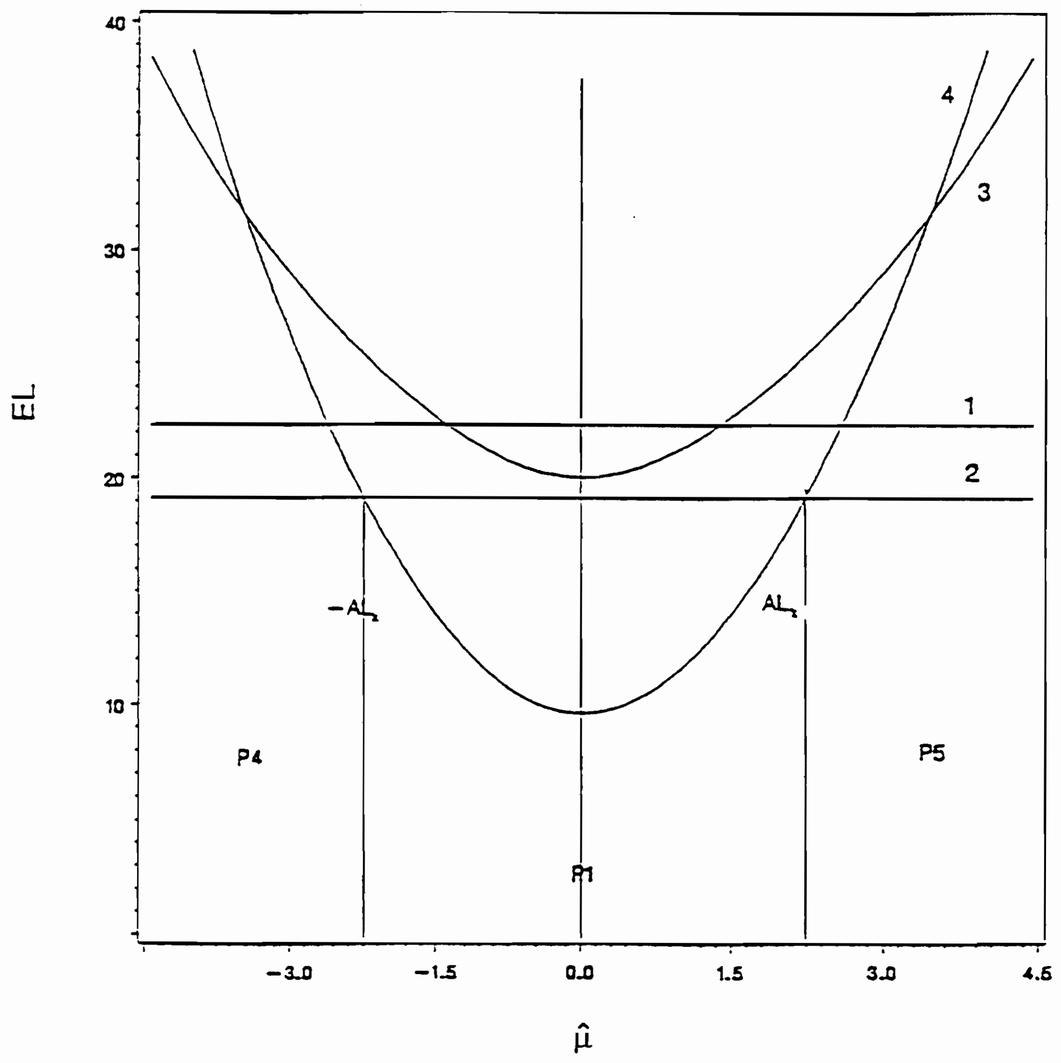
2 - adjust and use the longer sampling interval

3 - do not adjust and use the shorter sampling interval

4 - do not adjust and use the longer sampling interval

Figure 4.2.2 - (ii) General Shape of the Expected Loss Functions

(n-k)-period problem, $0 \leq k \leq n\text{-}m$



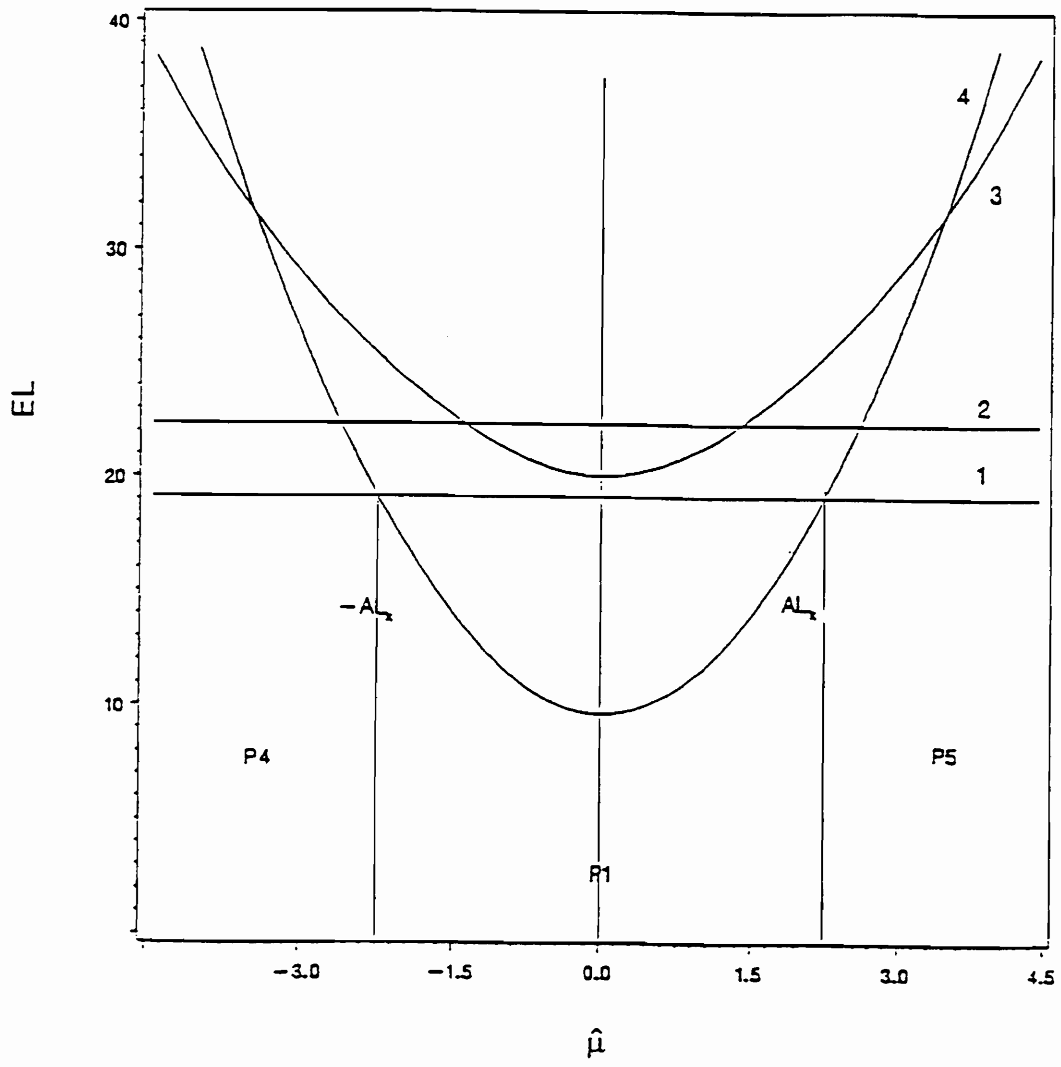
1 - adjust and use the shorter sampling interval

2 - adjust and use the longer sampling interval

3 - do not adjust and use the shorter sampling interval

4 - do not adjust and use the longer sampling interval

Figure 4.2.2 - (iii) General Shape of the Expected Loss Functions (n-k)-period problem, $0 \leq k \leq n\text{-}m$



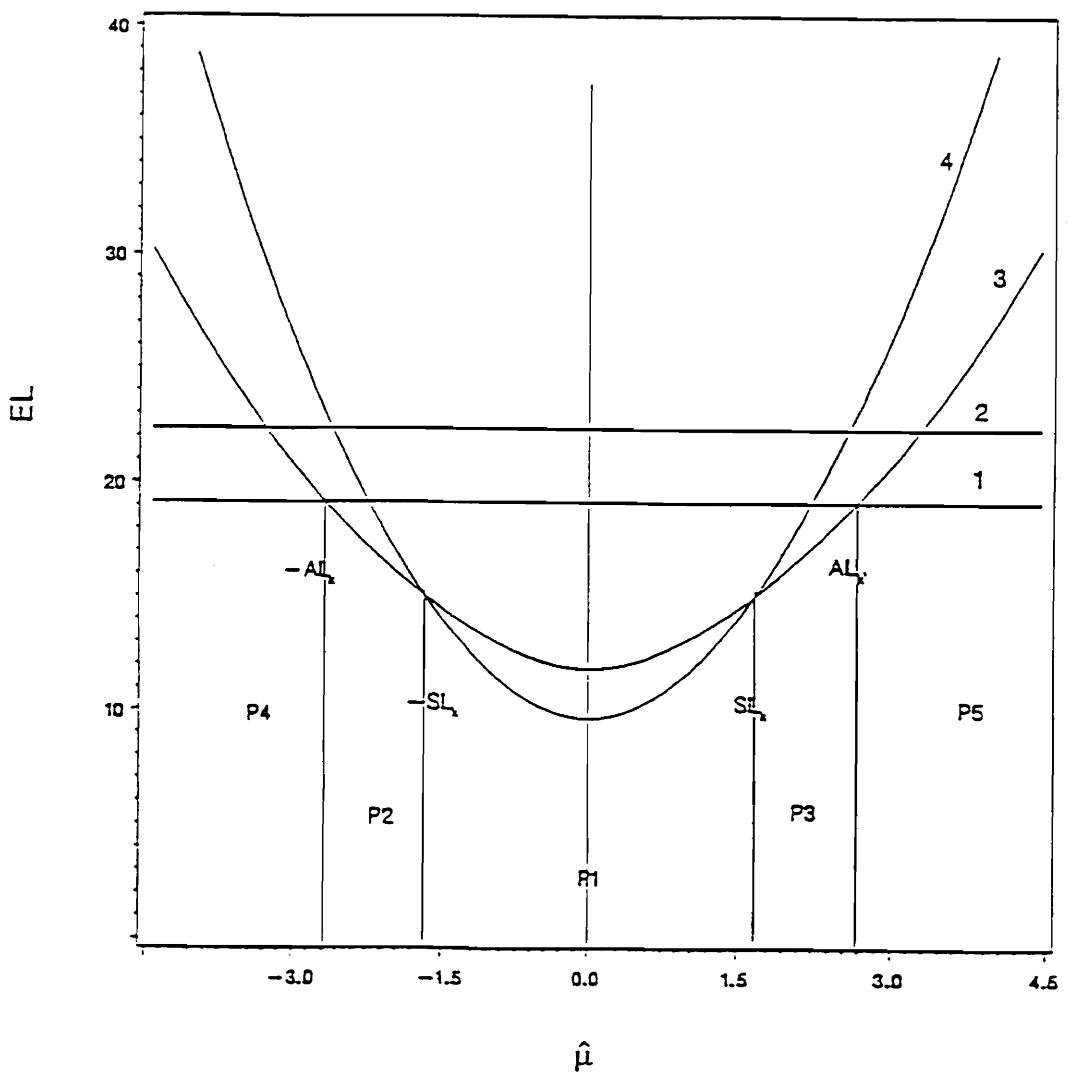
1 - adjust and use the shorter sampling interval

2 - adjust and use the longer sampling interval

3 - do not adjust and use the shorter sampling interval

4 - do not adjust and use the longer sampling interval

Figure 4.2.2 - (iv) General Shape of the Expected Loss Functions (n-k)-period problem, $0 \leq k \leq n\text{-}m$



1 - adjust and use the shorter sampling interval

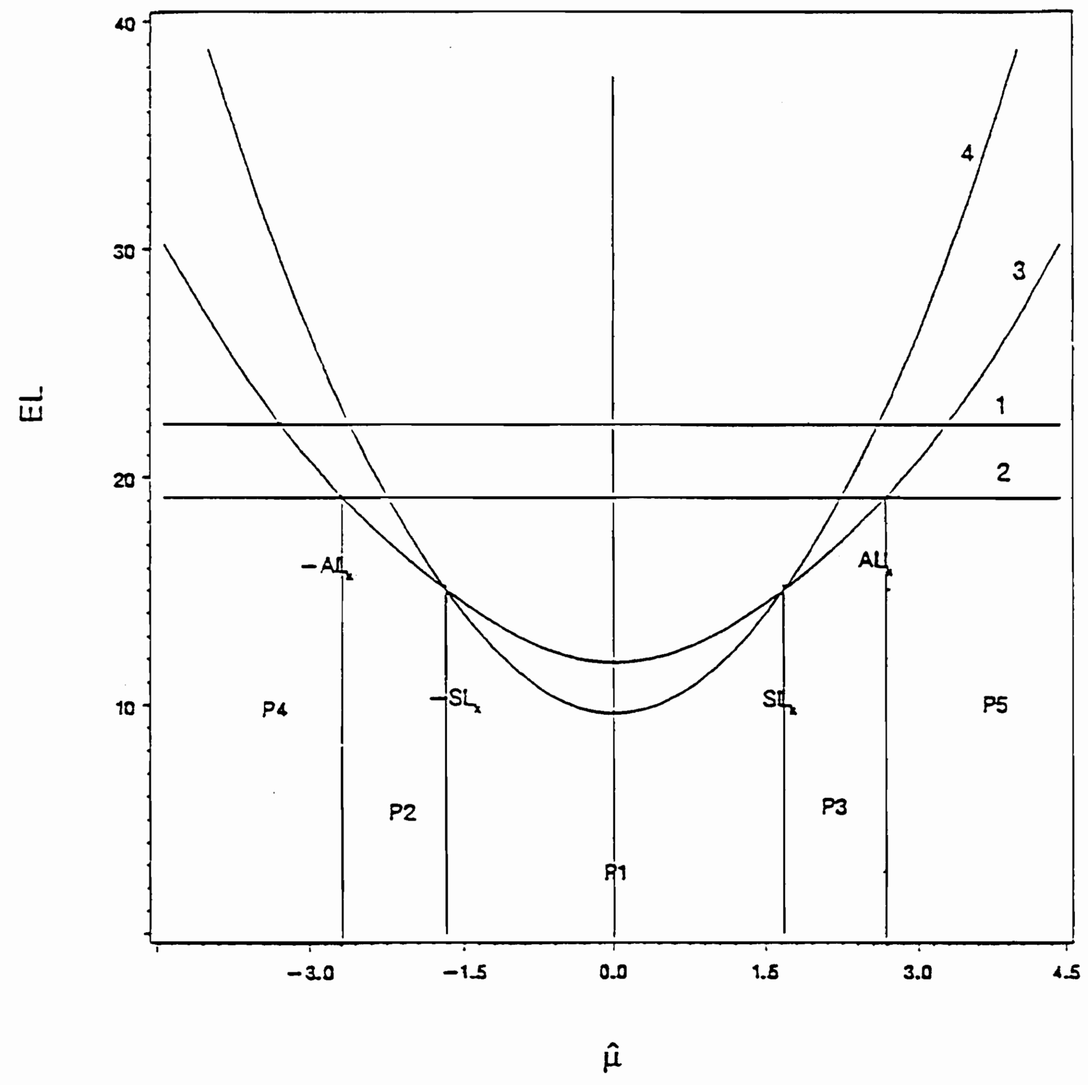
2 - adjust and use the longer sampling interval

3 - do not adjust and use the shorter sampling interval

4 - do not adjust and use the longer sampling interval

Figure 4.2.2 - (v) General Shape of the Expected Loss Functions

(n-k)-period problem, $0 \leq k \leq n\text{-}m$



1 - adjust and use the shorter sampling interval

2 - adjust and use the longer sampling interval

3 - do not adjust and use the shorter sampling interval

4 - do not adjust and use the longer sampling interval

Figure 4.2.2 - (vi) General Shape of the Expected Loss Functions (n-k)-period problem, $0 \leq k \leq n\text{-}m$

adjustment limits which are denoted by SL_k and AL_k respectively. The lower envelope determined by SL_k and/or AL_k denotes $\Re_{k,n}$, the minimum expected loss function to the m-period problem. Next, in Figures 4.2.2-(v) and (vi) if the curves 4 and 3 are interchanged, then it implies the following: (1) if there is indication that the process is operating close to target, then one would use the shorter sampling interval and will not adjust the process, and (2) if there is indication that the process has wandered away from the target but not far enough to call for an adjustment, then the longer sampling interval will be used. Intuitively, the above two implications with respect to sampling are contradictory to the idea of the VSI scheme. Therefore, we assume that it is not possible to have the roles of curves 4 and 3 interchanged in Figures 4.2.2-(v) and (vi). Further, in all of the Figures (i) through (vi) it is possible that the constant functions curve 1 or curve 2 do not intersect both the nonconstant functions represented by curves 3 and 4. Such a possibility could arise when the adjustment cost coefficient is zero.

By calculating the four integrated expected loss functions at $\hat{\mu}_k = 0$ one can tell which one of the above listed possibilities occur at each stage. Depending on which one of the above possibilities occur, the appropriate functional values are calculated and stored at each stage. For each of the nonconstant portions, a grid of values for $\hat{\mu}_k$ is obtained between the appropriate sampling/adjustment limits. The respective functional values at each of the grid of values are computed and stored. For the constant portions, the functional value at AL_k+5.0 is stored. These functional values will be used in the iterative computations that follow.

Step 4: kth adjustment and sampling strategies, $m \geq 2$ and $0 \leq k \leq (n-m-1)$

As described in Section 3.5, in this iteration step the adjustment and sampling limits are found recursively for the (n-k)-period problem, where $0 \leq k \leq (n-m-1)$. There are four equations for the expected loss functions corresponding to the four decisions namely,

$\overline{A}_k S_{d_1}$, $A_k S_{d_1}$, $\overline{A}_k S_{d_2}$, and $A_k S_{d_2}$. The expected loss functions $\overline{A}_k S_{d_1}$ and $A_k S_{d_1}$ are recursively computed using the functional values stored at step k+1 for each element in $[v_j]_{k'}$ if $k > k'$ and each element of the actual set $[v_j]_k$ if $k \leq k'$. The expected loss functions $\overline{A}_k S_{d_2}$ and $A_k S_{d_2}$ are recursively computed using the functional values stored at step k+m for each element in $[v_j]_{k'}$ if $k > k'$ and each element of the actual set $[v_j]_k$ if $k \leq k'$. Similar to that of Step 3, the types of behavior of the four expected loss functions could be portrayed by any one of the possible graphs described in the previous step. The constant and nonconstant portions in Figures 4.2.2-(i) through (vi) are determined by the points of intersection SL_k and/or AL_k of the expected loss functions; these points of intersection define the lower envelope to the (n-k)-period problem. The lower envelope determined by SL_k and/or AL_k denotes $\Re_{k,n}$, the minimum expected loss function to the (n-k)-period problem. One can recursively compute the sampling, adjustment limits and the minimum expected loss function for a (n-k)-period problem using information from stages k+1 and k+m. Thus using the backward induction method one can eventually obtain the minimum expected loss function $\Re_{0,n}(x)$ to an n+1 stage problem. The total minimum expected loss to an n+1 stage problem is then given by

$$\int \Re_{0,n}(x) f(\hat{\mu}_0 | \ldots, x_{-2}, x_{-1}, x_0) dx,$$

where $f(\hat{\mu}_0 | \ldots, x_{-2}, x_{-1}, x_0)$ is the normal density with mean zero and variance $\frac{(v_0+\sigma_v^2)^2}{v_0+\sigma_v^2+\sigma_\varepsilon^2}$. The vector of observations $(\ldots, x_{-2}, x_{-1}, x_0)$ represents the past information obtained up to stage 0. The above integral is calculated as a sum of the integral values for the constant and the nonconstant portions. The integrals for the nonconstant portions use the appropriate previously calculated sampling/adjustment limits and previously stored functional values in their calculations. The integrals are then computed using the Gaussian quadrature approximation. For the constant portion the integral is calculated to be the constant functional value times the approximate tail probability for the above given normal distribution. Therefore, given a set of cost combination, a VSI scheme and the observation error to an n+1 stage problem, one can calculate the adjustment and sampling limits at each stage and the total minimum expected loss to an n+1 stage problem.

Chapter 5: Results

5.1 Introduction

In this chapter numerical examples comparing the performance of the VSI scheme with that of the FSI scheme are given. In Section 5.2 with the help of a numerical example the roles of the Kalman filter estimator, adjustment and sampling limits in the solution to the problem are described. Section 5.3 compares the performance of the VSI scheme using two sampling intervals with that of the FSI scheme. Section 5.4 concludes the chapter with the study of the effects of the cost combinations and the observation errors on the adjustment limits, sampling limits, and the sampling interval lengths.

5.2 Adjustment and Sampling strategies - computational solution

With a numerical example, the computations of the adjustment and sampling limits are explained in this section. The role of the Kalman filter estimator in the final solution to the problem is also discussed.

Let the cost coefficients assume the following values:

off-target cost coefficient $c_1 = 1$,

adjustment cost coefficient $c_2 = 2$,

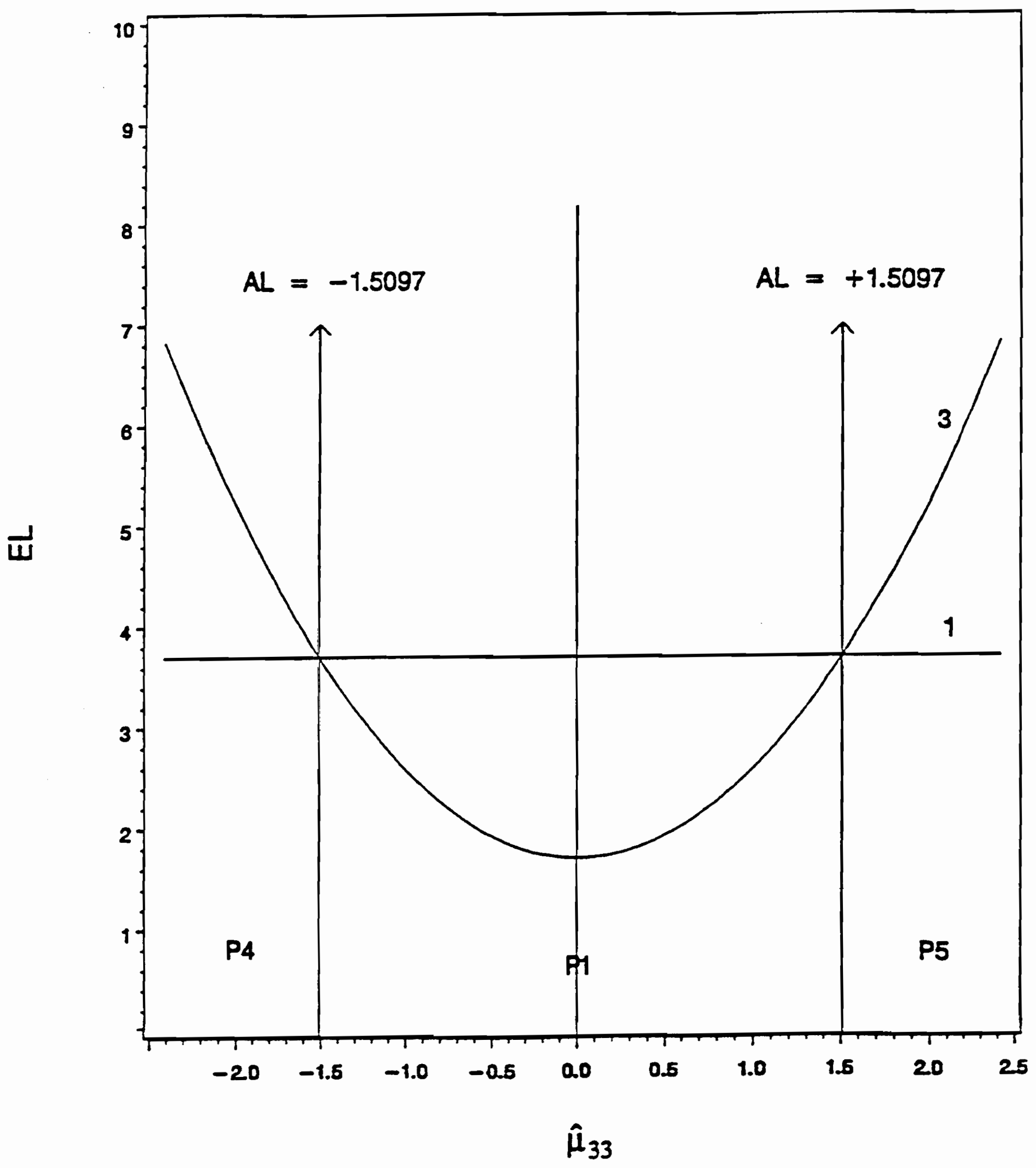
and

sampling cost coefficient $c_3 = 1$.

The above cost coefficients are interpreted as follows: the cost associated with making an adjustment is twice that of being off-target and that associated with taking a sample. As stated earlier without loss of generality the target can be assumed to be zero and the system error $\sigma_v^2=1$. In this example, let $t' = 30$, $\sigma_\varepsilon^2 = 0.5$, the shorter sampling interval $d_1 = 0.8825$ units, and the longer sampling interval $d_2 = 1.765$ units. Note that the multiple m that relates d_2 to d_1 is 2. Since $t' = 30$ and $d_1 = 0.8825$ the total number of stages in the problem that represent the possible sampling points of d_1 units apart is $n = 34$.

Figure 5.2.1 shows the plot of the expected loss functions for the 1-period problem. The functions are seen to be symmetric in $\hat{\mu}_{n-1}$. The two functions plotted are for the decisions when an adjustment is made denoted by curve 1 and when an adjustment is not made denoted by curve 3. The points of intersection of the functions corresponding to the two decisions are given by the last adjustment limits denoted by ± 1.5097. The band outside the adjustment limits are called the adjustment bands denoted by P4 and P5 and the band inside the limit is called the nonadjustment band denoted by P1. Whenever the computed value of the Kalman filter estimator $\hat{\mu}_{33}$ falls inside the band (-1.5097, 1.5097), no adjustment is called for. On the other hand, if $\hat{\mu}_{33}$ falls outside the band (-1.5097, 1.5097), then an adjustment equivalent to $-\hat{\mu}_{33}$ is called for. The adjustment limits define the minimum expected loss for the 1-period problem. It is assumed that no sample will be taken at stage 34; stage 34 being the very last stage of the production run.

Since $d_2 = 2*d_1$, the sampling and adjustment limits are determined for the 2-period problem using information from Figure 5.2.1. The expected loss functions for the four decisions are shown in Figure 5.2.2. The four decisions are adjust and use $d_1=0.8825$ as the next sampling interval represented by curve 1, adjust and use $d_2=1.765$ as the next sampling interval represented by curve 2, do not adjust and use $d_1=0.8825$ as the next sampling interval represented by curve 3, and do not adjust and use $d_2=1.765$ as the next sampling interval represented by curve 4. The minimum envelope of the graphs is given by the regions P1, corresponding to the decision do not adjust and use $d_2=1.765$ as the next sampling interval and P4 and P5, corresponding to the decision adjust and use $d_2=1.765$ as the next sampling interval respectively. The adjustment band P1 is given by (-1.066, +1.066). Figure 5.2.2 corresponds to Figure 4.2.2-(iii) and so the sampling limit

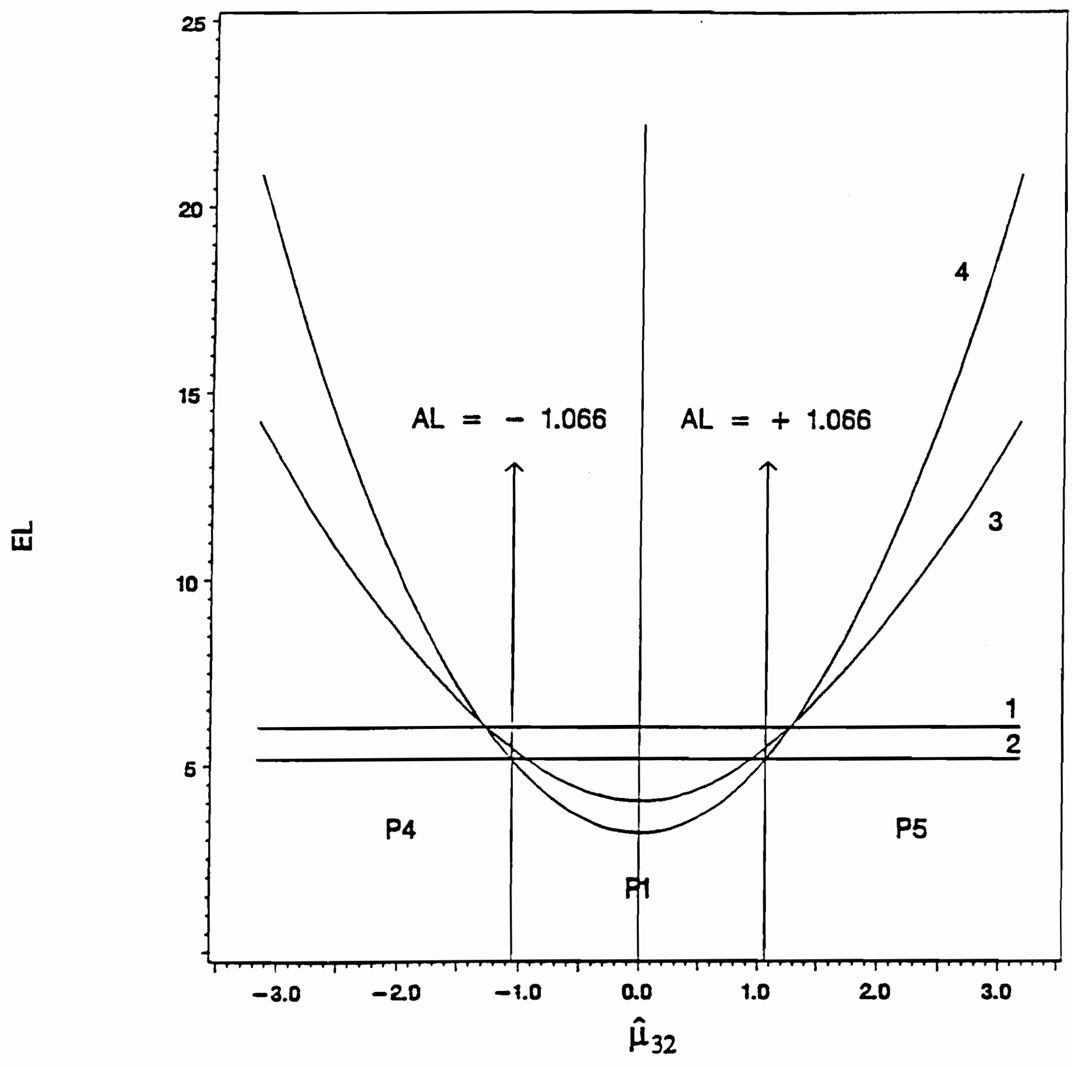


1 - adjust

3 - do not adjust

Figure 5.2.1 1-period problem

$c_1 = 1,\ c_2 = 2,\ c_3 = 1,\ \sigma_\varepsilon^2 = 0.5,\ t' = 30,\ n = 34$
$d_1 = 0.8825,\ d_2 = 1.765$



1 - adjust and use the shorter sampling interval

2 - adjust and use the longer sampling interval

3 - do not adjust and use the shorter sampling interval

4 - do not adjust and use the longer sampling interval

Figure 5.2.2 2-period problem

$c_1 = 1,\ c_2 = 2,\ c_3 = 1,\ \sigma_\varepsilon^2 = 0.5,\ t' = 30,\ n = 34$
$d_1 = 0.8825,\ d_2 = 1.765$

is ∞. If the computed value of the Kalman filter estimator $\hat{\mu}_{32}$ falls outside the band specified by (-1.066, +1.066), then an adjustment equivalent to $-\hat{\mu}_{32}$ is called for, else no adjustment is made to the process. In either case the decision with respect to sampling is the same, namely to use the longer sampling interval the next time.

Table 5.2.1 gives a list of adjustment and sampling limits for a 34 stage problem. In the table, SL and AL denote the corresponding sampling and adjustment limits at that stage. We have assumed that no sample will be taken at the final stage of the problem. As a result, there are no decisions to be made at this final stage and hence no limits. In our example (Table 5.2.1), since stage 34 is the final stage, we don't have any limits corresponding to stage 34. The calculation of the limits for stages 33 and 32 are described above using Figures 5.2.1 and 5.2.2 respectively. At stage 33 there are no decisions with regard to sampling since stage 33 is the second to the last stage. The sampling strategy for stages 31 is to always use the shorter sampling interval length the next time. For stage 32, the sampling strategy is to always use the longer sampling interval length the next time. In order to understand the decisions for the remaining stages let us consider an example. Figure 5.2.3 displays the expected loss functions for a 5-period problem (stage=29, n=34). The sampling limits $\pm SL_{29}$ are given by ± 0.5377. The adjustment limits $\pm AL_{29}$ are given by ± 1.2162. Region P1 corresponds to the decision do not adjust and use the longer sampling interval, regions P2 and P3 correspond to the decision do not adjust and use the shorter sampling interval, and regions P4 and P5 correspond to the decision adjust and use the longer sampling interval. If the computed value of the Kalman filter estimate $\hat{\mu}_{29}$ falls inside (-0.5377, 0.5377), then the decision is do not adjust and use $d_2=1.765$. If $\hat{\mu}_{29}$ falls in either (-1.2162, -0.5377) or (0.5377, 1.2162), then the decision is do not adjust and use the shorter sampling interval, $d_1 = 0.8825$. Lastly, if $\hat{\mu}_{29}$ falls outside (-1.2162, 1.2162), then the decision is adjust and use

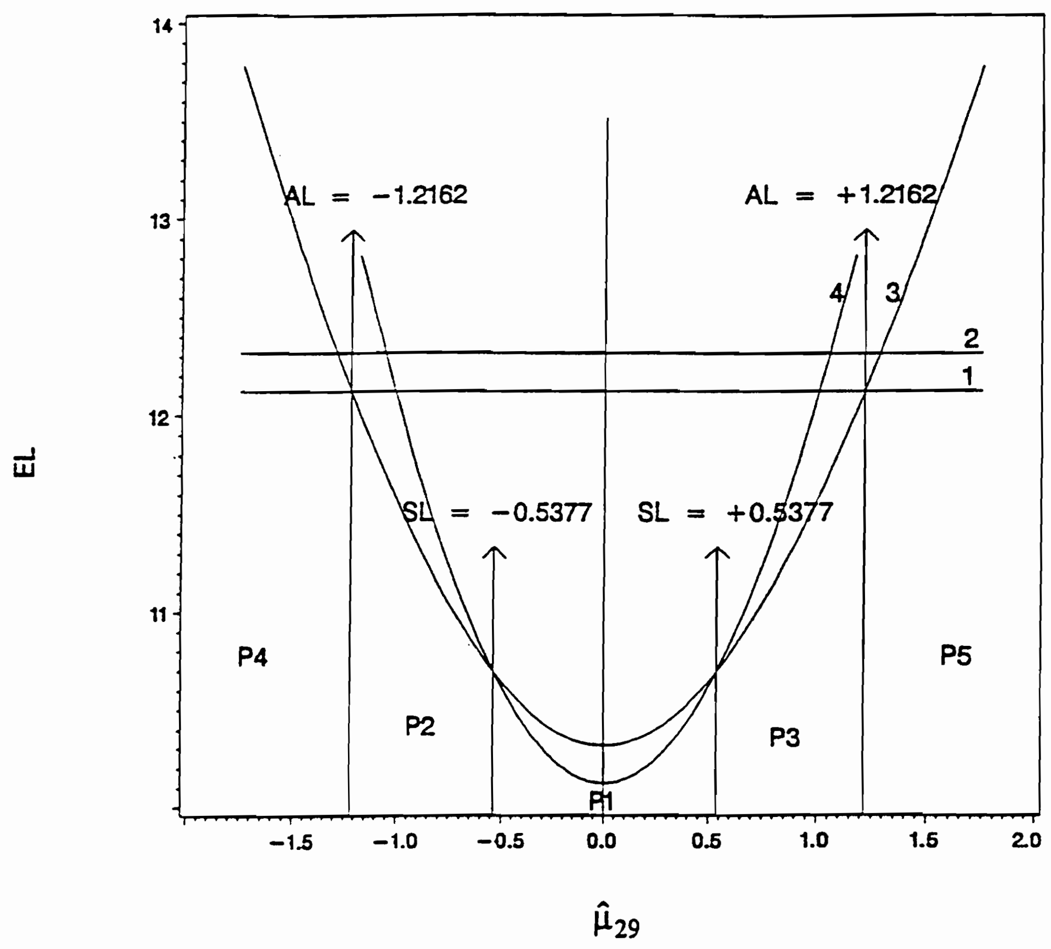
Table 5.2.1

Sampling and Adjustment Limits - Example

$c_1 = 1$, $c_2 = 2$, $c_3 = 1$; $\sigma_\varepsilon^2 = 0.5$, $t' = 30$, $n = 34$

$d_1 = 0.8825$, $d_2 = 1.765$

Stage	SL	AL
34	-	-
33	-	1.5097
32	∞	1.0660
31	0	1.2804
30	1.0032	1.0194
29	0.5377	1.2162
28	0.9250	1.0665
27	0.6455	1.1852
26	0.8807	1.0895
25	0.7006	1.1664
24	0.8507	1.1041
23	0.7329	1.1545
22	0.8304	1.1136
21	0.7529	1.1467
20	0.8166	1.1198
19	0.7656	1.1416
18	0.8074	1.1239
17	0.7737	1.1383
16	0.8012	1.1266
15	0.7790	1.1361
14	0.7971	1.1284
13	0.7825	1.1346
12	0.7943	1.1296
11	0.7847	1.1337
10	0.7925	1.1303
9	0.7862	1.1330
8	0.7462	1.1495
7	0.7770	1.1367
6	0.7942	1.1292
5	0.7800	1.1356
4	0.7933	1.1300
3	0.7826	1.1345
2	0.7912	1.1309
1	0.7837	1.1342
0	0.7826	1.1356



1 - adjust and use the shorter sampling interval

2 - adjust and use the longer sampling interval

3 - do not adjust and use the shorter sampling interval

4 - do not adjust and use the longer sampling interval

Figure 5.2.3 5-period problem

$c_1 = 1,\ c_2 = 2,\ c_3 = 1,\ \sigma_\varepsilon^2 = 0.5,\ t' = 30,\ n = 34$
$d_1 = 0.8825,\ d_2 = 1.765$

the longer sampling interval d_2=1.765. An adjustment equivalent to - $\hat{\mu}_{29}$ is then called for. Thus, in general for stages 0 up to 30, if the Kalman filter estimate falls in the band ± SL, then the decision is do not adjust and use d_2 as the next sampling interval. Since there seems to be evidence that the process is running close to the target an adjustment is not called for. Moreover, in this case it is profitable to use the longer sampling interval instead of the shorter sampling interval. If the Kalman filter estimate falls in either(SL, AL) or (-AL, -SL),then the decision is do not adjust and use d_1 as the next sampling interval. This means that there is an indication that the process is not far enough from the target to call for an adjustment, but far enough to decide on using the shorter sampling interval the next time. On using the shorter sampling interval d_1, it is possible to capture the drift in the process (if any) sooner. Lastly, if the Kalman filter estimate falls outside ± AL, then the decision is adjust and use d_2 as the next sampling interval. When the Kalman filter estimate falls in the adjustment band, it is an indication that the process has drifted far enough from the target that an adjustment is required. An adjustment equivalent to the negative of the Kalman filter estimate at that stage is called for. Thus the adjustment is made and the longer sampling interval is used next time.

The sampling and adjustment limits of Table 5.2.1 are plotted in Figure 5.2.4. Figure 5.2.4 gives a better idea of the behavior of the adjustment and sampling limits for the given example. We notice that toward the end of the production run, the adjustment and the sampling limits display an oscillating behavior and this oscillating behavior almost washes out as one moves away from the end of the production run. This oscillation may be because the limits at any stage k are functions of the number of stages left in the problem. As one moves away from stage n, the number of stages left in the problem gets larger and so after a certain point the number of stages left in the problem have little effect on the limits. Also, there is an unexplainable blip at stage 8. Now, when the SL is

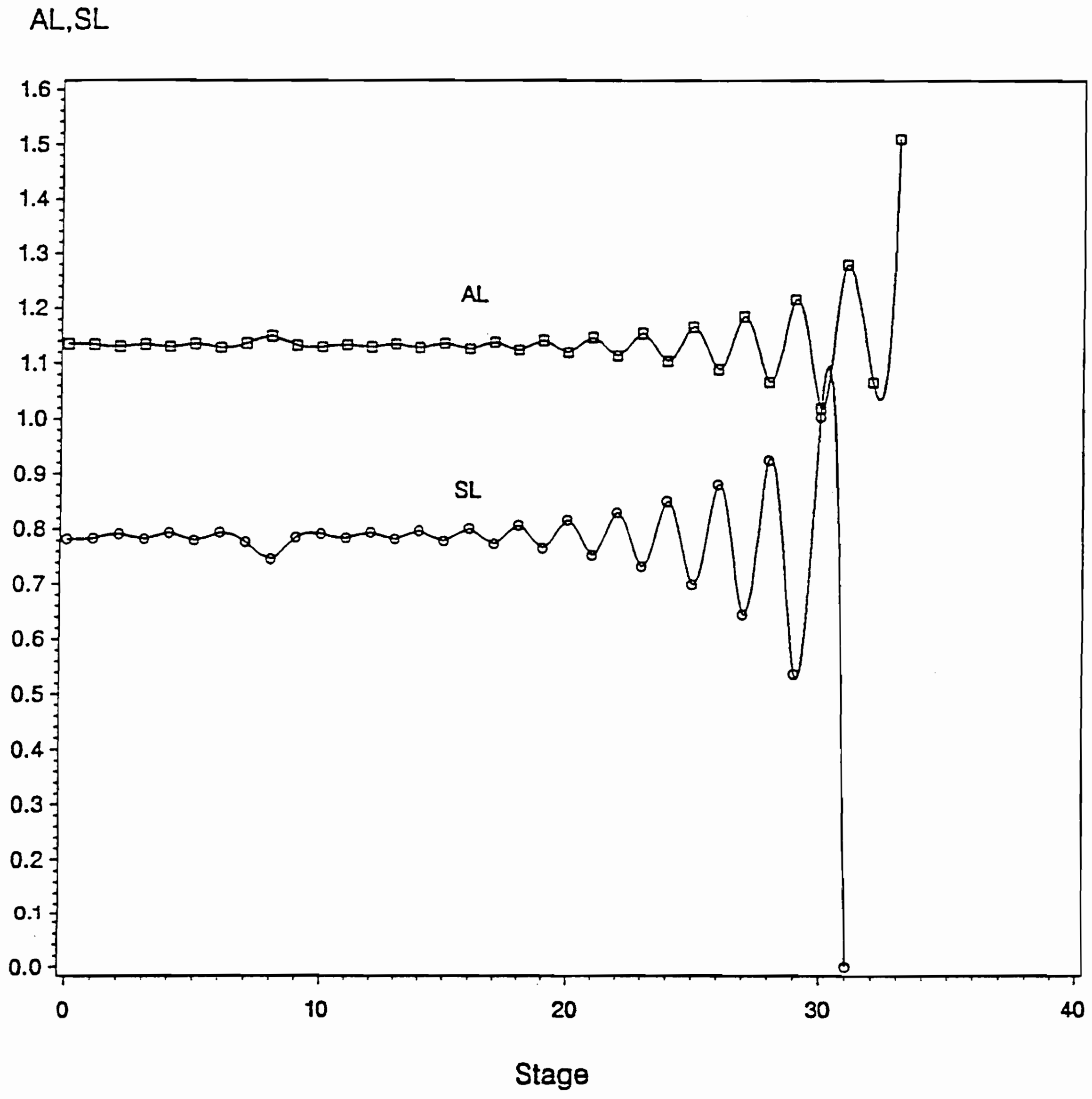


Figure 5.2.4 Graph of the Sampling and the Adjustment Limits

$c_1 = 1$, $c_2 = 2$, $c_3 = 1$, $\sigma_\varepsilon^2 = 0.5$, $t' = 30$, $n = 34$
$d_1 = 0.8825$, $d_2 = 1.765$

∞, the decision is to always use the longer sampling interval the next time and when the SL is zero, the decision is to always use the shorter sampling interval the next time. As one approaches the end of the production run, we see that whenever the SL is high, the corresponding AL is low. This could imply that the chance of using the shorter sampling interval the next time is small and so the chance of making an adjustment the current time increases. For example, we see that there is a dip in the AL from 1.5097 at stage 33 to 1.066 at stage 32. At stage 32, the decision is never to sample at stage 33. This implies that definitely no adjustment will be made at stage 33. Therefore, one would increase the probability of making an adjustment at stage 32. Next, we see that the AL increases from 1.066 at stage 32 to 1.2804 at stage 31. The decision with respect to sampling is to definitely sample at stage 32. This implies that there is a chance to make an adjustment at stage 32. So, one need not be too concerned about increasing the probability of making an adjustment at stage 31.

Crowder (1986) had not considered the sampling cost. His computer program uses the cost ratio of the adjustment cost to the off-target cost, namely c_2/c_1. He assumed that samples are taken from the process at fixed intervals of 1.0 unit apart. By assuming $c_3 = 0$ (zero sampling cost) in our program, a comparison with Crowder's results can be made. For any cost ratio c_2/c_1, the adjustment limits computed by our program show clear evidence of convergence. Crowder has obtained these limits analytically. Table 5.2.2 gives the convergent value of the adjustment limits as obtained by our program and the convergent value of the adjustment limits obtained analytically by Crowder. The results are seen to agree with one another up to 4 decimal points. The convergent adjustment limits can be used for problems involving a large number of stages. When the sampling cost is zero, the expected losses corresponding to decisions $\overline{A}_k S_{d_1}$ and $A_k S_{d_1}$ will always be less than the expected losses corresponding to decisions $\overline{A}_k S_{d_2}$, and

Table 5.2.2

Comparison with Crowder's Results
$d_1 = d_2 = 1.0$

Cost Ratio c_2/c_1	Adjustment Limits (from Crowder)	Adjustment Limits ($c_3 = 0$)
0.2	0.4425	0.4425
0.5	0.6832	0.6832
1	0.9283	0.9283
3	1.4321	1.4321
5	1.7176	1.7176
10	2.1659	2.1659
20	2.6966	2.6966
50	3.5537	3.5537
100	4.3447	4.3447
500	6.8029	6.8029

$A_k S_{d_2}$. This is because the trade off is only between the adjustment cost and the off-target cost. Therefore by using the longer sampling interval at certain times, a greater off-target cost may be incurred leading to a larger expected loss for the decisions that involve the longer sampling interval d_2. As a result, only adjustment limits are obtained. We notice that as the cost ratio c_2/c_1 increases, the adjustment limits increase. This is because one would be conservative in making adjustments when the cost of adjustment gets larger.

5.3 VSI vs. FSI

In this section we will compare the performance of the VSI scheme with that of the FSI scheme in terms of the total expected loss. We will first define two important terms, namely the optimal FSI and optimal VSI. Strictly speaking, what we refer to as optimal FSI and VSI in this dissertation is not optimal. Since we don't have an analytical proof to finding the optimal FSI and VSI, we shall refer to our results as the optimal results within the limits of our numerical calculations. Let the optimal FSI be defined to be <u>that</u> fixed sampling interval which gives the *least* expected loss among the expected losses calculated for a set of fixed sampling intervals. Further let the optimal VSI be defined to be <u>that</u> pair of (d_1, d_2) that gives the *least* expected loss among the expected losses calculated for a set of variable sampling intervals (d_1, d_2). Then the performance of the optimal VSI with that of the optimal FSI is compared in terms of the *least* expected loss. We will further see the effects of the costs on the percentage reduction in the expected loss when the optimal VSI scheme is used instead of the optimal FSI scheme. The percentage reduction gives a measure of the performance of the VSI scheme over that of the FSI scheme.

Before we compare the results of the VSI scheme with that of the FSI scheme, we will explain how the values for d' which is the fixed sampling interval length, were selected for the comparisons. Let us recall from Chapter 3 that if t'/d' is not an integer, then n, the number of stages in the problem is defined to be the integer part of t′/d′ plus one. As a result, in an n stage problem there would be one period of unequal length. Theoretically any period in an n stage problem can represent this period of unequal length. But for computational ease, we shall assume that this period of unequal length is the last period. However, we found that in finding the optimal d′, it is sufficient to consider only those values of d′ for which t′/d′ is an integer. This is because there is discontinuity in the expected loss when graphed as a function of t′/d′. This is illustrated in Figure 5.3.1. Curves 2 and 3 represent the expected losses for noninteger values of t′/d′. Curve 1 represents the expected losses for integer values of t′/d′ and the noninteger values of t′/d′ close to an integer. From Figure 5.3.1, we see that if one considered all values of t′/d′, then there is a point of discontinuity in the expected loss at the integer. For example, we see that the expected loss is continuous for values of t′/d′ in the interval (25,26], but there is discontinuity at the first t′/d′ > 26. But for integer values of t′/d′ and noninteger values of t′/d′ close to an integer, the curve is parabola-like. This implies that we need only to consider values of d′ for which t′/d′ is an integer thereby reducing the computation time in search of the optimal results. Now, let us consider an example to reason the cause for the discontinuities at the integer values of t′/d′ in the graph. The definition of the number of stages n implies that n is 27 between t′/d′ > 26 and t′/d′ = 27. This means that the number of possible sampling points is 27 for any d′ that gives rise to t′/d′ > 26 and < 27 and so there is one more sampling point available when n = 27 (for t′/d′ > 26 and < 27) than when n = 26. This results in an increased jump in the expected loss at n = 26. The amount of the increased jump in the expected loss at the integer would be a function of the sampling cost.

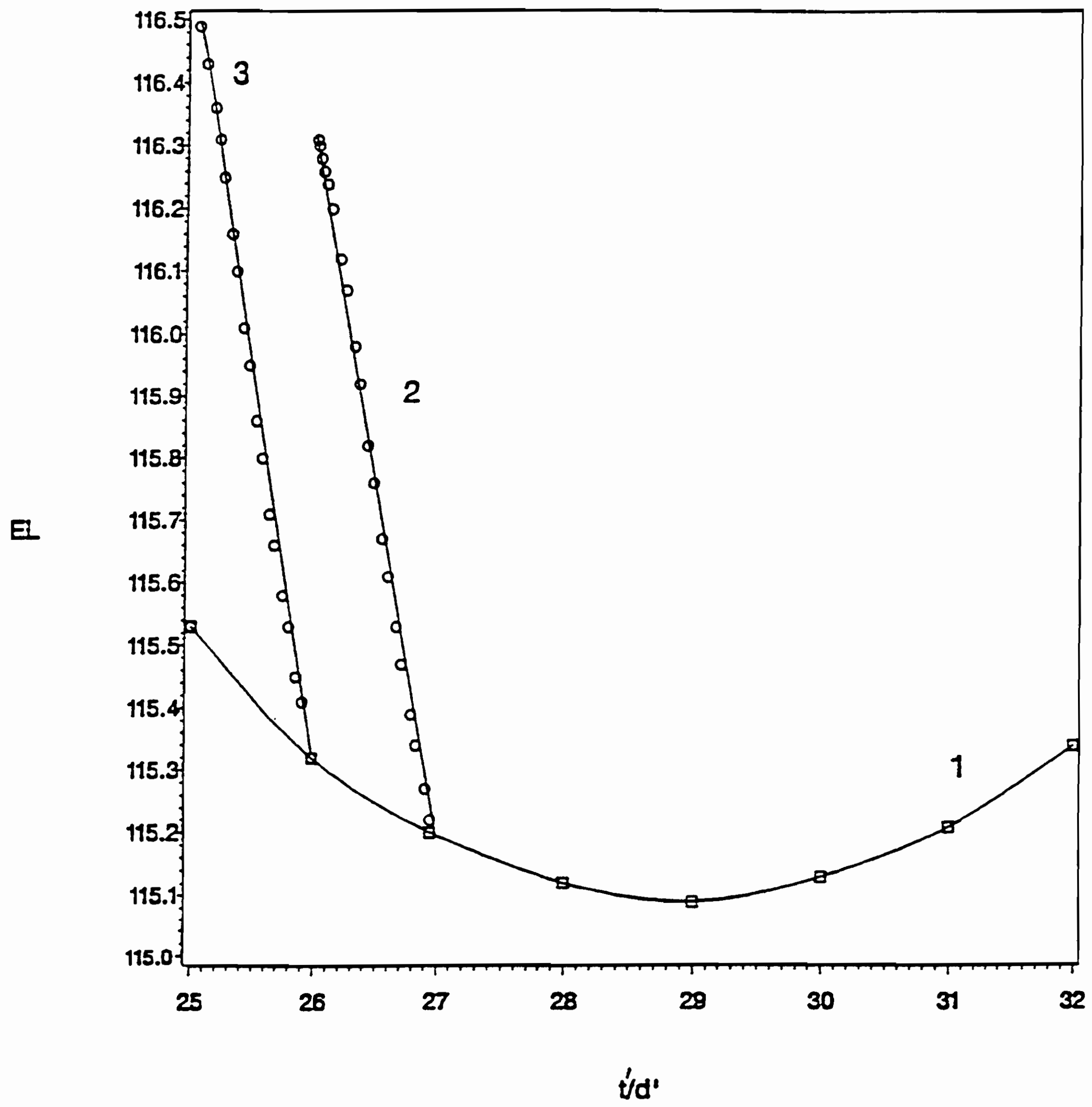


Figure 5.3.1 Graph of the Expected Loss as a Function of t′/d′

$c_1 = 1, c_2 = 1, c_3 = 1$

$\sigma_\varepsilon^2 = 5.0, t' = 30$

Table 5.3.1-(a)

Optimal FSI - Example
$c_1 = 1, c_2 = 1, c_3 = 1$
$\sigma_\varepsilon^2 = 5.0, t' = 30$

Number of Stages n	FSI with d′ units	Exp. Loss FSI
60	0.5000	128.61
50	0.6000	122.45
40	0.7500	117.61
32	0.9375	115.34
30	1.0000	115.13
25	1.2000	115.53
20	1.5000	118.00
15	2.0000	124.42

Table 5.3.1-(b)

Optimal FSI - Example
$c_1 = 1, c_2 = 1, c_3 = 1$
$\sigma_\varepsilon^2 = 5.0, t' = 30$

Number of Stages n	FSI with d′ units	Exp. Loss FSI
31	0.9678	115.21
30	1.0000	115.13
29	1.0350	115.09
28	1.0720	115.12
27	1.1120	115.20

We shall now explain how the optimal FSI and the optimal VSI are found using an example. We have used the trial and error method to find the optimal FSI and VSI. For the cost combination $c_1 = c_2 = c_3 = 1$, observation error $\sigma_\varepsilon^2 = 5.0$, and $t' = 30$, the expected loss for each of the different fixed sampling intervals with d' units is shown in Table 5.3.1-(a) and (b). The number of stages column refers to the possible number of sampling points or stages in the problem when samples of d' units are taken from a process whose total production time is 30 time units. First the total expected losses for some values of d' that give rise to integer values of n were calculated as shown in Table 5.3.1-(a). Among these values of the expected losses we see that the FSI corresponding to the least expected loss is for $d' = 1.0$ unit. So the expected losses for values of d' around $d' = 1.0$ unit were calculated as shown in Table 5.3.1-(b). Now it is seen from Tables 5.3.1-(a) and (b) that the expected loss corresponding to the FSI scheme with $d' = 1.035$ units is the least, namely 115.09. So for the example in Table 5.3.1-(a) and (b) the optimal FSI is found to be $d' = 1.035$ units. We also notice from Table 5.3.1-(a) and (b) that in the range of 0.94 to 1.2 for d', the expected loss is not very different from one another.

Next, we will use the optimal FSI as the starting point to find the optimal VSI. Once the optimal FSI scheme is found, the expected losses for certain combinations of (d_1, d_2) that sandwich the optimal FSI are computed. This is because preliminary work on finding the optimal FSI and VSI have shown that the optimal FSI falls between the optimal VSI of (d_1, d_2). For example, from Tables 5.3.1-(a) and (b), we see that the optimal FSI scheme is $d' = 1.035$ units. From Table 5.3.2, we see that the values of (d_1, d_2) sandwich the optimal FSI scheme of $d' = 1.035$ units. Among the values of (d_1, d_2) that sandwich the optimal FSI scheme, it is sufficient to consider those values of d_1 for which t/d_1 is an integer in finding the optimal (d_1, d_2) for the VSI expected loss

Table 5.3.2

Optimal VSI - Example
$c_1 = 1, c_2 = 1, c_3 = 1$
$\sigma_\varepsilon^2 = 5.0, t' = 30$

Number of Stages n	Shorter Interval d_1	Longer Interval d_2	Multiple m	Exp. Loss VSI
170	0.1765	1.4120	8	106.50
170	0.1765	1.5885	9	105.85
170	0.1765	1.7650	10	107.34
165	0.1819	1.6371	9	106.36
165	0.1819	1.8190	10	105.45
165	0.1819	2.0009	11	107.55
160	0.1875	1.5000	8	106.39
160	0.1875	1.6875	9	105.60
160	0.1875	1.8750	10	107.92
150	0.2000	1.0000	5	107.32
150	0.2000	1.2000	6	106.80
150	0.2000	1.4000	7	106.86

calculations. Using trial and error, we see that the combination of (d_1, d_2) that has the least expected loss is for d_1 = 0.1819 and d_2 = 1.8190 units. This is the optimal VSI scheme and the corresponding expected loss is 105.45. It is seen that when the optimal VSI scheme is used instead of the optimal FSI scheme the percentage reduction in the expected loss is calculated to be (115.09 - 105.45)/115.09 which is roughly 8.4%. Thus on comparing the expected loss for the optimal VSI scheme with that of the optimal FSI scheme, an improvement in the performance of the VSI scheme over that of the FSI scheme is seen. This shows that for this combination of the three costs and observation error, there is some improvement in the expected loss on using the optimal VSI scheme over that of the optimal FSI scheme.

Now we shall discuss the effects of the cost ratios and the observation errors on the percentage reduction of the VSI scheme over that of the FSI scheme. Let SCR be defined as the sampling cost ratio. It denotes the ratio of the sampling cost to the off-target cost. Let ACR be defined as the adjustment cost ratio. It denotes the ratio of the adjustment cost to the off-target cost. In Tables 5.3.3-(a) and (b) for certain cost combinations SCR and ACR, the percentage reduction in the expected loss of the optimal VSI scheme over the optimal FSI scheme is shown for the observation errors σ_ε^2 = 1.0 and σ_ε^2 = 5.0 respectively. The optimal fixed and variable sampling lengths are also given. In Table 5.3.3-(a) based on the percentage reduction in the expected loss of the optimal VSI scheme over the optimal FSI scheme, it is seen that the VSI scheme performs better than the FSI scheme for the case when the SCR is at the low level of 0.2. In Table 5.3.3-(b) a similar observation can be made for the case when the SCR is at the low level of 1.0. But for certain combinations of ACR, SCR and σ_ε^2 the optimal VSI is same as the optimal FSI. For example, in Table 5.3.3-(a) we see that for 5 pairs of ACR and SCR, the corresponding multiple m is 1. Next, if the values in the percentage reduction columns

Table 5.3.3-(a)

Comparison of Optimal VSI vs. FSI
$\sigma_\varepsilon^2 = 1.0,\ t' = 30$

Cost Ratio		Opt. VSI			Opt. FSI		Exp. Loss		% cost reduction of
ACR c_2/c_1	SCR c_3/c_1	d_1	d_2	m	d'	AL	Opt. VSI	Opt. FSI	VSI over FSI
0.2	1	1.3650	1.3650	1	1.3650	0.3810	330.71	330.71	0.0
1	1	1.5000	1.5000	1	1.5000	0.7913	76.36	76.36	0.0
2	1	0.8825	1.7650	2	1.5800	1.0574	84.57	85.44	1.01
0.2	2	2.0000	2.0000	1	2.0000	0.3157	421.88	421.88	0.0
1	2	2.1430	2.1430	1	2.1430	0.6745	92.78	92.78	0.0
2	2	2.3070	2.3070	1	2.3070	0.9071	100.89	100.89	0.0
0.2	0.2	0.1091	0.6546	6	0.5770	0.5674	199.32	206.75	3.6
1	0.2	0.2400	0.7200	3	0.6250	1.0812	260.16	271.57	4.2
2	0.2	0.2381	0.9524	4	0.6383	1.3832	304.73	320.96	5.1

c_3/c_1 (SCR) denotes the ratio of the sampling cost to that of the off-target cost

c_2/c_1 (ACR) denotes the ratio of the adjustment cost to that of the off-target cost

Table 5.3.3-(b)

Comparison of Optimal VSI vs. FSI

$\sigma_\varepsilon^2 = 5.0$, $t' = 30$

Cost Ratio		Opt. VSI			Opt. FSI		Exp. Loss		% cost reduction of VSI over FSI
ACR c_2/c_1	SCR c_3/c_1	d_1	d_2	m	d′	AL	Opt. VSI	Opt. FSI	
0.2	1	0.1500	1.0500	7	0.9678	0.4495	478.47	516.62	7.4
1	1	0.1819	1.8190	10	1.0350	0.9161	105.45	115.09	8.4
2	1	0.1797	1.6173	9	1.0720	1.2019	113.68	124.79	8.9
0.2	2	0.2000	1.6000	8	1.4286	0.3724	619.89	640.27	3.2
1	2	0.3410	2.3870	7	1.5800	0.7707	132.70	137.89	3.8
2	2	0.3750	2.6250	7	1.6667	1.0284	140.33	146.80	4.4

c_3/c_1 (SCR) denotes the ratio of the sampling cost to that of the off-target cost

c_2/c_1 (ACR) denotes the ratio of the adjustment cost to that of the off-target cost are compared for corresponding cost ratios SCR and ACR in Tables 5.3.3-(a) and (b), then it is clear that the performance of the optimal VSI scheme over that of the optimal FSI scheme for the smaller observation error shows less improvement than for the larger observation error. Contrary to one's intuition this seems to suggest that as the observation error increases, the percentage reduction in cost for a given cost combination increases.

5.4 Effect of the input parameters

For the remainder of this section, we will be referring to the Tables 5.3.3-(a) and (b). For any cost combination, the adjustment limits computed by our program for the FSI scheme show clear evidence of convergence, in agreement with Crowder's (1986) results (for example, see Table 5.2.2). This convergent value is referred to as the AL under optimal FSI in Tables 5.3.3-(a) and (b). On the other hand for certain cost combinations it is not easy to see the convergence of the adjustment and sampling limits for the VSI scheme. As a result the adjustment and sampling limits for the optimal VSI scheme are not listed in the tables.

Effect of the observation error, σ_ε^2

For each pair of SCR and ACR, on comparing the AL columns for the optimal FSI scheme in these tables it is clear that the optimal AL is larger for the larger observation error. This means that the larger the observation error, the more conservative one would be in making adjustments since one has less reliable information in this case. Further, for each pair of SCR and ACR, the optimal fixed sampling interval for a larger observation error is smaller compared to that of the optimal fixed sampling interval for a smaller observation error. This means that when one has a large observation error, one would sample more often since the observation is less reliable.

Effect of ACR and SCR

Now the effect of SCR and ACR on the sampling intervals for the FSI and VSI schemes and AL of the FSI scheme will be discussed. For a given observation error the optimal fixed sampling interval gets larger as the SCR increases. Similarly, it can be seen from Table 5.3.3-(b) that in the case of the optimal VSI scheme the shorter and longer sampling intervals get larger as the SCR increases. Moreover it can also be seen that the optimal variable sampling intervals d_1 and d_2 sandwich the optimal fixed sampling interval of d' units. Next, for a given observation error the AL of the optimal FSI scheme gets larger as the ACR increases. The larger the cost of making an adjustment, the less often an adjustment is made.

The above conclusions and comparisons of the performance of the optimal VSI scheme to that of the optimal FSI scheme apply only for the combinations of SCR, ACR and observation error given in the tables. Due to enormous time consuming calculations the investigation of the performance of the optimal VSI scheme relative to that of the optimal FSI scheme in this dissertation has been restricted to only certain combinations of the SCR, ACR and the observation error.

Effect of the total production time, t'

Let us consider two examples given in Tables 5.4.1 and 5.4.2. For the different values of t', the expected loss per unit time, the convergent SL and AL of the VSI scheme and the convergent AL of the FSI scheme are given in the tables. We see that the expected loss per unit time is almost the same for the different values of t' for both the VSI and the FSI schemes. The ALs of the FSI scheme show convergence very quickly, for t' as small as 10 (Tables 5.4.1 and 5.4.2) and n, the number of stages as small as 10 (Table 5.4.2). The SL and AL of the VSI scheme for $t' = 10$, 20 and 30 are not listed in the tables since the SL and AL didn't show clear evidence of convergence. In order to be

Table 5.4.1

Effect of t' on the Expected Loss per unit time

$c_1 = 1, c_2 = 1, c_3 = 1; \sigma_\varepsilon^2 = 1.0, d_1 = 1.0, d_2 = 2.0$

t'	VSI	FSI	VSI		$FSI = d_1$
	EL/unit time	EL/unit time	SL	AL	AL
10	2.55	2.66	-	-	0.9283
20	2.56	2.66	-	-	0.9283
30	2.57	2.66	-	-	0.9283
40	2.57	2.67	0.4208	0.8554	0.9283
50	2.57	2.67	0.4208	0.8554	0.9283
60	2.57	2.67	0.4208	0.8554	0.9283
100	2.57	2.67	0.4208	0.8554	0.9283

Table 5.4.2

Effect of t' on the Expected Loss per unit time

$c_1 = 5, c_2 = 5, c_3 = 1; \sigma_\varepsilon^2 = 1.0, d_1 = 0.25, d_2 = 2.0$

t'	VSI	FSI	VSI		$FSI = d_1$
	EL/unit time	EL/unit time	SL	AL	AL
10	8.77	10.29	-	-	1.2578
20	8.73	10.29	-	-	1.2578
30	8.71	10.29	-	-	1.2578
40	8.69	10.29	0.6341	1.1161	1.2578
50	8.69	10.29	0.6341	1.1161	1.2578
60	8.69	10.29	0.6341	1.1161	1.2578

able to see convergence of the SL and AL for the VSI scheme, we see from Tables 5.4.1 and 5.4.2 that t' needs to be at least 40. This seems to indicate that the results for $t' = 40$ is essentially the same as those for $t' = 60$ or $t' = 100$. This implies that for a problem that has a large production run, for example $t' = 100$, one needs to obtain the AL, SL and the expected loss per unit time for a problem that has $t' = 40$ and then use these results for the $t' = 100$ thereby cutting down the computation time.

5.5 Simulation results for the expected loss

Some simulations were performed to provide a check on the results obtained using the computer program in Appendix B. Using the computer program in Appendix B, the adjustment limits, the sampling limits and the total expected loss were obtained for a given set of cost combinations, observation error, the total production run time and the sampling interval lengths. Then, using these adjustment and the sampling limits as inputs to the simulation, the process was simulated and the total expected loss was obtained which was compared with that obtained using the computer program in Appendix B. A description of the simulation technique is given below.

To begin the simulation, the IML program with RANNOR was used to generate a value of a standard random variable which was then transformed to obtain a starting value for $\hat{\mu}_0$ from a normal distribution with mean zero and variance $\dfrac{(v_0+\sigma_v^2)^2}{v_0+\sigma_v^2+\sigma_\varepsilon^2}$ (refer to Chapter 4) where the value of v_0 was obtained using (3.3.1). The decisions with respect to sampling and adjustment at stage 0 were made depending on the value of the starting value for $\hat{\mu}_0$ and the sampling and adjustment limits at stage 0. Then based on these decisions, the process was simulated to obtain the process value for the next stage. At each stage a standard normal value was generated and transformed to provide a value for the system error from a normal distribution with mean zero and variance $d_1\sigma_v^2$ or $d_2\sigma_v^2$ depending on the sampling decision made at that stage. Another standard normal value was generated and transformed to obtain an observational value from a normal distribution of mean zero and variance σ_ε^2. Then the Kalman filter estimate was then calculated using (3.3.3). Next depending on the value of the Kalman filter estimate for the current stage, the appropriate decisions were taken with respect to sampling and adjustment. Then based on these decisions the process value and the observational value for the next stage were generated. Continuing this way, the process values and the observation values were generated at each stage. The integrated off-target cost was calculated for each stage. Then the total integrated expected loss, the total adjustment cost (depending on how many times an adjustment was made during the production run) and the sampling cost (depending on how many samples were taken) were calculated. The total expected loss was then calculated as the sum of the total integrated off-target cost, the adjustment cost and the sampling cost. This procedure was repeated 100 times and the average of the expected losses were calculated. It was seen that the total expected loss results obtained using simulation agreed with the total expected loss results obtained using the computer program in Appendix B. The following table gives few examples to illustrate that the simulation results agree with the results obatined using the program in Appendix B. In the table, SE stands for standard error, EL(VSI) and EL(FSI) represent the total expected loss for the VSI scheme with d_1 and d_2 as the sampling interval lengths and the FSI scheme with sampling interval length of d_1 respectively.

Table 5.5.1

ACR c_2/c_1	SCR c_3/c_1	σ_ε^2	d_1	d_2	n	Simulation				Program (Appendix B)	
						EL(VSI)	SE	EL(FSI)	SE	EL(VSI)	EL(FSI)
300	1	5.0	1.0	2.0	10	83.64	5.55	86.62	5.49	83.71	87.76
1	1	5.0	0.3	0.9	30	35.12	0.23	47.76	0.16	34.62	48.15
1	10	5.0	0.3	0.9	30	126.17	0.25	317.70	0.18	124.64	318.15
3	0.4	0.5	0.4	1.6	30	139.82	2.12	165.70	2.33	137.76	164.49

Chapter 6: Summary and Further Research

6.1 Description of the Research Problem

In this dissertation, we have considered the problem of studying the optimal adjustment and sampling strategies for a process control model using a VSI scheme. The process control model consists of an observation equation and a process mean equation. We have assumed that the process mean model is a random walk with a control variable. We have assumed that samples from the process are taken using two possible time intervals, namely d_1 and d_2 units, depending on the observed data. The performance criterion used to obtain the adjustment and sampling strategies is an expected loss function. This is a function of three costs namely, the integrated off-target cost, the adjustment cost, and the sampling cost. The adjustment cost is independent of the size of the adjustment made to the process. Also, the sampling cost is the same every time a sample is taken. The goal is to obtain the adjustment and sampling strategies for the above mentioned VSI process control model such that the total expected loss is minimized.

At each stage in an n stage problem, the adjustment and sampling strategies are obtained using the technique of dynamic programming. At each stage, the adjustment limits determine whether an adjustment should be made to the process or not. The sampling limits determine whether to use the shorter interval d_1 the next time, or to use the longer interval d_2 the next time. The question of how much adjustment is required, if any, is determined by the Kalman filter estimate. We have derived the Kalman filter estimator for the VSI process control problem at each stage of the process. Further, we have also derived the conditional distributions of the process mean given past information up to and including stage k and the Kalman filter estimator given past information. These conditional distributions are used in the derivation of the functional equations which are used in determining the adjustment and sampling strategies.

Given the off-target cost, adjustment cost, sampling cost and the observation error, we have shown how one can determine the adjustment limits, the sampling limits and the expected loss for an n stage problem. We have compared the performance of the FSI scheme with that of the VSI scheme. The comparison made is based on the percentage reduction in cost when the VSI scheme is used instead of the FSI scheme. For a given set of cost combinations and observation error, we have determined the FSI scheme that gives the *least* expected loss among a set of FSI schemes. The FSI scheme that gives the *least* expected loss is referred to as the optimal FSI scheme. Then, among a reasonable set of (d_1, d_2), we have determined the combination of (d_1, d_2) that gives the *least* expected loss. This (d_1, d_2) that gives the *least* expected loss is referred to as the optimal VSI scheme. We have then compared the performance of the optimal FSI and VSI schemes. The effects of the sampling cost ratio (ratio of the sampling cost to the off-target cost), the adjustment cost ratio (ratio of the adjustment cost to the off-target cost), and the observation error on the percentage reduction in cost, optimal sampling intervals and optimal adjustment limits are studied.

6.2 Summary of Results

On comparing the optimal VSI scheme with that of the optimal FSI scheme, it is seen that for certain combinations of the cost parameters and observation error the VSI performs much better than the FSI scheme. We have determined the optimal VSI and FSI schemes for observation errors $\sigma_\varepsilon^2=1.0$ and 5.0. For $\sigma_\varepsilon^2=1.0$ based on the percentage reduction in the cost, it was seen that the performance of the VSI over that of the FSI ranges from no improvement to moderate improvement.

For each σ_ε^2=1.0 and 5.0, we have seen that the performance of the optimal VSI scheme is better than the optimal FSI scheme when the sampling cost ratio is at the lower level. This means that when the sampling cost is less than the off-target cost or comparable to the off-target cost, one would profit more by toggling between the shorter and longer sampling interval based on the observed data rather than using fixed sampling intervals. Further we have also seen that for the same SCR (sampling cost ratio) and ACR (adjustment cost ratio), the percentage reduction in cost is higher for the larger observation error.

We have studied the effects of the two cost ratios SCR and ACR, and the observation error on the optimal VSI and FSI sampling intervals and the adjustment limits. As the SCR increases, the optimal sampling intervals of the FSI and VSI schemes increases. This is because when SCR increases it means that the sampling cost relative to the off-target cost increases. As it gets more expensive to take samples, one would want to take samples less often. Next, as the ACR increases, the adjustment limits of the optimal FSI scheme increases. This is because when the ACR increases it means that the adjustment cost relative to the off-target cost increases. As it gets more expensive to make an adjustment, one tends to be more conservative in making adjustments. The adjustment limits define the adjustment and the nonadjustment bands. Larger adjustment limits give rise to wider nonadjustment bands. Therefore, the larger the adjustment cost relative to the off-target cost, the larger are the adjustment limits.

For the optimal FSI scheme, we have compared the adjustment limits for the two observation errors σ_ε^2=1.0 and 5.0. We have seen that the adjustment limits are larger for σ_ε^2=5.0 than for σ_ε^2=1.0. This means, as the information available gets more unreliable, more conservative adjustments are made to the process. We have also compared the sampling interval lengths for the two observation errors σ_ε^2=1.0 and 5.0. We have seen that the sampling intervals lengths are larger for the smaller observation error namely, $\sigma_\varepsilon^2=1.0$ than for the larger observation error namely, $\sigma_\varepsilon^2=5.0$. This means that one would sample more often when there is unreliable information.

For certain sets of the input parameters, the limiting or convergent value for the adjustment and the sampling limits can be obtained numerically. In such cases, these limiting values can be used for an infinite horizon problem, that is, for an n stage problem as $n \to \infty$.

6.3 Future Research

Further Work on the Current Research Problem

In this research, each run of the computer program consists of time consuming computations for the adjustment limits, the sampling limits and the expected loss. Due to time consuming runs, the comparisons of the VSI and FSI schemes were limited to certain combinations of the cost parameters and the observation errors. It may be useful to investigate the effects of various other cost parameters and observation errors on the percentage reduction in cost. On doing so, one may be able to make more general statements regarding the effects of the cost parameters and the observation errors on the percentage reduction in the cost. Moreover, this may help understand why the percentage reduction in cost is higher for the larger observation error in the examples seen so far. Also, one can investigate the behavior of the expected loss if the period of unequal length (in the case when t/d' is not an integer) is not necessarily taken to be the last period.

In Tables 5.3.3-(a) and (b), the AL of the optimal VSI scheme for the various cost ratios were not given since the convergence of the SL and AL were not clearly seen for certain combinations of the ACR, SCR and the observation errors. The reason being that the number of stages n wasn't large enough. It would be interesting to see if an analytical solution for the adjustment and sampling limits can be found to the research problem, may be based on a different approach. The analytical solution to the SL and AL for the optimal VSI scheme would enable one to use these convergent values for a problem that has a large number of stages. Moreover, one could study the effects of ACR, SCR and the observation errors on the SL and AL without having to do a lot of time consuming computations.

The following are some other areas in which this research can be extended.

Use of VSI schemes with more than two sampling intervals

We have developed the adjustment and the sampling strategies for the VSI process control problem that used the VSI scheme with two sampling intervals. A possible extension to this problem could consider determining the adjustment and sampling strategies for the process control problem when more than two sampling intervals are used in the VSI scheme. It may be interesting to see how the performance of the VSI schemes with more than two sampling intervals compare with the VSI scheme with two sampling intervals. In the context of SPC control charts, it was shown by Reynolds and Arnold (1989) that the VSI scheme with two sampling intervals is optimal when using a Shewhart chart to detect changes in the process mean parameter. It may be interesting to investigate if the VSI scheme with two sampling intervals is optimal based on the integrated expected loss for a process control problem that uses a feedback regulation system.

Other loss functions

In this dissertation, we have assumed that the adjustment cost is the same irrespective of the size of adjustment made to the process. We have also assumed that whenever a sample is taken, the sample size is the same. A possible extension of our problem could be to develop adjustment and sampling strategies to the VSI process control problem when the expected loss function is a function of the adjustment cost that is proportional to the size of adjustment made to the process, and the sampling cost is proportional to the sample size.

<u>Implementing Process Dynamics in the problem</u>

In our VSI process control problem, we did not consider any process dynamics. We have assumed that any adjustments made to the process has an immediate effect on the process. In some cases an adjustment made at time *k* may not have any effect until time $k+a$. A possible extension to our problem could be to develop adjustment and sampling strategies to the VSI process control problem in situations when the delay in the effect of an adjustment is a periods.

<u>Alternative process mean models</u>

Throughout this research it was assumed that the process mean followed a random walk model. A more interesting model for the process mean could be the AR(1) autoregressive model, namely $\mu_k = \varphi\mu_{k-1}+a_{k-1}+v_k$ with the autoregressive parameter φ. Further research could consider determining the optimal control strategies when the process mean assumes an AR(1) model. The effect of the autoregressive parameter on the control strategies may be of interest. The random walk model considered in this dissertation is a special case of the AR(1) model. Also, further research could be done in the lines of determining optimal control strategies when the process mean model assumes different time series models such as the moving average MA, autoregressive integrated moving average ARIMA etc. The effects of the different model parameters on the process control strategies could be investigated.

<u>Multiple input-output system</u>

We have considered a process control model which is a single input-output system. In some process control problems, there could be situations where a set of control variables is known to affect a set of output variables. A possible extension to this dissertation could consider determining the process control strategies to a multiple input-output system where the various control variables are correlated.

Bibliography

Abraham, B. and Box, G.E.P. (1979), "Sampling Interval and Feedback Control", *Technometrics*, Vol.21, No.1, 1-8.

Abraham, B. and Kartha, C.P. (1979), "Forecast Stability and Control Charts", *1979 ASQC Technical Conference Transactions*, Houston, Texas, 675-680.

Adams, B. M. and Woodall, W.H. (1989). "An Analysis of Taguchi's On-Line Process Control Procedure under a Random Walk Model", *Technometrics*, Vol. 31, 4, 401- 413.

Alwan, L.C. and Roberts, H.V. (1988), "Time series Modeling for Statistical Process Control", *Journal of Business and Economic Statistics*, 6, 87-95.

Amin, R.W. (1987), "Variable Sampling Interval Control Charts", Ph.D. Thesis, Virginia Polytechnic Institute and State University, Dept. of Statistics.

Amin, R.W. and Hemasinha, R. (1993), "The Swicthing Behavior of $\overline{X}$ Charts With Variable Sampling Intervals", *Communications in Statistics-Theory and Methods*, 22, 2081-2102.

Amin, R.W. and Letsinger, W.C. II (1991), "Improved Swicthing Rules in Control Procedures Using Variable Sampling Intervals", *Communications in Statistics-Simulation and Computations*, 20, 205-230.

Amin, R.W. and Miller, R.W. (1993), "A Robustness Study of $\overline{X}$ Charts With Variable Sampling Intervals", *Journal of Quality Technology*, 25, 26-44.

Amin, R.W. and Ncube, M.M. (1991), "Variable Sampling Interval Combined Shewhart-Cumulative Score Quality Control Procedure", *Applied Statistics*, 40, 1-12.

Anderson, T.W. (1971), An Introduction to Multivariate Statistical Analysis. Wiley, New York.

Arnold, J.C. (1970), "A Markovian Sampling Policy Applied to Quality Monitoring of Streams", *Biometrics*, 26, 739-747.

Arnold, J.C. and Crigler, J.R. (1971), "On Economically Optimal Markovian Sampling Policies With Application to Water Pollution Monitoring", *Proceedings of the International Statistical Institute* (42nd Session), 501-525.

Astrom, K.J. (1970), Introduction to Stochastic Control Theory. Academic, NY.

Bagshaw, M. and Johnson, R. A. (1975), "The Effect of Serial Correlation on the performance of CUSUM Tests II", *Technometrics*, Vol. 17, 73-80.

Astrom, K.J. and Wittenmark, B. (1984), Computer Controlled Systems. Prentice-Hall, NY.

Bagshaw, M. and Johnson, R.A. (1975), "The Effect of Serial Correlation on the performance of CUSUM Tests II", *Technometrics*, Vol. 17, 73-80.

Bellman, R.E. (1957), Dynamic Programming, Princeton University Press, Princeton, NJ.

Bellman, R.E. (1967), Introduction to the Mathematical Theory of Control Processes, Academic Press, New York, NY.

Bellman, R.E. and Dreyfus, S.E. (1962), Applied Dynamic Programming, Princeton University Press, Princeton, NJ.

Bellman, R.E. and Kalaba, R.E. (1965), Dynamic Programming and Modern Control Theory, Academic Press, New York, NY.

Bishop, A.B. (1957), "A Model for Optimum Control of Stochastic Sampled data Systems", *Operations Research*, 5, No.4, 546-550.

Bishop, A.B. (1960), "Discrete Random Feedback Models in Industrial Quality Control", *The Ohio State University Engineering Experimental Station Bulletin*.

Box, G.E.P. and Jenkins, G.M. (1962), "Some Statistical Aspects of Adaptive Optimization and Control", *Journal of the Royal Statistical Society*, B 24, 297-343.

Box, G.E.P. and Jenkins, G.M. (1963), "Further Contributions to Adaptive Quality Control: Simultaneous Estimation of Dynamics: Nonzero Costs", *Bulletin of the International Statistical Institute*, 34, 943-974.

Box, G.E.P. and Jenkins, G.M. (1970). Time Series Analysis: Forecasting and Control. Holden Day, San Francisco.

Box, G.E.P. and Kramer, T. (1992), "Statistical Process Monitoring and Feedback Adjustment-A Discussion", *Technometrics*, Vol. 34, 3, 251-285.

Box, G.E.P. and Luceño, A. (1994), "Selection of Sampling Interval and Action Limit for Discrete Feedback Adjustment", *Technometrics*, Vol. 36, 4, 369-378.

Box, G.E.P. and MacGregor, J.F. (1974), "An Analysis of Closed-Loop Dynamic-Stochastic Systems", *Technometrics*, Vol. 16, 3, 391-398.

Chengular-Smith, I.N., Arnold, J.C., and Reynolds, M.R., Jr. (1989), "Variable Sampling Intervals for Multiparameter Shewhart Charts", *Communications in Statistics - Theory and Methods*, 18, 1769-1792.

Cooper, L. and Cooper, M.W. (1981). Introduction to Dynamic Programming. Pergamon Press, Oxford.

Crowder, S. (1986), "Kalman Filtering and Statistical Process Control", Ph.D., Thesis, Iowa State University, Dept. of Statistics. (Published as "An SPC model for Short Production Runs: Minimizing Expected Cost", *Technometrics*, (1992) Vol. 34, 1, 64-73.)

Cui, R. and Reynolds, M.R., Jr. (1988), "$\overline{X}$ Charts With Runs Rules and Variable Sampling Intervals", *Communications in Statistics-Theory and Methods*, 17, 1073- 1093.

Duncan, D.B. and Horn, S.D. (1972), "Linear dynamic recursive estimation from the viewpoint of regression analysis", *Journal of the American Statistical Association*, 67, 815-821.

Ermer, D.S. (1980), "A Control Chart for Dependent Data", *1980 ASQC Technical Conference Transactions*, Atlanta, Georgia, 121-128.

Girshick, M.A. and Rubin, H. (1952), "A Bayes Approach to a Quality Control Model", *Annals of Mathematical Statistics*, 23, 114-125.

Goldsmith, P.L. and Whitfield, H. (1961), "Average Run Lengths In Cumulative Chart Quality Control Schemes", *Technometrics*, Vol. 3, 11-20.

Harrison, P.J. and Stevens, C.F. (1971), "A Bayesian Approach to Short-term Forecasting", *Operational Research Quarterly*, 22, 341-362.

Harrison, P.J. and Stevens, C.F. (1976), "Bayesian Forecasting", *Journal of the Royal Statistical Society*, B, 38, 205-228.

Hui, Y.V. and Jensen, D.R. (1980), "Markovian time-delay sampling policies", *Technical Report No.* Q-5, Department of Statistics, VPI&SU, Blacksburg, Va.

Jensen, K.L. (1989), "Optimal Adjustment in the Presence of Deterministic Process Drift and Random Adjustment Error", Ph.D., Thesis, Iowa State University, Dept. of Statistics. (Published in *Technometrics*, (1993) Vol. 35, 4, 376-389.)

Johnson, R.A. and Bagshaw, M. (1974), "The Effect of Serial Correlation on the performance of CUSUM Tests", *Technometrics*, Vol. 16, 103-112.

Kalman, R.E. (1960), "A New Approach to Linear Filtering and Prediction Theory", *Journal of Basic Engineering*, 82, 34-45.

Kalman, R.E. and Bucy, R.S. (1961), "New Results in Linear Filtering and Prediction Theory", *Journal of Basic Engineering*, 83, 95-108.

Kartha, C.P. and Abraham, B. (1979), "ARL for CUSUM Charts with Correlated Observations", *1979 ASQC Technical Conference Transactions*, Houston, Texas, 206-210.

Kelly, S.J., MacGregor, J.F., and Hoffman, T.W. (1987), "Control of a Continuous Polybutadiene Polymerization Reactor Train", *The Canadian Journal of Chemical Engineering*, Vol.65, 852-857.

Ledolter, J. (1979), "A recursive Approach to Parameter Estimation in Regression and Time Series Models", *Communications in Statistics*, A, 1227-1246.

Lu, C.W. (1994), "Control Charts Based on Residuals for Monitoring Autocorrelated processes", *Technical Report* No: 94-8, Department of Statistics, VPI&SU, Blacksburg, VA.

MacGregor, J.F. (1973), "Optimal Discrete Stochastic Control Theory for Process Application", *Canadian Journal of Chemical Engineering*, 51, 468-478.

MacGregor, J.F. (1991), "A Different View of the Funnel Experiment", *Journal of Quality Technology*, 22, 255-259.

MacGregor, J.F. and Wong, A.K.L. (1980), "Multivariate Model Identification and Stochastic Control of a Chemical Reactor", *Technometrics*, 22, 453-464.

Mathews, J.H. (1992), Numerical Methods for Mathematics, Science and Engineering.

Montgomery, D.C. (1985), Introduction to Statistical Quality Control, John Wiley and Sons, New York.

Montgomery, D.C. and Mastrangelo, C.M. (1991), "Some Statistical Process Control Methods for Autocorrelated Data", *Journal of Quality Technology*, 23, 179-193.

Phadke, M.S. (1981), "Quality Evaluation Plan Using Adaptive Kalman Filtering", *Bell System Technical Journal*, 61, 2081-2107.

Prabhu, S.S., Montgomery, D.C., and Runger, G.C. (1994), "A Combined Adaptive Sample Size and Sampling Interval $\overline{X}$ Control Scheme", *Journal of Quality Technology*, 26, 164-176.

Rendtel, U. (1987), (1990), "CUSUM-Schemes With Variable Sampling Intervals and Sample Sizes", *Statistical Papers*, 31, 103-118.

Reynolds, M.R., Jr. (1988a), "Optimal Markov Chain and Two-sided Shewhart Control Charts with Variable Sampling Intervals", *Technical Report* No. 88-2, Department of Statistics, VPI&SU, Blacksburg, Va.

Reynolds, M.R., Jr. (1988b), "Markovian Variable Sampling Interval Control Charts", *Technical Report* No. 88-22, Department of Statistics, VPI&SU, Blacksburg, Va.

Reynolds, M.R., Jr. (1989), "Optimal Variable Sampling Interval Control Charts", *Sequential Analysis*, 8, 361-379.

Reynolds, M.R., Jr. (1995), "Evaluating Properties of Variable Sampling Interval Control Charts", *Sequential Analysis*, 14(1), 59-97.

Reynolds, M.R., Jr., Amin, R.W., and Arnold, J.C. (1990), "CUSUM Charts With Variable Sampling Intervals", *Technometrics*, Vol. 32, 371-384.

Reynolds, M.R., Jr., Amin, R.W., Arnold, J.C., and Nachlas, J.A. (1988), "$\overline{X}$ Charts With Variable Sampling Intervals", *Technometrics*, Vol. 30, 181-192.

Reynolds, M.R., Jr., and Arnold, J.C. (1989), "Optimal One-Sided Shewhart Control Charts With Variable Sampling Intervals", *Sequential Analysis*, 8, 51-77.

Reynolds, M.R., Jr., Arnold, J.C., and Baik, J.W., "Variable Sampling Interval $\overline{X}$-Charts in the Presence of Correlation", to be published in the *Journal of Quality Technology*.

Runger, G.C. and Montgomery, D.C. (1993), "Adaptive Sampling Enhancements for Shewhart Control Charts", *IIE Transactions*, 25, 41-51.

Runger, G.C. and Pignatiello, J.J., Jr. (1991), "Adaptive Sampling for Process Control", *Journal of Quality Technology*, 23, 135-155.

Saccucci, M.S., Amin, R.W., and Lucas, J.M. (1992), "Exponentially Weighted Moving Average Control Schemes With Variable sampling Intervals", *Communications in Statistics-Simulation and Computations*, 21, 627-657.

Sallas, W.M. and Harville, D.A. (1981), "Best Linear Recursive Estimation for Mixed Linear Models", *Journal of the American Statistical Association*, 76, 860-869.

Sant, D.T. (1977), "Generalized Least Squares Applied to Time-varying Parameter Models", *Annals of Economic and Social Measurement*, 6, 301-311.

Sarris, A.H. (1973), "A Bayesian Approach to Estimation of Time-varying Regression Coefficients", *Annals of Economic and Social Measurement*, 2, 501-523.

Shamma, S.E., Amin, R.W., and Shamma, A.K. (1991), "A Double Exponentially Weighted Moving Average Control Procedure With Variable Sampling Intervals", *Communications in Statistics-Theory and Methods*, 20, 511-528.

Sorenson, H.W. (1966), "Kalman Filtering Techniques", *Advances in Control Systems Theory and Applications*, Vol.3, 219-292.

Taguchi, G., Elsayed, E.A., and Hsiang, T. (1989). Quality Engineering in Production Systems, New York.

Taguchi, G. (1985), "Quality Engineering in Japan", *Communications in Statistics-Theory and Methods*, 14, 2785-2801.

VanBrackle, L.N. (1991), "EWMA and CUSUM Control Charts in the Presence of Correlation", Ph.D., Thesis, Virginia Polytechnic Institute and State University, Dept. of Statistics.

Vander Wiel, S.A., Tucker, W.T., Faltin, F.W., and Doganaksoy, N. (1992), "Algorithmic Statistical Process Control: Concepts and an Application", *Technometrics*, Vol. 34, 286-297.

Vaughan, T.S. (1993), "Variable Sampling Interval np Process Control Chart", *Communications in Statistics-Theory and Methods*, 22, 147-167.

Vining, G.G. and Reynolds, M.R. Jr. (1989), "A Multilevel Sampling Interval Approach to Control Charts", unpublished manuscript.

APPENDIX A

The following results (Anderson (1971)) are used in the derivation of the Kalman filter estimator given in Theorem 1 below.

Result 1:

Let Y_1 and Y_2 be two random variables whose bivariate normal distribution is given by

$$\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} \sim N\left[\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right]. \qquad \text{(A1)}$$

Then, the conditional distribution of Y_1 given Y_2 is given by

$$Y_1 | Y_2 = y_2 \sim N[\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}]. \qquad \text{(A2)}$$

Result 2:

If (A2) holds and $Y_2 \sim N[\mu_2, \Sigma_{22}]$, then (A1) holds.

Theorem 1:

The general expression of the Kalman filter estimator at stage k for the process mean of the VSI process control problem is given by the mean of the following posterior distribution

$$\mu_k | x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}, x_k \sim N[\hat{\mu}_k, v_k], \qquad \text{(A3)}$$

where

$$\hat{\mu}_k = (\hat{\mu}_{s(k)} + a_{s(k)}) + k_{d(k),k}(x_k - (\hat{\mu}_{s(k)} + a_{s(k)})) \qquad \text{(A4)}$$

and

$$k_{d(k),k} = \frac{v_{s(k)} + d(k)\sigma_v^2}{v_{s(k)} + d(k)\sigma_v^2 + \sigma_\varepsilon^2},$$

$$v_k = k_{d(k),k}\, \sigma_\varepsilon^2, \qquad \text{(A5)}$$

where s(k) denotes the last stage before stage k at which a sample was taken with s(k) = 0 if no previous sample is taken and d(k) denotes the sampling interval used before stage k.

Also, the conditional distribution of the Kalman filter $\hat{\mu}_k$ given information available up to stage s(k), which is used in deriving the optimal adjustment and sampling strategies (Section 3.5), is given by

$$\hat{\mu}_k | x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, .., x_{s(k)} \sim N[M_k, u_k],$$

where $M_k = \hat{\mu}_{s(k)} + a_{s(k)}$ and the conditional variance u_k is given by

$$\begin{aligned} u_k &= \frac{(v_{s(k)} + d(k)\sigma_v^2)^2}{v_{s(k)} + d(k)\sigma_v^2 + \sigma_\varepsilon^2} \\ &= k_{d(k),k}(v_{s(k)} + d(k)\sigma_v^2). \end{aligned} \tag{A6}$$

<u>Proof:</u>

We will first prove (A3) for the case when stage k is the first stage at which a sample is taken which implies that s(k)=0. Then, assuming that (A3) holds for any stage j < k, we will prove that (A3) holds for any stage k.

Let time t_{j+1} be the time associated with stage k. Then time t_j is the time associated with stage s(k). Let s(k) = 0. Let v_k denote the system error at stage k. At stage k, we see from (3.2.1) and (3.2.2) that the sequential model definitions expressed in terms of stages, are given by

$$x_k = \mu_k + \varepsilon_k$$

and

$$\mu_k = \mu_0 + a_0 + v_k,$$

where v_k, the system error at stage k incorporates the last sampling interval d(k). Using the knowledge available about μ_0 which is given by

$$\mu_0 | x_0 \sim N[\hat{\mu}_0, v_0]$$

our state of knowledge about μ_k prior to observing x_k is described by the conditional distribution

$$\mu_k \,|x_0 \sim N[M_k, w_k\,] \tag{A7}$$

where

$$M_k = \hat{\mu}_0 + a_0,$$

and

$$w_k = v_0 + d(k)\sigma_v^2.$$

In the above expression for w_k, we note that since the distance between stage 0 and stage k is interval d(k), the variance of $v_k|x_0$ is $d(k)\sigma_v^2$. The very first adjustment, a_0, made to the process denoted at stage 0 is a function of the past history of the process. Let $\hat{x}_k$ denote the predicted x_k when μ_k is unknown. Let e_k denote the error in predicting x_k, given data x_0. Then,

$$\begin{aligned}\hat{x}_k &= E(\mu_k|x_0)\\ &= \hat{\mu}_0 + a_0\end{aligned}$$

and

$$\begin{aligned}e_k &= x_k - \hat{x}_k\\ &= x_k - (\hat{\mu}_0 + a_0)\\ &= \mu_k + \varepsilon_k - (\hat{\mu}_0 + a_0).\end{aligned}$$

Now,

$$e_k|\mu_k, x_0 \sim N[\mu_k - (\hat{\mu}_0 + a_0), \sigma_\varepsilon^2]. \tag{A8}$$

On observing x_k, the goal is to compute the posterior distribution of $\mu_k|x_0,x_k$. Results 1 and 2 are used to obtain the posterior distribution of $\mu_k|x_0,x_k$. Let the conditional distribution of $Y_1|Y_2$ in Result 1 correspond to that of $e_k|\mu_k,x_0$. Let the distribution of Y_2 correspond to that of $\mu_k|\ x_0$. Then equations (A7) and (A8) give the distributions of Y_2 and $Y_1|Y_2$ respectively. Hence applying Result 2, we get

$$\begin{pmatrix} e_k \\ \mu_k \end{pmatrix} |\ x_0 \sim N\left[\begin{pmatrix} 0 \\ \hat{\mu}_0 + a_0 \end{pmatrix}, \begin{pmatrix} \sigma_\varepsilon^2 + v_0 + d(k)\sigma_v^2 & v_0 + d(k)\sigma_v^2 \\ v_0 + d(k)\sigma_v^2 & v_0 + d(k)\sigma_v^2 \end{pmatrix}\right]. \tag{A9}$$

Since observing x_k is equivalent to observing e_k, the distribution of $\mu_k|x_0,x_k$ is equivalent to that of $\mu_k|x_0,e_k$. The distribution of $\mu_k|e_k,x_0$ is determined by using Result 1. Now let Y_1 and Y_2 be redefined, i.e., let the joint distribution of $\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}$ correspond to that of $\begin{pmatrix} \mu_k \\ e_k \end{pmatrix} | x_0$. Using (A9) we obtain

$$\mu_k|e_k,x_0 \sim N[\hat{\mu}_k, v_k].$$

Since $\mu_k|x_0,x_k$ is equivalent to $\mu_k|x_0,e_k$, we can write

$$\mu_k|x_0,x_k \sim N[\hat{\mu}_k, v_k],$$

where

$$\hat{\mu}_k = (\hat{\mu}_0 + a_0) + k_{d(k),k}(x_k - (\hat{\mu}_0 + a_0)) \tag{A10}$$

and

$$k_{d(k),k} = \frac{v_0 + d(k)\sigma_v^2}{v_0 + d(k)\sigma_v^2 + \sigma_\varepsilon^2},$$
$$v_k = k_{d(k),k}\sigma_\varepsilon^2,$$

thus proving (A3) for the case when s(k)=0. The Kalman filter estimator of μ_k based on x_0 and x_k is the posterior mean of $\mu_k|x_0,x_k$ and thus this estimator is $\hat{\mu}_k$. Crowder (1986) derived the Kalman filter estimator given above for the case d(k) = 1.

When the equation (A10) is rewritten as $\hat{\mu}_k = (1-k_{d(k),k})(\hat{\mu}_0+a_0)+k_{d(k),k}x_k$, it can be easily seen that for $\sigma_\varepsilon^2 >> \sigma_v^2$ (implying large observation error), the weight $k_{d(k),k}$ is small and the current observation x_k receives relatively little weight. On the other hand for $\sigma_\varepsilon^2 << \sigma_v^2$, $k_{d(k),k}$ is large (close to 1) and the new data point x_k receives most of the weight.

Also, the conditional distribution of the Kalman filter $\hat{\mu}_k$ given x_0, which will be later used in deriving the optimal adjustment and sampling strategies, is obtained as follows. From (A10), using $\varepsilon_k \sim N[0,\sigma_\varepsilon^2]$ and (A7), the conditional distribution $\hat{\mu}_k$ given x_0 is given by

$$\hat{\mu}_k | x_0 \sim N[M_k, u_k],$$

where M_k is given at (A7) and the conditional variance u_k is given by

$$u_k = \frac{(v_0 + d(k)\sigma_v^2)^2}{v_0 + d(k)\sigma_v^2 + \sigma_\varepsilon^2}$$
$$= k_{d(k),k}(v_0 + d(k)\sigma_v^2),$$

thus proving the theorem for the case $s(k) = 0$.

We shall now show that (A3) holds for any stage k, given that it holds for any previous stage $j < k$. At stage k, the model equations are given by

$$x_k = \mu_k + \varepsilon_k$$

and

$$\mu_k = \mu_{s(k)} + a_{s(k)} + v_k,$$

where v_k, the system error incorporates the last sampling interval from stage s(k) to stage k, where $s(k) \neq 0$ now. Since we have assumed that (A3) holds for any stage $j < k$, (A3) will therefore hold for $j = s(k)$. Now using (A3) for $j=s(k)$ we find that the prior distribution for μ_k is given by

$$\mu_k | x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)} \sim N[M_k, w_k] \tag{A11}$$

where

$$M_k = \hat{\mu}_{s(k)} + a_{s(k)}$$

and

$$w_k = v_{s(k)} + d(k)\sigma_v^2.$$

In the above expression for w_k we note that since the distance between stage s(k) and stage k is d(k), the variance of $v_k | x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}$ is $d(k)\sigma_v^2$. Let $\hat{x}_k$ denote the predicted x_k when μ_k is unknown. Let e_k denote the error in predicting x_k, given data up to stage s(k). Then,

$$\hat{x}_k = E(\mu_k | x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)})$$
$$= \hat{\mu}_{s(k)} + a_{s(k)}$$

and

$$\begin{aligned} e_k &= x_k - \hat{x}_k \\ &= x_k - (\hat{\mu}_{s(k)} + a_{s(k)}) \\ &= \mu_k + \varepsilon_k - (\hat{\mu}_{s(k)} + a_{s(k)}). \end{aligned}$$

Now,

$$e_k | \mu_k, x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)} \sim N[\mu_k - (\hat{\mu}_{s(k)} + a_{s(k)}), \sigma_\varepsilon^2]. \tag{A12}$$

On observing x_k, using results 1 and 2 we shall now compute the posterior distribution of $\mu_k | x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}, x_k$. Let the conditional distribution of $Y_1|Y_2$ in Result 1 correspond to that of $e_k|\mu_k, x_0, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}$. Let the distribution of Y_2 correspond to that of $\mu_k|x_0, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}$. Then equations (A11) and (A12) give the distributions of Y_2 and $Y_1|Y_2$ respectively. Hence applying Result 2, we get

$$\begin{pmatrix} e_k \\ \mu_k \end{pmatrix} | \, x_0, \ldots, x_{s(k)} \sim N\left[\begin{pmatrix} 0 \\ \hat{\mu}_{s(k)} + a_{s(k)} \end{pmatrix}, \begin{pmatrix} \sigma_\varepsilon^2 + v_{s(k)} + d(k)\sigma_v^2 & v_{s(k)} + d(k)\sigma_v^2 \\ v_{s(k)} + d(k)\sigma_v^2 & v_{s(k)} + d(k)\sigma_v^2 \end{pmatrix}\right]. \tag{A13}$$

Since observing x_k is equivalent to observing e_k, the distribution of $\mu_k|x_0, \gamma(1)x_1, \ldots, x_{s(k)}, x_k$ is equivalent to that of $\mu_k|x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}, e_k$. The distribution of $\mu_k|e_k, x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}$ is determined by using Result 1. Redefining Y_1 and Y_2, i.e., let the joint distribution of $\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}$ correspond to that of $\begin{pmatrix} \mu_k \\ e_k \end{pmatrix} | \, x_0, \gamma(1)x_1, \ldots, x_{s(k)}$. Using (A13) we obtain

$$\mu_k | e_k, x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)} \sim N[\hat{\mu}_k, v_k].$$

Since $\mu_k|x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}, x_k$ is equivalent to $\mu_k|x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}, e_k$, we can write

$$\mu_k | x_0, \gamma(1)x_1, \ldots, \gamma(h)x_h, \ldots, x_{s(k)}, x_k \sim N[\hat{\mu}_k, v_k],$$

where

$$\hat{\mu}_k = (\hat{\mu}_{s(k)} + a_{s(k)}) + k_{d(k),k}(x_k - (\hat{\mu}_{s(k)} + a_{s(k)}))$$

and

$$k_{d(k),k} = \frac{v_{s(k)}+d(k)\sigma_v^2}{v_{s(k)}+d(k)\sigma_v^2+\sigma_\varepsilon^2},$$

$$v_k = k_{d(k),k}\,\sigma_\varepsilon^2,$$

thus proving that (A3) holds for any stage k.

The Kalman filter estimator of μ_k based on information up to stage k is the posterior mean of $\mu_k|x_0,\gamma(1)x_1,\ldots,x_{s(k)},x_k$ and thus this estimator is $\hat{\mu}_k$. When equation (A4) is rewritten as $\hat{\mu}_k=(1-k_{d(k),k})^*(\hat{\mu}_{s(k)}+a_{s(k)})+k_{d(k),k}x_k$, it can be easily seen that for $\sigma_\varepsilon^2>>\sigma_v^2$ (implying large observation error), the weight $k_{d(k),k}$ is small and the current observation x_k receives relatively little weight. On the other hand for $\sigma_\varepsilon^2<<\sigma_v^2$, $k_{d(k),k}$ is large (close to 1) and the new data point x_k receives most of the weight.

Also, the conditional distribution of the Kalman filter $\hat{\mu}_k$ given information available up to stage s(k), which will be later used in deriving the optimal adjustment and sampling strategies, is obtained as follows. From (A11) and (A4), the conditional distribution $\hat{\mu}_k$ given information available up to stage s(k) is given by

$$\hat{\mu}_k|x_0,\gamma(1)x_1,\ldots,\gamma(h)x_h,..,x_{s(k)} \sim N[M_k,u_k],$$

where M_k is given at (A11) and the conditional variance u_k is given by

$$u_k = \frac{(v_{s(k)}+d(k)\sigma_v^2)^2}{v_{s(k)}+d(k)\sigma_v^2+\sigma_\varepsilon^2}$$
$$= k_{d(k),k}(v_{s(k)}+d(k)\sigma_v^2),$$

thus proving the theorem for any stage k.

The following Lemma states that the conditional variance at any stage k for the case when the shorter sampling interval is always used, is less than the conditional variance for the case when the number of samples taken from stage 0 up to stage k is the least.

Lemma 1:

Given v_0 as the steady state variance when a FSI scheme of 1 unit is used and given that for any stage k, the set

$$[v_j]_k = (v_{k,1,d_1}, v_{k,2,d_1}, \ldots, v_{k,j,d_i}, \ldots, v_{k,p,d_{max}(k)})$$

where $d_{max}(k)$ (= $m_{max}(k)d_1$) denotes the biggest possible sampling interval used just before stage k, $m_{max}(k)$ denotes the multiple that relates $d_{max}(k)$ to d_1, i.e., $d_{max}(k) = m_{max}(k)d_1$, $s_{max}(k)$ denotes the previous stage before k at which $d_{max}(k)$ was used, $d_i = m_i d_1$, for i=1,2,...,s with $m_1 = 1$ and $1 < m_2 < m_3 \ldots < m_s$ where m_i are all integers, the following holds

$$v_{k,1,d_1} < v_{k,p,d_{max}(k)}, \text{ for } 1 \le k \le n. \tag{A14}$$

Proof:

When k=1, $[v_1]_1$ contains only one element, namely $v_{k,1,d_1}$ and therefore the proof is trivial. So we shall start by proving that (A14) holds for k = 2 and then using the method of mathematical induction it will be shown that if (A14) is true for k = r-1, then (A14) is true for k = r, for $3 \le r \le n$.

case (i): k = 2

When k = 2, the set $[v_j]_2 = (v_{2,1,d_1}, v_{2,2,d_2})$ contains two elements. These two elements are given by

$$v_{2,1,d_1} = \left(\frac{v_{1,1,d_1} + d_1\sigma_v^2}{v_{1,1,d_1} + d_1\sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2$$

and

$$v_{2,2,d_2} = \left(\frac{v_0 + d_2\sigma_v^2}{v_0 + d_2\sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2.$$

In this case, equivalent to proving (A14) is to prove that

$$v_{2,1,d_1} < v_{2,2,d_2}. \tag{A15}$$

Now,

$$v_{2,1,d_1} - v_{2,2,d_2} = \left(\frac{v_{1,1,d_1} + d_1\sigma_v^2}{v_{1,1,d_1} + d_1\sigma_v^2 + \sigma_\varepsilon^2}\right) * \sigma_\varepsilon^2 - \left(\frac{v_0+d_2\sigma_v^2}{v_0+d_2\sigma_v^2+\sigma_\varepsilon^2}\right)*\sigma_\varepsilon^2$$

$$= \left(\frac{(v_{1,1,d_1} - v_0)+(d_1-d_2)*\sigma_v^2}{(v_{1,1,d_1}+d_1\sigma_v^2+\sigma_\varepsilon^2)*(v_0+d_2\sigma_v^2+\sigma_\varepsilon^2)}\right)*\sigma_\varepsilon^4. \qquad \text{(A16)}$$

First we will show that the following (A17) hold.

$$(v_{1,1,d_1} - v_0) < d_1 * \sigma_v^2 \qquad \text{(A17)}$$

The expression for $v_{1,1,d_1}$ is given by

$$v_{1,1,d_1} = \left(\frac{v_0+d_1\sigma_v^2}{v_0+d_1\sigma_v^2+\sigma_\varepsilon^2}\right)*\sigma_\varepsilon^2.$$

Then $v_{1,1,d_1} - v_0 = \left(\frac{v_0+d_1\sigma_v^2}{v_0+d_1\sigma_v^2+\sigma_\varepsilon^2}\right)*\sigma_\varepsilon^2 - v_0$. Now, since $v_0+d_1\sigma_v^2+\sigma_\varepsilon^2 > \sigma_\varepsilon^2$, we have

$$v_{1,1,d_1} - v_0 < \left(\frac{v_0+d_1\sigma_v^2}{\sigma_\varepsilon^2}\right)*\sigma_\varepsilon^2 - v_0$$

$$= d_1*\sigma_v^2.$$

Thus, $v_{1,1,d_1} - v_0 < d_1*\sigma_v^2$, proving (A17).

Using (A17) in (A16), we see that

$$v_{m,1,d_1} - v_{m,2,d_2} < 0,$$

thus proving (A15).

Now let's assume that (A14) is true for k = r-1. We will now prove (A14) for k = r. Recursive expressions for $v_{r,1,d_1}$ and $v_{r,p,d_{max}(r)}$ are given by

$$v_{r,1,d_1} = \left(\frac{v_{r-1,1,d_1}+d_1\sigma_v^2}{v_{r-1,1,d_1}+d_1\sigma_v^2+\sigma_\varepsilon^2}\right)*\sigma_\varepsilon^2$$

and

$$v_{r,p,d_{max}(r)} = \left(\frac{v_{s_{max}(r),p,d_{max}(s_{max}(r))}+d_{max}(r)\sigma_v^2}{v_{s_{max}(r),p,d_{max}(s_{max}(r))}+d_{max}(r)\sigma_v^2+\sigma_\varepsilon^2}\right)*\sigma_\varepsilon^2.$$

Then,

$v_{r,1,d_1} - v_{r,p,d_{max}(r)}$

$$= \left(\frac{v_{r-1,1,d_1} + d_1\sigma_v^2}{v_{r-1,1,d_1} + d_1\sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2 - \left(\frac{v_{s_{max}(r),p,d_{max}(s_{max}(r))} + d_{max}(r)\sigma_v^2}{v_{s_{max}(r),p,d_{max}(s_{max}(r))} + d_{max}(r)\sigma_v^2 + \sigma_\varepsilon^2} \right) * \sigma_\varepsilon^2$$

$$= \left(\frac{(v_{r-1,1,d_1} - v_{s_{max}(r),p,d_{max}(s_{max}(r))}) + (d_1 - d_{max}(r)) * \sigma_v^2}{(v_{r-1,1,d_1} + d_1\sigma_v^2 + \sigma_\varepsilon^2) * (v_{s_{max}(r),p,d_{max}(s_{max}(r))} + d_{max}(r)\sigma_v^2 + \sigma_\varepsilon^2)} \right) * \sigma_\varepsilon^4. \qquad \text{(A18)}$$

Now,

$$(v_{r-1,1,d_1} - v_{s_{max}(r),p,d_{max}(s_{max}(r))}) + (d_1 - d_{max}(r)) * \sigma_v^2$$

$$= (v_{r-1,1,d_1} - v_{s_{max}(r),1,d_1}) + (v_{s_{max}(r),1,d_1} - v_{s_{max}(r),p,d_{max}(s_{max}(r))}) + (d_1 - d_{max}(r)) * \sigma_v^2$$

$$= (v_{r-1,1,d_1} - v_{s_{max}(r),1,d_1}) + (v_{s_{max}(r),1,d_1} - v_{s_{max}(r),p,d_{max}(s_{max}(r))}) + (1 - m_{max}(r))d_1 * \sigma_v^2$$

(A19)

As proved for k = 2, it can be shown that

$$(v_{r-1,1,d_1} - v_{s_{max}(r),1,d_1}) < (m_{max}(r) - 1)d_1 * \sigma_v^2 \qquad \text{(A20)}$$

.Since we have assumed that (A14) is true for k = r-1, (A14) is also true for k = $s_{max}(r)$. Therefore,

$$(v_{s_{max}(r),1,d_1} - v_{s_{max}(r),p,d_{max}(s_{max}(r))}) < 0. \qquad \text{(A21)}$$

Now using (A20) and (A21) in (A19), we see that (A19) is less than 0. But (A19) is the numerator of (A18) which then implies that $v_{r,1,d_1} - v_{r,p,d_{max}(r)} < 0$. Therefore we have proved that if (A14) is true for k = r-1, then (A14) is true for k = r. So we conclude that (A14) holds for $1 \le k \le n$.

The proofs of Lemma 2 and 3 can be found in Jensen(1989).

Lemma 2:

Let $f_0(y) = \min\{f_1(y), f_2(y)\}$ where $f_1(y)$ and $f_2(y)$ are nondecreasing in y. Then $f_0(y)$ is nondecreasing in y.

Lemma 3:

Suppose that $Y \sim N[\mu, \sigma^2]$ and h(y) is nonnegative, symmetric about y = 0 and nondecreasing in |y|. Then $E_\mu[h(y)]$ is nondecreasing in $|\mu|$ for any fixed σ^2.

Lemma 4:

Let $f(y) = \min\{(f_1(y), f_2(y), f_3(y), f_4(y)\}$ where $f_i(y)$, $i = 1,2,3,4$ are all nondecreasing in y. Then f(y) is nondecreasing in y.

Proof:

The function $f(y) = \min\{(f_1(y), f_2(y), f_3(y), f_4(y)\}$ can be written as follows

$$f(y) = \min\{\min((f_1(y), f_2(y)), \min(f_3(y), f_4(y))\}.$$

Now, let

$$h_0(y) = \min(f_1(y), f_2(y)) \tag{A22}$$

and

$$h_1(y) = \min(f_3(y), f_4(y)). \tag{A23}$$

Then,

$$f(y) = \min((h_0(y), h_1(y)). \tag{A24}$$

Since $f_1(y)$ and $f_2(y)$ are nondecreasing in y, on applying Lemma 2 to (A22) we see that $h_0(y)$ is nondecreasing in y. Similarly, on applying Lemma 2 to (A23) we see that $h_1(y)$ is nondecreasing in y. Since $h_0(y)$ and $h_1(y)$ are both nondecreasing in y, on applying Lemma 2 to (A24) we see that f(y) is nondecreasing in y. Thus, f(y) is nondecreasing in y.

Theorem 2:

Given the following recursive minimum expected loss function $\Re_{n-k}(y,q)$ formula

$$\Re_{n-k,n}(y,q) = c_1qd_1+\frac{c_1\sigma_v^2d_1^2}{2}+c_3+$$
$$\min\left[c_1y^2d_1+\int\Re_{n-k+1,n}(x,q)f(x:y,w)dx;\ c_2+\int\Re_{n-k+1,n}(x,q)f(x:0,w)dx\right] \tag{A25}$$

for $2 \leq k \leq m-1$ and $m > 2$, where

$$\mathfrak{R}_{n-1,n}(x,q) = c_1qd_1+\frac{c_1\sigma_v^2d_1^2}{2}+c_3+\min(c_1x^2d_1,c_2), \tag{A26}$$

the function $\mathfrak{R}_{n-k,n}(y,q)$ which is the minimum expected loss function for a k-period problem has the following properties:

(i) nonnegative

(ii) symmetric in y about the origin and

(iii) nondecreasing in $|y|$.

Proof:

We shall first prove that (i), (ii) and (iii) hold for $\mathfrak{R}_{n-1,n}(x,q)$. Then, using the method of mathematical induction we will show that if $\mathfrak{R}_{n-k,n}(x,q)$ has properties (i), (ii) and (iii) for $k = i$, then $\mathfrak{R}_{n-k,n}(x,q)$ has properties (i), (ii) and (iii) for $k = i+1$ for $2 \le i \le m-1$.

In $\mathfrak{R}_{n-1,n}(x,q)$, which is given by (A26), c_1, c_2, c_3, q, d_1, and σ_v^2 are positive constants. The function $\mathfrak{R}_{n-1,n}(x,q)$ is quadratic in x for $|x| \le \left|\sqrt{\frac{c_2}{c_1d_1}}\right|$ and is a constant for $|x| > \left|\sqrt{\frac{c_2}{c_1d_1}}\right|$. Therefore, $\mathfrak{R}_{n-1,n}(x,q)$ has properties (i), (ii) and (iii).

Now let us assume that $\mathfrak{R}_{n-k,n}(x,q)$ has properties (i), (ii) and (iii) hold for $k = i$. Now, substituting for $k = i+1$ in (A25), we get

$$\mathfrak{R}_{n-i-1,n}(y,q) = c_1qd_1+\frac{c_1\sigma_v^2d_1^2}{2}+c_3+ \min\left[c_1y^2d_1+\int\mathfrak{R}_{n-i,n}(x,q)f(x{:}y,w)dx;\ c_2+\int\mathfrak{R}_{n-i,n}(x,q)f(x{:}0,w)dx\right] \tag{A27}$$

Since $\mathfrak{R}_{n-i,n}(x,q)$ is assumed to have properties (i), (ii) and (iii), using Lemma 2 we can conclude that $\int\mathfrak{R}_{n-i,n}(x,q)f(x{:}y,w)dx$ has properties (i), (ii) and (iii) for a given w. Further, we see that the first expression in min[., .] of (A27) is nonnegative, symmetric about the origin and strictly increasing in $|y|$. Moreover, the second expression in min[., .] of (A27) is a constant for a given w. Therefore, using Lemma 1 we can conclude that min[., .] has properties (i), (ii) and (iii) and thus, $\Re_{n-i-1,n}(y,q)$ given by (A27) has properties (i), (ii) and (iii). Thus, properties (i), (ii) and (iii) hold for (A25) where $2 \leq i \leq m-1$ and $m > 2$.

APPENDIX B

Computer Program

```
/*
This program computes the adjustment limits, sampling limits and the expected loss for a VSI process
control problem with two sampling intervals d1 and d2 where d2 is an integer multiple of d1.
The following inputs are required:
ac, adjustment cost
sc, sampling cost
oc, off-target cost
d1, the shorter sampling interval,
fsi, fixed sampling interval
obsv, observation error
p_run, total production run time
vsi, multiple relating d2 to d1
q_rows, the number of elements in the conditional variance that is used as an approximation
q_cols, the stage at which this approximation occurs.
To input values of q_rows and q_cols, refer to Table 4.2.1-(b)
*/

/* Built-in functions required by this program */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define NUM3 "q48pt.dat"       /* input file that contains the quadrature points and weights */
#define NUM4 "ltmknw.out"      /* output file that contains the adjustment and sampling limits */
#define NUM5 "elmknw.out"      /* output file that contains the expected loss */
/* Some other variables and functions defined */
#define cf 1
#define ncf 0
#define f13 1
#define f24 0
#define INITLARGE 1000000.0
#define mum(inta,intb) ((inta<intb) ? inta:intb)
#define max(inta,intb) ((inta>intb) ? inta:intb)
#define Q(i1,i2,x) (((i2+i1)+(i2-i1)*qx[0][x])*0.5)
#define QW(i1,i2,x) ((i2-i1)*(qx[1][x])*0.5)
#define TOLERANCE 0.00000001
#define numquadpts 48
#define halfnumqd 24
#define sysv 1.0
#define constant 0.3989422804014327
#define Abs(x) ((x < 0.0) ? (-1.0 * x) : (x))
#define NUMFUNCS 4
```

```
 /* pointers to files */

FILE *f_in, *f_outel, *f_outlt;

/* Matrix type */

typedef struct m_type
{
  char name[10];
  int numrows;
  int numcols;
  double *ptr;
} m_type;

/* prototypes */

int Compute_addr2D( m_type *, int row, int column);
void Store(m_type *info, int row, int column, double value);
double Retrieve(m_type *info, int row, int column);
double *Alloc_matrix(m_type *x);
void Open_files(void);
void Get_input(void);
void sort2(int n, double ra[], double rb[]);
void Quadpts(void);
void Q_qhat(void);
double (*minfunc(int iter, double x, double y, int u ))(int, double, double, int);
double F_err(int, double, double, int);
double Closest(double q, int x);
void First_part(void);
void Second_part(void);
double f1(int, double, double, int);
double f2(int, double, double, int);
double f3(int, double, double, int);
double f4(int, double, double, int);
double XMin(double, double, double);
double Normden(double, double, double);
double Sumnorm(int, double, double, double, double, int, int, int);
double Suminf1(int, double, double, int, int, int);
double Suminf2(int, double, double, int, int, int);
double Find_Inter(double (*g1)(int, double, double, int), double (*g2)(int, double, double, int), double x1,
          double xn, double step,int IT, double vr, int indc);
int Four_Inter(int, double *, double *, double, int);
double Tot_Loss(int itr, double lim1, double lim2, double wvr, double qvr, int indc);
void Inter_Crow(double, int);

/* Global variables */

  double ac, sc, oc, q0, obsv, q_st, *valf;
  int vsi, qct, n_it, q_it, b, qct_13, qct_24, s_rows, IT, q_cols, q_rows, q_what;
  double d1, d2, d_last13, d_last24, qS, qNS, fsi, c_fsi=1.0;
  m_type q_info, *s_info;
  double qx[2][numquadpts], qmu[2][numquadpts], (*funcs[NUMFUNCS])(int,double,double,int);

/* Main body of the program */

int main()
 {
  int i, j, l;
  char CR;

   d_last13=d_last24=-1.0;
   Open_files();  /* function to open files */
   Get_input();   /* function to get the necessary input */
   valf=(double *)malloc(q_cols*sizeof(double));
   H=sqrt(ac/(oc*d1))+5.0;
  d2=vsi*d1;
  fprintf(f_outel,"ac=%lf,oc=%lf,sc=%lf,d1=%lf,d2=%lf,fsi=%lf\n\n",ac,oc,sc,d1,d2,fsi);
  fprintf(f_outel,"obsv=%lf q_rows=%d q_cols=%d\n\n",obsv,q_rows,q_cols);
  fprintf(f_outel,"s_rows=%d,n_it=%d H=%lf\n\n",s_rows,n_it,H);
  fprintf(f_outlt,"ac=%lf,oc=%lf,sc=%lf,d1=%lf,d2=%lf,fsi=%lf\n\n",ac,oc,sc,d1,d2,fsi);
  fprintf(f_outlt,"obsv=%lf q_rows=%d q_cols=%d\n\n",obsv,q_rows,q_cols);
  fprintf(f_outlt,"s_rows=%d,n_it=%d H=%lf\n\n",s_rows,n_it,H);

/* Initializing the matrices */

  for (i=0; i < q_info.numrows; ++i)
   for (j=0; j < q_info.numcols; ++j)
    Store(&q_info,i,j,-1.0);
  for (l=0; l < vsi+1; ++l)
   for (i=0; i < s_info[l].numrows; ++i)
    for (j=0; j < s_info[l].numcols; ++j)
     Store(&s_info[l],i,j,-1.0);

  Quadpts();      /* Reading 24 quadrature points and creating 48 quadrature points */
  Q_qhat();       /* Getting the conditional variance sets */
  if (n_it <= q_cols)
   q_it=n_it;
  else
   q_it=q_cols;
  First_part();   /* getting the first part  from IT=0 to IT < vsi */
  Second_part(); /* getting the second part from IT >= vsi to IT = n_it */
  printf("Mission Accomplished!\n");
  fprintf(f_outel,"Mission Accomplished!\n");
  fprintf(f_outlt,"Mission Accomplished!\n");
  fclose(f_outel);
  fclose(f_outlt);
  return(0);
  }

/* End of the Main body of the program */
/* Compute_addr2D - finds the offset in the chunk of allocated memory that row, column should be (it
will be in the range [0,1,....,n]) */

  int Compute_addr2D(m_type *memory, int row, int column)

  {
   if ((row >= 0) && (row < memory->numrows) && (column >= 0) && (column < memory->numcols))
     return((row * memory->numcols) + column);
   else
    {
     fprintf(f_outel,"\nError in Compute_addr2D, row=%d, column=%d, IT=%d\n", row, column, IT+1);
     printf("Error in Compute_addr2D, row=%d, column=%d, IT=%d\n", row, column, IT+1);
     fprintf(f_outel," numrows=%d, numcols=%d name=%s\n", memory->numrows, memory->numcols,
memory->name);
     printf(" numrows=%d, numcols=%d name=%s\n", memory->numrows, memory->numcols, memory-
>name);
    exit(1);
    }
    return(0);
  }

/* Alloc_matrix - this returns a pointer to a chunk of dynamically-allocated memory, the size of which is
defined by the value passed to us in the size_info struct (numrows, numcols) */

  double *Alloc_matrix(m_type *size_info)
  {
   size_t index ;

   index = size_info->numrows ;
   index *= size_info->numcols ;
   index *= sizeof(double) ;
   return ((double *) malloc(index));
  }

  /* Store - stores "value" in matrices pointed to by info, at row, column */

  void Store(m_type *info, int row, int column, double value)
  {
   *(info->ptr + Compute_addr2D(info, row, column)) = value;
  }

  /* Retrieve - looks in the matrices pointed to by info, and returns the
  value at row, column */

  double Retrieve(m_type *info, int row, int column)
  {
   return(*(info->ptr + Compute_addr2D(info, row, column)));
  }
```

```
/*  function to open files */

  void Open_files(void)
  {
   int l;

   if ((f_in=fopen(NUM3,"r")) == NULL)
      {
       printf("Error_01: Could not open input data file\n");
       exit(1);
      }
   if ((f_outlt=fopen(NUM4,"w")) == NULL)
      {
       printf("Error_02: Could not open output limit data file\n");
       exit(1);
      }
   if ((f_outel=fopen(NUM5,"w")) == NULL)
      {
       printf("Error_03: Could not open output exp.loss data file\n");
       exit(1);
      }
   setbuf(f_outel,NULL);
   setbuf(f_outlt,NULL);
   }

/*   function to get all the necessary input from keyboard! */

  void Get_input(void)
  {
   double p_run, d1_scale, actual_it;
   int l, rc, int_d1, int_prun;
   char CR;

   printf("Don't enter ac < 0.0000001");
   printf("Enter ac,oc,sc,d1,fsi :");
   scanf("%lf,%lf,%lf,%lf,%lf", &ac, &oc, &sc, &d1, &fsi);
   printf("ac=%lf,oc=%lf,sc=%lf,d1=%lf,fsi=%lf\n",ac,oc,sc,d1,fsi);
   printf("Enter obsv, p_run, vsi :");
   scanf("%lf,%lf,%d", &obsv, &p_run, &vsi);
   printf("Enter rows and columns for q matrix: ");
   rc=scanf("%d,%d", &q_rows, &q_cols);
   if (rc != 2)
     {
      printf("Invalid input for q matrix\n");
      fprintf(f_outel,"Invalid input for q matrix\n");
      exit(0);
     }

  /* Calculating n_it, d_last13 and d_last24 */

   p_run=p_run*INITLARGE;
```

```
    int_prun=p_run*1;
    d1_scale=d1*INITLARGE;
    int_d1=d1_scale*1;
    fprintf(f_outel,"p_run=%lf  int_prun=%d  d1_scale=%lf  int_d1=%d\n",p_run,  int_prun,  d1_scale,
int_d1);
    fprintf(f_outlt,"p_run=%lf  int_prun=%d  d1_scale=%lf  int_d1=%d\n",p_run,  int_prun,  d1_scale,
int_d1);
    printf("p_run=%lf int_prun=%d d1_scale=%lf int_d1=%d\n",p_run, int_prun, d1_scale, int_d1);
    n_it=(int_prun/int_d1)+((int_prun % int_d1) != 0 );
    actual_it=p_run/d1_scale;
    if ((int_prun % int_d1) != 0 )
     {
      d_last13=(actual_it-(int_prun/int_d1))*d1;
      d_last24=(vsi-1)*d1+d_last13;
     }
    else
     {
      d_last13=-1.0;
      d_last24=-1.0;
     }
    fprintf(f_outel,"n_it=%d  actual_it=%lf  d_last13=%lf  d_last24=%lf\n",  n_it,  actual_it,  d_last13,
d_last24);
   printf("n_it=%d actual_it=%lf d_last13=%lf d_last24=%lf\n", n_it, actual_it, d_last13, d_last24);
   s_info=(m_type *)malloc((vsi+1)*sizeof(m_type));
   s_rows=q_rows*13;
   printf("q_rows=%d q_cols=%d s_rows=%d n_it=%d\n",
     q_rows, q_cols, s_rows, n_it);
   scanf("%c",&CR);

  /* Checking to see if proper memory allocated wrt reusing matrices */

   for (l=0; l < vsi+1; ++l)
   {
     strcpy(s_info[l].name,"sinfox");
     s_info[l].name[5]=48+l;
     s_info[l].numrows = s_rows;
     s_info[l].numcols = numquadpts+1;
     s_info[l].ptr = Alloc_matrix( &s_info[l] );
     if (s_info[l].ptr == NULL)
      {
       fprintf(f_outel,"Need more memory!\n");
       exit(0);
      }
   }
   strcpy(q_info.name,"qinfo");
   q_info.numrows = q_rows;
   q_info.numcols = q_cols;
   q_info.ptr = Alloc_matrix( &q_info );
   if (q_info.ptr == NULL)
    {
     fprintf(f_outel,"Need more memory for the q chunk!\n");
```

```
      exit(0);
    }
  }

/* function that reads the 24 abs. and wts. from q24pt.dat file; Creats the entire 48 quad. pts */

   void Quadpts(void)
   {
    int i, quady;
    double q[2][halfnumqd], *q1, *q2;

    quady=numquadpts-1;
    for (i=0; i<halfnumqd; ++i)
    {
     fscanf(f_in, "%lf %lf\n", &q[0][i], &q[1][i]);
     qx[0][i] = q[0][i]; qx[0][quady-i] = -q[0][i];
     qx[1][i] = q[1][i]; qx[1][quady-i] = q[1][i];
    }

   /* Converting mu = 0 to H=15 into quad pts */

     for (i = 0; i < numquadpts; ++i)
     {
      qmu[0][i] = (H+(H*qx[0][i]))*0.5; qmu[1][i] = H*qx[1][i]*0.5;
     }
     q1=qmu[0]; q2=qmu[1];          /* sorting the quad pts. by abs. sort2(numquadpts,q1,q2), double *q1,
*q2 */
     sort2(numquadpts,q1,q2);
   }

/* function that calculates the q's - condl.vars. & qh's - Est. condl. vars. */

   void Q_qhat(void)
   {
    int k, i, j, s;
    double NS1,S1,S0,val_q,val1_q,val2_q, value_q;
    char CR;

    S0 = sqrt(0.25 + (obsv/(c_fsi*sysv)));
    q_st = (c_fsi*sysv)*(S0 - 0.5);
    d2=vsi*d1;
    NS1 = sqrt(0.25 + (obsv/(d2*sysv)));  S1 = sqrt(0.25 + (obsv/(d1*sysv)));
    qNS = (d2*sysv)*(NS1 - 0.5);  qS = (d1*sysv)*(S1 - 0.5);
    if (fsi==d1)
     q_what=0;
    else
    if (fsi==d2)
     q_what=1;
    else
    if ((fsi != d1) && (fsi != d2))
     q_what=2;
```

```
fprintf(f_outel,"q_what=%d fsi=%lf\n",q_what,fsi);
valf[0]=q_st;
Store(&q_info,0,0,q_st);
fprintf(f_outel,"q_st=%lf qS=%lf qNS=%lf\n\n",q_st,qS,qNS);
fprintf(f_outlt,"q_st=%lf qS=%lf qNS=%lf\n\n",q_st,qS,qNS);
for (IT=1; IT < vsi; ++IT)
  {
  valf[IT]=((valf[IT-1]+fsi*sysv)/(valf[IT-1]+fsi*sysv+obsv))*obsv;
   if (IT==1)
    {
  value_q = ((q_st+d1*sysv)/(q_st+d1*sysv+obsv))*obsv;
    Store(&q_info,0,1,value_q);
    }
   else
    {
    val_q = Retrieve(&q_info,0,IT-1);
    value_q = ((val_q+d1*sysv)/(val_q+d1*sysv+obsv))*obsv;
    Store(&q_info,0,IT,value_q);
    }
  }
for (IT=vsi; IT < q_info.numcols; ++IT)
  {
  valf[IT]=((valf[IT-1]+fsi*sysv)/(valf[IT-1]+fsi*sysv+obsv))*obsv;
  for (i=0; i < q_info.numrows; ++i)
   {
    val_q=Retrieve(&q_info,i,IT-1);
    if (val_q > -1.0)
  {
   val1_q=((val_q+d1*sysv)/(val_q+d1*sysv+obsv))*obsv;
   Store(&q_info,i,IT,val1_q);
  }
    else
  {
   k=i;
   break;
  }
   }
 for (i=0; i < q_info.numrows; ++i)
  {
   val_q=Retrieve(&q_info,i,IT-vsi);
   if (val_q > -1.0)
    {
  val2_q=((val_q+d2*sysv)/(val_q+d2*sysv+obsv))*obsv;
  Store(&q_info,k,IT,val2_q);
  k=k+1;
    }
   else
    {
  break;
    }
  }
```

```
    }
  }

/* The following function finds the func. that gives the min. functional value at a particular x */

  double (*minfunc(int iter, double x, double y, int u))(int, double, double, int)
  {
   double min1=0.0, val=0.0;
   int i, min_i;

   for (i = 0 ; i < NUMFUNCS ; ++i)
    {
     if (funcs[i] != NULL)
      {
       val = (funcs[i])(iter,x,y,u);
       if (i == 0)
        {
         min1 = val;  min_i = 0;
        }
       else
       if (min1 > val)
        {
         min1 = val;  min_i = i;
        }
      }
    }
   return (funcs[min_i]);
  }

/* Function to get the closest value to q */

  double Closest(double q, int x)
  {
   int i, min_ndx=0;
   double min_v=INITLARGE, next_v=0.0;
   char CR;

   for (i=0; i < q_info.numrows; ++i)
    {
       next_v=fabs(q-Retrieve(&q_info,i,x));
       if ( next_v < min_v )
       {
         min_ndx = i; min_v = next_v;
       }
    }
   return(min_ndx);
  }
```

```
/* function - to get the exp. losses for all IT <= multiple vsi, we have only 2 Exp. losses until IT=vsi */

  void First_part(void)
  {
   int i, q_iter, j, ndx_q;
   double EL1=0.0, tot_el, tot_elc, i1, i2, lim=-1.0, qvar1, wvar, wvar1;
   double LTy1=-1.0,LTy2=-1.0,LT1V1=-1.0,LT1V2=-1.0,LT2V1=-1.0,LT2V2=-1.0, var1, var2,tot;
   double (*minfl1)(int, double, double, int)=&F_err,  (*minf2)(int, double, double, int)=&F_err;
   char CR;

   for (IT = 0; IT < (vsi-1); ++IT)
    {
    LTy1=LTy2=LT1V1=LT1V2=LT2V1=LT2V2=-1.0;
   minfl1=&F_err;  minf2=&F_err;
   for (qct=0; qct < q_info.numrows; ++qct)
     {
       if (IT > (n_it-q_it))
     q_iter=n_it-IT-1;
      else
      if (IT <= (n_it-q_it))
     q_iter=q_it-1;
      else
        {
     fprintf(f_outel,"Another case ???\n");
     exit(1);
        }
     q0=Retrieve(&q_info,qct,q_iter); qct_13=qct;
     var1 = ((q0+d1*sysv)/(q0+d1*sysv+obsv))*obsv;
     var2 = ((q0+d2*sysv)/(q0+d2*sysv+obsv))*obsv;
     wvar=pow((q0+fsi*sysv),2.0)/(q0+fsi*sysv+obsv);

 /* to get the closest q in the last col. of the q matrix when you are within vsi of the last cols. in the iters.
*/

     if (IT == (n_it-q_it))
      {
        ndx_q=Closest(var1,(q_it-1)); qct_13=ndx_q;
      }
      if (q0 >= 0.0)
         EL1=f1(IT,0.0,q0,ncf);
      if (q0 < 0.0)
        {
     if (((IT==0) && ( n_it < q_cols)) || (IT != 0))
      {
        if (qct != 1)
         {
       fprintf(f_outel,"STAGE=%d tot_el=%6.10lf\n",IT+1, tot_el);
       fprintf(f_outlt,"STAGE=%d LTy1=%2.10lf LTy2=%2.10lf\n",IT+1, LTy1, LTy2);
       if (minfl1==funcs[2])
        fprintf(f_outlt,"when < LTy1 - AcS");
       else
      fprintf(f_outlt,"Always AS");
     if ((minf2 == funcs[2]) || (minf2 == funcs[3]))
      {
       if (minf2==funcs[2])
       {
        fprintf(f_outlt,"minf2=funcs[2]\n");
        exit(1);
       }
        fprintf(f_outlt,"Beyond LTy1, the function is not a constant!\n");
      }
       else
     fprintf(f_outlt," : when > LTy1 - AS\n\n");
       }  /* corresponds to ' if (qct != 1)' */
        break; /* should break from the qct loop! */
    }  /* corresponds to 'if .. && .. || ' */
   else
   if ((IT==0) && (n_it >= q_cols))
    {
     fprintf(f_outel,"Error in col. q_it-1\n");
     exit (1);
    }
     }

/* to find the only pt. of intersection-only 1 pt. of intersection this is used to find the exp.loss for my case
*/

   LTy1=Find_Inter(f1,f3,H,0.0,1.0,IT,q0,ncf);  LTy2=LTy1;

/* Calculating the exp.loss */

  funcs[0]=f1; funcs[2]=f3; funcs[1]=funcs[3]=NULL;
  tot_el=0.0;
  tot_el=Tot_Loss(IT,LTy1,LTy2,wvar,q0,ncf);

/* Calculating Crowder's */

  if (qct==0)
   {
    tot_elc=0.0;
    if (f1(IT,0.0,valf[q_iter],cf) < f3(IT,0.0,valf[q_iter],cf))
     {
      fprintf(f_outel,"MINIMUM FUNC. IS A CONSTANT (Crowder's)\n");
      exit(0);
     }
   wvar1=pow((valf[q_iter]+fsi*sysv),2.0)/(valf[q_iter]+fsi*sysv+obsv);
   Inter_Crow(valf[q_iter],IT);
   lim=Retrieve(&s_info[IT],qct_13*13+12,numquadpts);
   tot_elc=Tot_Loss(IT,lim,lim,wvar1,valf[q_iter],cf);
   fprintf(f_outlt,"crow_lim=%2.10lf stage=%d\n",lim,IT+1);
   }
```

```
/* printing tot_el tot_elc limits */

  if (qct==0)
   {
    fprintf(f_outel,"STAGE=%d tot_el=%6.10lf tot_elc=%6.10lf\n",IT+1, tot_el, tot_elc);
    fprintf(f_outlt,"STAGE=%d LTy1=%2.10lf LTy2=%2.10lf\n",IT+1, LTy1, LTy2);
   }
  if ((q0 > 0.0) && (qct==q_info.numrows-1))
   {
    fprintf(f_outel,"STAGE=%d tot_el=%6.10lf\n",IT+1, tot_el);
    fprintf(f_outlt,"STAGE=%d LTy1=%2.10lf LTy2=%2.10lf\n",IT+1, LTy1, LTy2);
   }

/* to find and store the limits to be used in the integral for the next stage & the min. funcs.in (-LT1,LT1)
and (LT1,inf)*/

   LT1V1=Find_Inter(f1,f3,H,0.0,1.0,IT,var1,ncf);
   LT2V1=LT1V1;
   Store (&s_info[IT],qct*13,numquadpts,LT1V1);
   Store (&s_info[IT],qct*13+1,numquadpts,LT2V1);

  if (qct==0)
   {
   qvar1=((valf[q_iter]+fsi*sysv)/(valf[q_iter]+fsi*sysv+obsv))*obsv;
   Inter_Crow(qvar1,IT);
   lim=Retrieve(&s_info[IT],qct*13+12,numquadpts);
   fprintf(f_outlt,"stage=%d c_lt=%lf\n",IT+1,lim);
   }
   LT1V2=Find_Inter(f1,f3,H,0.0,1.0,IT,var2,ncf);
   LT2V2=LT1V2;
   Store (&s_info[IT],qct*13+2,numquadpts,LT1V2);
   Store (&s_info[IT],qct*13+3,numquadpts,LT2V2);
   minf11=minfunc(IT,0.0,q0,ncf);
   if ((qct==0) || ((q0 > 0.0) && (qct==q_info.numrows-1)))
    {
    if (minf11==funcs[2])
     fprintf(f_outlt,"when < LTy1 - AcS");
    else
     fprintf(f_outlt,"Always AS");
    }
   minf11=minfunc(IT,0.0,var1,ncf);
   i1=-LT1V1; i2=LT1V1;
   for (i = 0; i < numquadpts; ++i)
     Store(&s_info[IT],qct*13,i,(minf11)(IT,Q(i1,i2,i),var1,ncf));
   minf11=minfunc(IT,0.0,var2,ncf);
   i1=-LT1V2; i2=LT1V2;
   for (i = 0; i < numquadpts; ++i)
     Store(&s_info[IT],qct*13+5,i,(minf11)(IT,Q(i1,i2,i),var2,ncf));
   minf2=minfunc(IT,LTy1+1.0,q0,ncf);
  if ((qct==0) || ((q0 > 0.0) && (qct==q_info.numrows-1)))
   {
```

```
   if ((minf2 == funcs[2]) || (minf2 == funcs[3]))
    {
     if (minf2==funcs[2])
      {
       fprintf(f_outlt,"minf2=funcs[2]\n");
       exit(1);
      }
     fprintf(f_outlt,"Beyond LTy1, the function is not a constant!\n");
    }
   else
     fprintf(f_outlt," : when > LTy1 - AS\n\n");
   }
   minf2=minfunc(IT,LT2V1+1.0,var1,ncf);
   i1=0.0; i2=LT2V1;
   for (i = 0; i < numquadpts; ++i)
     Store(&s_info[IT],qct*13+3,i,(minf2)(IT,Q(i1,i2,i),var1,ncf));
   i1=-LT2V1; i2=0.0;
   for (i = 0; i < numquadpts; ++i)
     Store(&s_info[IT],qct*13+4,i,(minf2)(IT,Q(i1,i2,i),var1,ncf));
   minf2=minfunc(IT,LT2V2+1.0,var2,ncf);
   i1=0.0; i2=LT2V2;
   for (i = 0; i < numquadpts; ++i)
     Store(&s_info[IT],qct*13+8,i,(minf2)(IT,Q(i1,i2,i),var2,ncf));
   i1=-LT2V2; i2=0.0;
   for (i = 0; i < numquadpts; ++i)
    Store(&s_info[IT],qct*13+9,i,(minf2)(IT,Q(i1,i2,i),var2,ncf));
   }     /* qct loop ends here! */
  }                          /* IT loop ends here! */
 }

/* to find the 4 exp. losses for IT > info.nummat, the multiple */

 void Second_part(void)
 {
 int i, j, n, l, q_iter, counter_m=0, counter=0, ndx_q, qcty,  iter_1, iter, q_itfsi;
 double EL1=0.0, EL2=0.0, tot_el,tot_elc,LT1=-1.0,LT2=-1.0,mid=0.0,i1, i2, mid1, mid2, wvar1, wvar;
 double LTy1=-1.0,LTy2=-1.0,LT1V1=-1.0,LT1V2=-1.0,LT2V1=-1.0,LT2V2=-1.0,var1,var2,val,qvar1;
 double (*minfl1)(int, double, double, int)=&F_err,(*minfl2)(int, double,double, int)=&F_err,q00,q_gr;
 double (*mfl1)(int, double, double, int)=&F_err, (*mfl2)(int, double, double, int)=&F_err, lim=-1.0;
 double (*mf2)(int, double, double, int)=&F_err, (*minf2)(int, double, double, int)=&F_err, q_m=0.0;
 char CR;

 if (n_it == q_it)
  {
   for (qcty=0;qcty < q_info.numrows; ++qcty)
    {
  q_m = Retrieve(&q_info,qcty,(n_it-1));
  if (q_m != -1.0)
    counter_m = counter_m +1;
    }
   b=counter_m;
   }
  else
   b = q_info.numrows;
 q_m=0.0; counter_m=0;
 for (IT = (vsi-1); IT < n_it; ++IT)
  {
  LTy1=LTy2=LT1V1=LT1V2=LT2V1=LT2V2=-1.0; LT1=LT2=0.0;
  minfl1=&F_err; minfl2=&F_err; mfl1=&F_err;
  mfl2=&F_err; mf2=&F_err; minf2=&F_err;
 if (IT > (n_it-q_it))
   q_itfsi=n_it-IT-1;
 else
 if (IT <= (n_it-q_it))
   q_itfsi=q_it-1;
  if (IT==(vsi-1))
   iter=IT;
  else
   iter=vsi;
  counter_m=0; counter=0;
 if (IT >= (n_it-vsi))
 {
  if (abs(n_it-IT-q_it) >= vsi)
   {
    for (qcty=0; qcty < q_info.numrows; ++qcty)
     {
   q_m = Retrieve(&q_info,qcty,(n_it-IT-1+vsi));
   if (q_m != -1.0)
     counter_m = counter_m +1;
     }
   }
 }
  if (IT >= (n_it-q_it))
   {
    for (qcty=0; qcty < q_info.numrows; ++qcty)
     {
   q00 = Retrieve(&q_info,qcty,(n_it-IT-1));
   if (q00 != -1.0)
     counter = counter +1;
     }
   }
  if ((IT >= (n_it-q_it+vsi)) && (IT < (n_it-vsi)))
     b=b-counter;
  for (qct=0; qct < q_info.numrows; ++qct)
   {
    if ((vsi < (n_it-q_it+1)) && (IT < (n_it-q_it+1)))
   q_iter = q_it-1;
    else
   q_iter = n_it-IT-1;
    q0=Retrieve(&q_info,qct,q_iter);

  if (q0 < 0.0)
```

```
   {
    if(qct == 0)
    {
    fprintf(f_outel,"q0 <= 0.0 & qct=%d\n",qct);
    exit(1);
    }
    else
     q_gr=Retrieve(&q_info,qct-1,q_iter);
     if (qct != 1)
      {
     fprintf(f_outel,"STAGE=%d tot_el=%6.10lf\n",IT+1, tot_el);
     fprintf(f_outlt,"STAGE=%d LTy1=%2.10lf LTy2=%2.10lf\n",IT+1, LTy1, LTy2);
  if (mfl1==funcs[2])
    fprintf(f_outlt,"when < LTy1 - AcS*\n");
  else
  if (mfl1==funcs[3])
    fprintf(f_outlt,"when < LTy1 - AcSc*\n");
  if (mfl2==funcs[2])
    fprintf(f_outlt," : Bet LTy1 & LTy2 - AcS*\n");
  else
  if (mfl2==funcs[3])
    fprintf(f_outlt," : Bet LTy1 & LTy2 - AcSc*\n");
  if (mf2==funcs[0])
    fprintf(f_outlt," : when > LTy2 - AS*\n\n");
  else
  if (mf2==funcs[1])
    fprintf(f_outlt," : when > LTy2 - ASc*\n\n");
  else
  if ((mf2==funcs[2]) || (mf2==funcs[3]))
     {
    fprintf(f_outel,"Beyond LTy2, the function is not a constant!\n");
    printf("Beyond LTy2, the function is not a constant!\n");
      }
     }
     for (i=0; i < 13; ++i)
       for (j=0; j < s_info[iter].numcols; ++j)
         Store(&s_info[iter],qct*13+i,j,-1.0);
  break;
      }
 qct_13=qct;
 var1 = ((q0+d1*sysv)/(q0+d1*sysv+obsv))*obsv;
 var2 = ((q0+d2*sysv)/(q0+d2*sysv+obsv))*obsv;
 wvar = pow((q0+fsi*sysv),2.0)/(q0+fsi*sysv+obsv);
 EL1=f1(IT,0.0,q0,ncf);

/* getting the appropriate qct to be able to restore the correct previous m step exp.loss and limits) */

    iter_1 = mum((n_it-q_it+vsi),(n_it-vsi));
    if ((IT >= (vsi-1)) && (IT < (n_it-q_it+1)))
     {
   qct_24=qct;
```

```
      if (IT == (n_it-q_it))
        {
         ndx_q=Closest(var1,(q_it-1));
         qct_13=ndx_q;

         ndx_q=Closest(var2,(q_it-1));
         qct_24=ndx_q;
        }
       }
      else
      if ((IT > ( n_it-q_it)) && (IT < iter_1))
        {
     ndx_q = Closest(var2,(q_it-1));
      qct_24 = ndx_q;
        }
      else
      if ((IT >= (n_it-q_it+vsi)) && (IT < (n_it-vsi)))
       qct_24 = b+qct;
      else
      if (IT >= (n_it-vsi))
       {
     if (abs(n_it-IT-q_it) < vsi)
       {
         ndx_q = Closest(var2,(q_it-1));
         qct_24 = ndx_q;
       }
     else
         qct_24 = counter_m-1;
        }
      else
        {
      fprintf(f_outel,"IT problem in f2!\n");
      exit(1);
        }

   EL2=f2(IT,0.0,q0,ncf);
   val=mum(f3(IT,0.0,q0,ncf),f4(IT,0.0,q0,ncf));
    if ((EL1 < val) || (EL2 < val))
     {
      fprintf(f_outel,"MINIMUM FUNC. IS A CONSTANT\n");
      if (EL1 < val)
        fprintf(f_outel,"Adjust and Sample always!\n");
      if (EL2 < val)
        fprintf(f_outel,"Adjust and Don't sample always!\n");
      exit(0);
     }

/* Calculating mine and Crowder's exp.loss */

   funcs[0]=f1; funcs[1]=f2; funcs[2]=f3; funcs[3]=f4;
   Four_Inter(IT,&LT1,&LT2,q0,ncf);
```

```
  LTy1=LT1; LTy2=LT2;
  tot_el = 0.0;
  tot_el=Tot_Loss(IT,LTy1,LTy2,wvar,q0,ncf);

/* getting Crowder's exp. loss for sc=0 & the pts. of intersection */

    if (qct==0)
    {
     if (f1(IT,0.0,valf[q_itfsi],cf) < f3(IT,0.0,valf[q_itfsi],cf))
      {
       fprintf(f_outel,"MINIMUM FUNC. IS A CONSTANT (Crowder's)\n");
       exit(0);
      }
     tot_elc=0.0;
     wvar1=pow((valf[q_itfsi]+fsi*sysv),2.0)/(valf[q_itfsi]+fsi*sysv+obsv);
     Inter_Crow(valf[q_itfsi],iter);
     lim=Retrieve(&s_info[iter],qct_13*13+12,numquadpts);
     fprintf(f_outlt,"c_lt=%lf\n",lim);
     tot_elc=Tot_Loss(iter,lim,lim,wvar1,valf[q_itfsi],cf);
     qvar1=((valf[q_itfsi]+fsi*sysv)/(valf[q_itfsi]+fsi*sysv+obsv))*obsv;
     Inter_Crow(qvar1,iter);
    }

/* printing tot_el tot_elc */

    if (qct==0)
      fprintf(f_outel,"stage=%d tot_el=%6.10lf tot_elc=%6.10lf\n", IT+1, tot_el, tot_elc);
    else
    if ((q0 > 0.0) && (qct==q_info.numrows-1))
      fprintf(f_outel,"STAGE=%d tot_el=%6.10lf\n", IT+1,tot_el);

/* to find the 4 pts. of intersection now */

   funcs[0]=f1; funcs[1]=f2; funcs[2]=f3; funcs[3]=f4;
   Four_Inter(IT,&LT1,&LT2,q0,ncf);
   LTy1=LT1; LTy2=LT2;
   if (LTy1 < TOLERANCE)
     LTy1=TOLERANCE;

/*printing limits */

   if ((qct==0) || ((q0 > 0.0) && (qct==q_info.numrows-1)))
     fprintf(f_outlt,"stage=%d LTy1=%2.10lf LTy2=%2.10lf\n",IT+1, LTy1, LTy2);

     Four_Inter(IT,&LT1,&LT2,var1,ncf);
     LT1V1=LT1; LT2V1=LT2;
     if (LT1V1 < TOLERANCE)
        LT1V1=TOLERANCE;
     Store (&s_info[iter],qct*13,numquadpts,LT1V1);
     Store (&s_info[iter],qct*13+1,numquadpts,LT2V1);
     LT1=LT2=0.0;
```

```
 Four_Inter(IT,&LT1,&LT2,var2,ncf);
 LT1V2=LT1; LT2V2=LT2;
if (LT1V2 < TOLERANCE)
   LT1V2=TOLERANCE;
Store (&s_info[iter],qct*13+2,numquadpts,LT1V2);
Store (&s_info[iter],qct*13+3,numquadpts,LT2V2);

/* to find the min. funcs.in (-LT1,LT1); (LT1,LT2) and (LT2,inf) + storing */

  if (LTy1==TOLERANCE)
   {
    mf11=minfunc(IT,0.00000011,q0,ncf);  minf11=minfunc(IT,0.00000011,q0,ncf);
   }
  else
   {
   mf11=minfunc(IT,0.0,q0,ncf); minf11=minfunc(IT,0.0,q0,ncf);
   }
 if ((qct==0) || ((q0 > 0.0) && (qct==q_info.numrows-1)))
 {
  if (minf11==funcs[2])
   fprintf(f_outlt,"when < LTy1 - AcS");
  else
  if (minf11==funcs[3])
   fprintf(f_outlt,"when < LTy1 - AcSc");
 }
  if (LTy1==TOLERANCE)
 minf11=minfunc(IT,0.00000011,var1,ncf);
  else
 minf11=minfunc(IT,0.0,var1,ncf);
i1=-LT1V1; i2=LT1V1;
 for (i = 0; i < numquadpts; ++i)
   Store (&s_info[iter],qct*13,i,(minf11)(IT,Q(i1,i2,i),var1,ncf));
 if (LTy1==TOLERANCE)
 minf11=minfunc(IT,0.00000011,var2,ncf);
 else
 minf11=minfunc(IT,0.0,var2,ncf);
i1=-LT1V2; i2=LT1V2;
 for (i = 0; i < numquadpts; ++i)
  Store (&s_info[iter],qct*13+5,i,(minf11)(IT,Q(i1,i2,i),var2,ncf));
mid=(LTy1+LTy2)*0.5;
minf12=minfunc(IT,mid,q0,ncf);
mf12=minfunc(IT,mid,q0,ncf);
if ((qct==0) || ((q0 > 0.0) && (qct==q_info.numrows-1)))
{
if (minf12==funcs[2])
 fprintf(f_outlt," : Bet LTy1 & LTy2 - AcS");
else
if (minf12==funcs[3])
 fprintf(f_outlt," : Bet LTy1 & LTy2 - AcSc");
}
 mid1=(LT1V1+LT2V1)*0.5;
```

```
    minf12=minfunc(IT,mid1,var1,ncf);
  i1=LT1V1; i2=LT2V1;
    for (i = 0; i < numquadpts; ++i)
     Store (&s_info[iter],qct*13+1,i,(minf12)(IT,Q(i1,i2,i),var1,ncf));
  i1=-LT2V1; i2=-LT1V1;
    for (i = 0; i < numquadpts; ++i)
     Store (&s_info[iter],qct*13+2,i,(minf12)(IT,Q(i1,i2,i),var1,ncf));
    mid2=(LT1V2+LT2V2)*0.5;
    minf12=minfunc(IT,mid2,var2,ncf);
  i1=LT1V2; i2=LT2V2;
    for (i = 0; i < numquadpts; ++i)
     Store (&s_info[iter],qct*13+6,i,(minf12)(IT,Q(i1,i2,i),var2,ncf));
  i1=-LT2V2; i2=-LT1V2;
    for (i = 0; i < numquadpts; ++i)
     Store (&s_info[iter],qct*13+7,i,(minf12)(IT,Q(i1,i2,i),var2,ncf));
   minf2=minfunc(IT,LTy2+1.0,q0,ncf);
  if ((qct==0) || ((q0 > 0.0) && (qct==q_info.numrows-1)))
  {
   if (minf2==funcs[0])
    fprintf(f_outlt," : when > LTy2 - AS\n\n");
   else
   if (minf2==funcs[1])
    fprintf(f_outlt," : when > LTy2 - ASc\n\n");
   else
   if ((minf2==funcs[2]) || (minf2==funcs[3]))
    {
     fprintf(f_outel,"Beyond LTy2, the function is not a constant!\n");
     exit(1);
    }
  }
    minf2=minfunc(IT,LT2V1+1.0,var1,ncf);
  i1=0.0; i2=LT2V1;
    for (i = 0; i < numquadpts; ++i)
     Store(&s_info[iter],qct*13+3,i,(minf2)(IT,Q(i1,i2,i),var1,ncf));
  i1=-LT2V1; i2=0.0;
    for (i = 0; i < numquadpts; ++i)
     Store(&s_info[iter],qct*13+4,i,(minf2)(IT,Q(i1,i2,i),var1,ncf));
    minf2=minfunc(IT,LT2V2+1.0,var2,ncf);
  i1=0.0; i2=LT2V2;
    for (i = 0; i < numquadpts; ++i)
     Store(&s_info[iter],qct*13+8,i,(minf2)(IT,Q(i1,i2,i),var2,ncf));
  i1=-LT2V2; i2=0.0;
    for (i = 0; i < numquadpts; ++i)
     Store(&s_info[iter],qct*13+9,i,(minf2)(IT,Q(i1,i2,i),var2,ncf));
  } /* qct loop ends here! */

/* Reusing the matrices now! */

  if (IT >= vsi)
   {
    for (n=0; n < vsi; ++n)
```

```
      for (i=0; i < (s_info[n].numrows*s_info[n].numcols); ++i)
         s_info[n].ptr[i] = s_info[n+1].ptr[i];
   }
  } /* IT ends here */
 }
```

/* error function that wil l be returned if the pointer to the function is not taking the right pointer, i.e, it takes the NULL pointer */

```
  double F_err(int IT, double z, double vr, int indc)
  {
  fprintf(f_outel,"Error in the pointer to a function, check minfl1, minfl2 and minf2\n");
  fprintf(f_outlt,"Error in the pointer to a function, check minfl1, minfl2 and minf2\n");
  fprintf(f_outel,"IT=%d z=%lf vr=%lf indc=%d\n", IT+1, z, vr, indc);
  fprintf(f_outlt,"IT=%d z=%lf vr=%lf indc=%d\n", IT+1, z, vr, indc);
  exit(1);
  return(0.0);
  }

  /* function fl - Equ1 which is a constant for a particular iteration */

double fl(int IT, double z, double vr, int indc)
 {
 int w;
 double v,m,LT1=-1.0,LT2=-1.0,i1=0.0,i2=0.0,y=0.0,fsum=0.0,sum1,sum2a,sum2b,sum3a,sum3b,k1,d0;

   z=z;
   if (indc==cf)
    {
     if ((q_what==2) || (q_what==0))
      d0=fsi;
     else
     if (q_what==1)
      d0=d2;
     else
      {
      fprintf(f_outel,"q_what invalid in fl\n");
      exit(1);
      }
    }
   else
     d0=d1;
  k1 = (vr+d0*sysv)/(vr+d0*sysv+obsv); v = k1*(vr+d0*sysv); m = 0.0;
   if ((v <= 0.0) || (vr <= 0.0))
    {
     fprintf(f_outel,"In fl v=%3.15lf vr=%3.15lf indc=%d IT=%d\n",v,vr,indc,IT+1);
     exit(1);
    }
  if (IT == 0)
   {
    if (d_last13 != -1.0)
```

```
      d0=d_last13;
     fsum = ac+0.5*oc*sysv*(d0*d0)+vr*oc*d0+sc;
     return(fsum);
    }
     else
      {
    if (IT < vsi)
      w = IT-1;
    else
      w = vsi-1;
    if (indc==cf)
      {
       LT1 = Retrieve(&s_info[w],(qct_13)*13+12,numquadpts); LT2 = LT1;
      }
    else
    if (indc==ncf)
     {
      LT1 = Retrieve(&s_info[w],(qct_13)*13,numquadpts);
      LT2 = Retrieve(&s_info[w],(qct_13)*13+1,numquadpts);
     }
    else
     {
      fprintf(f_outel,"No such indc in f1\n");
      exit(1);
     }

/* Getting integral on (-LT1,LT1) */

    sum1=0.0;
    i1 = -LT1; i2 = LT1;
    sum1 = Sumnorm(w,i1,i2,v,m,qct_13,f13,indc);
    if (sum1 < 0.0)
    {
     fprintf(f_outel,"In f1 sum1=%lf\n",sum1);
     exit(1);
    }

/* ONLY when LT1 != LT2 is the integral in (LT1,LT2) and (-LT2,-LT1) calculated */

    if (LT1 == LT2)
      sum2a = sum2b = 0.0;
    else
    if (LT1 != LT2 && LT2 != -1.0)
     {
      i1 = LT1; i2 = LT2;
      sum2a = Sumnorm(w,i1,i2,v,m,qct_13,f13,indc);
      i1 = -LT2; i2 = -LT1;
      sum2b = Sumnorm(w,i1,i2,v,m,qct_13,f13,indc);
       if ((sum2a < 0.0) || (sum2b < 0.0))
        {
         fprintf(f_outel,"In f1 sum2a=%lf sum2b=%lf\n",sum2a,sum2b);
```

```
          exit(1);
         }
       }

 /* Integral in (LT2,inf) and (-inf,-LT2) */

      sum3a=sum3b=0.0;
      y = (LT2-m)/(sqrt(v));
      if (y >= 0.0)
       {
        i1 = 0.0; i2 = y;
       }
      else
       {
        i1 = y; i2 =0.0;
       }
        sum3a = Suminf1(w,i1,i2,qct_13,f13,indc);
      y = (-LT2-m)/(sqrt(v));
      if (y >= 0.0)
       {
        i1 = 0.0; i2 = y;
       }
      else
       {
        i1 = y; i2 =0.0;
       }
      sum3b = Suminf2(w,i1,i2,qct_13,f13,indc);
        if ((sum3a < 0.0) || (sum3b < 0.0))
         {
          fprintf(f_outel,"In f1 sum3a=%lf sum3b=%3.20lf\n",sum3a,sum3b);
          exit(1);
         }
        }
      fsum=sc+ac+0.5*oc*sysv*(d0*d0)+vr*oc*d0+sum1+sum2a+sum2b+sum3a+sum3b;
      return(fsum);
    }

/* function f2 - Equ1 which is a constant for a particular iteration */

 double f2(int IT, double z, double vr, int indc)
   {
    int w;
    double v, m, LT1=-1.0, LT2=-1.0, i1=0.0, i2=0.0, y=0.0,  fsum=0.0;
    double sum1, sum2a, sum2b, sum3a, sum3b, k1, d00;

      z=z;
      if (indc==cf)
       {
        fprintf(f_outel,"indc can't be =cf in f2\n");
        exit(1);
       }
   k1 = (vr+d2*sysv)/(vr+d2*sysv+obsv); v = k1*(vr+d2*sysv); m = 0.0;
   if ((v <= 0.0) || (vr <= 0.0))
    {
     fprintf(f_outel,"In f2 v=%3.15lf vr=%3.15lf indc=%d IT=%d\n",v,vr,indc,IT+1);
     exit(1);
    }
   if (IT == (vsi-1))
    {
     if (d_last24 != -1.0)
    d00=d_last24;
     else
    d00=d2;
      fsum = ac+0.5*oc*sysv*(d00*d00)+vr*oc*d00+sc;
      return(fsum);
    }
   else
   if (IT >= vsi)
     {
      w=0;
      LT1=Retrieve(&s_info[w],(qct_24)*13+2,numquadpts);
      LT2=Retrieve(&s_info[w],(qct_24)*13+3,numquadpts);

/* Getting integral on (-LT1,LT1) */

      sum1=0.0;
      i1=-LT1; i2=LT1;
      sum1 = Sumnorm(w,i1,i2,v,m,qct_24,f24,indc);
   if (sum1 < 0.0)
   {
    fprintf(f_outel,"In f2 sum1=%lf\n",sum1);
    exit(1);
    }

/* ONLY when LT1 != LT2 is the integral in (LT1,LT2) and (-LT2,-LT1) calculated */

   if (LT1==LT2)
    sum2a = sum2b = 0.0;
     else
     if (LT1 != LT2 && LT2 != -1.0)
    {
     i1=LT1; i2=LT2;
     sum2a = Sumnorm(w,i1,i2,v,m,qct_24,f24,indc);
     i1=-LT2; i2=-LT1;
     sum2b = Sumnorm(w,i1,i2,v,m,qct_24,f24,indc);
     if ((sum2a < 0.0) || (sum2b < 0.0))
      {
      fprintf(f_outel,"In f2 sum2a=%lf sum2b=%lf\n",sum2a,sum2b);
      exit(1);
      }
    }
```

```
/* Integral in (LT2,inf) and (-inf,-LT2) */

        sum3a=sum3b=0.0;
        y = (LT2-m)/(sqrt(v));
        if (y >= 0.0)
      {
       i1 = 0.0; i2 = y;
      }
        else
      {
       i1 = y; i2 =0.0;
      }
       sum3a = Suminf1(w,i1,i2,qct_24,f24,indc);
        y = (-LT2-m)/(sqrt(v));
        if (y >= 0.0)
      {
       i1 = 0.0; i2 = y;
      }
        else
      {
       i1 = y; i2 =0.0;
      }
        sum3b = Suminf2(w,i1,i2,qct_24,f24,indc);
       if ((sum3a < 0.0) || (sum3b < 0.0))
        {
         fprintf(f_outel,"In f2 sum3a=%lf sum3b=%lf\n",sum3a,sum3b);
         exit(1);
        }
       }
     fsum=sc+ac+0.5*oc*sysv*(d2*d2)+vr*oc*d2+sum1+sum2a+sum2b+sum3a+sum3b;
     return(fsum);
    }

/* function f3 - Equ3 which is not a constant for any particular iteration */

double f3(int IT, double z, double vr, int indc)
 {
   int w;
  double v,m,LT1=-1.0,LT2=-1.0,i1=0.0,i2=0.0,y=0.0,fsum=0.0,sum1,sum2a,sum2b,sum3a,sum3b,k1,d0;

    if (indc==cf)
     {
      if ((q_what==2) || (q_what==0))
       d0=fsi;
      else
      if (q_what==1)
       d0=d2;
      else
       {
       fprintf(f_outel,"q_what invalid in f3\n");
       exit(1);
      }
    }
    else
      d0=d1;
    k1 = (vr+d0*sysv)/(vr+d0*sysv+obsv);
    v = k1*(vr+d0*sysv); m = z;
     if ((v <= 0.0) || (vr <= 0.0))
      {
      fprintf(f_outel,"In f3 v=%3.15lf vr=%3.15lf indc=%d IT=%d\n",v,vr,indc,IT+1);
      exit(1);
      }
    if (IT==0)
     {
     if (d_last13 != -1.0)
       d0=d_last13;
     fsum = oc*z*z*d0+vr*oc*d0+0.5*oc*sysv*d0*d0+sc;
     return(fsum);
     }
     else
      {
       if (IT < vsi)
      w = IT-1;
       else
      w = vsi-1;
       if (indc==cf)
     {
      LT1=Retrieve(&s_info[w],(qct_13)*13+12,numquadpts);  LT2=LT1;
     }
       else
       if (indc==ncf)
    {
     LT1=Retrieve(&s_info[w],(qct_13)*13,numquadpts);
     LT2=Retrieve(&s_info[w],(qct_13)*13+1,numquadpts);
    }
       else
    {
     fprintf(f_outel,"no such indc in f3 \n");
    exit(1);
       }

/* Getting integral on (-LT1,LT1) */

      sum1=0.0;
      i1 = -LT1; i2 = LT1;
      sum1 = Sumnorm(w,i1,i2,v,m,qct_13,f13,indc);
    if (sum1 < 0.0)
    {
     fprintf(f_outel,"In f3 sum1=%lf\n",sum1);
     exit(1);
     }
```

```
/* ONLY when LT1 != LT2 is the integral in (LT1,LT2) and (-LT2,-LT1) calculated */

        if (LT1 == LT2)
       sum2a = sum2b = 0.0;
         else
         if (LT1 != LT2 && LT2 != -1.0)
      {
       i1=LT1; i2=LT2;
       sum2a = Sumnorm(w,i1,i2,v,m,qct_13,f13,indc);
       i1=-LT2; i2=-LT1;
       sum2b = Sumnorm(w,i1,i2,v,m,qct_13,f13,indc);
       if ((sum2a < 0.0) || (sum2b < 0.0))
        {
         fprintf(f_outel,"In f3 sum2a=%lf sum2b=%lf\n",sum2a,sum2b);
         exit(1);
        }
      }

/* Integral in (LT2,inf) and (-inf,-LT2) */

        sum3a=sum3b=0.0;
        y = (LT2-m)/(sqrt(v));
        if (y >= 0.0)
      {
       i1 = 0.0; i2 = y;
      }
        else
      {
       i1 = y; i2 =0.0;
      }
       sum3a = Suminf1(w,i1,i2,qct_13,f13,indc);
        y = (-LT2-m)/(sqrt(v));
        if (y >= 0.0)
      {
       i1 = 0.0; i2 = y;
      }
        else
      {
       i1 = y; i2 =0.0;
      }
     sum3b = Suminf2(w,i1,i2,qct_13,f13,indc);
       if ((sum3a < 0.0) || (sum3b < 0.0))
        {
         fprintf(f_outel,"In f3 sum3a=%lf sum3b=%4.20lf\n",sum3a,sum3b);
         exit(1);
        }
      }
    fsum=sc+oc*(z*z)*d0+0.5*oc*sysv*(d0*d0)+vr*oc*d0+sum1+sum2a+sum2b+sum3a+sum3b;
    return(fsum);
    }
```

```
/* function f4 - Equ3 which is not a constant for any particular iteration */

double f4(int IT, double z, double vr, int indc)
 {
  int w;
  double v, m, LT1=-1.0, LT2=-1.0, i1=0.0, i2=0.0, y=0.0;
  double fsum=0.0, sum1,sum2a,sum2b,sum3a,sum3b,k1,d00;

  k1 = (vr+d2*sysv)/(vr+d2*sysv+obsv); v = k1*(vr+d2*sysv); m = z;
   if ((v <= 0.0) || (vr <= 0.0))
    {
     fprintf(f_outel,"In f4 v=%3.15lf vr=%3.15lf indc=%d IT=%d\n",v,vr,indc,IT+1);
     exit(1);
    }
  if (indc==cf)
    {
     fprintf(f_outel,"indc can't be =cf in f4\n");
     exit(1);
    }
   if (IT == (vsi-1))
    {
     if (d_last24 != -1.0)
        d00=d_last24;
     else
        d00=d2;
      fsum = oc*z*z*d00+vr*oc*d00+0.5*oc*sysv*d00*d00+sc;
      return(fsum);
    }
   else
   if (IT >= vsi)
    {
     w=0;
     LT1=Retrieve(&s_info[w],(qct_24)*13+2,numquadpts);
     LT2=Retrieve(&s_info[w],(qct_24)*13+3,numquadpts);

/* Getting integral on (-LT1,LT1) */

     sum1=0.0;
     i1=-LT1; i2=LT1;
     sum1 = Sumnorm(w,i1,i2,v,m,qct_24,f24,indc);
       if (sum1 < 0.0)
       {
   fprintf(f_outel,"In f4 sum1=%lf\n",sum1);
        exit(1);
        }

/* ONLY when LT1 != LT2 is the integral in (LT1,LT2) and (-LT2,-LT1) calculated */

     if (LT1==LT2)
        sum2a = sum2b = 0.0;
     else
     if (LT1 != LT2 && LT2 != -1.0)
      {
       i1=LT1; i2=LT2;
       sum2a = Sumnorm(w,i1,i2,v,m,qct_24,f24,indc);
       i1=-LT2; i2=-LT1;
       sum2b = Sumnorm(w,i1,i2,v,m,qct_24,f24,indc);
              if ((sum2a < 0.0) || (sum2b < 0.0))
               {
            fprintf(f_outel,"In f4 sum2a=%lf sum2b=%lf\n",sum2a,sum2b);
               exit(1);
               }
      }

/* Integral in (LT2,inf) and (-inf,-LT2) */

    sum3a=sum3b=0.0;
    y = (LT2-m)/(sqrt(v));
    if (y >= 0.0)
     {
      i1 = 0.0; i2 = y;
     }
    else
     {
     i1 = y; i2 =0.0;
     }
     sum3a = Suminf1(w,i1,i2,qct_24,f24,indc);
    y = (-LT2-m)/(sqrt(v));
    if (y >= 0.0)
     {
      i1 = 0.0; i2 = y;
     }
    else
     {
     i1 = y; i2 =0.0;
     }
    sum3b = Suminf2(w,i1,i2,qct_24,f24,indc);
             if ((sum3a < 0.0) || (sum3b < 0.0))
              {
           fprintf(f_outel,"In f4 sum3a=%lf sum3b=%lf\n",sum3a,sum3b);
              exit(1);
              }
  }
  fsum = sc+oc*(z*z)*d2+0.5*oc*sysv*(d2*d2)+vr*oc*d2+sum1+sum2a+sum2b+sum3a+sum3b;
  return(fsum);
 }

/* function for normal density with mean m and variance v */

double Normden(double v, double m, double x)
{
  double invsq=0.0, normal=0.0;
  invsq = 1.0/sqrt(v);
  normal = constant*invsq*exp(-0.5*((x-m)*(x-m))/v);
  return(normal);
}

/* function Sumnorm - to find the sum1 and sum2 in the functions1,3 using var1 */

double Sumnorm(int w, double i1, double i2, double v, double m, int qvar, int flag, int indc)
{
  int i;
  double sum=0.0, fval=0.0, prob;

  for (i=0; i < numquadpts; ++i)
   {
    if (i2 == Abs(i1))
     {
      if ((flag==f13) && (indc==cf))
        fval=Retrieve(&s_info[w],qvar*13+10,i);
      else
      if ((flag==f13) && (indc==ncf))
        fval=Retrieve(&s_info[w],qvar*13,i);
      else
      if ((flag==f24) && (indc==ncf))
        fval=Retrieve(&s_info[w],qvar*13+5,i);
      else
        {
         fprintf(f_outel,"In Sumnorm no such (flag,indc) combo. exits flag=%d indc=%d\n",flag,indc);
         exit(1);
        }
     }
    else
    if ((i2 > 0.0) && (i1 > 0.0))
     {
      if (indc==cf)
       {
        fprintf(f_outel,"i1=%lf i2=%lf\n",i1,i2);
        fprintf(f_outel,"Error10: i2>0 & i1>0 no cf possible!\n");
        exit(1);
       }
      if (flag==f13)
        fval=Retrieve(&s_info[w],qvar*13+1,i);
      else
        fval=Retrieve(&s_info[w],qvar*13+6,i);
     }
    else
    if ((i2 < 0.0) && (i1 < 0.0))
     {
      if (indc==cf)
       {
        fprintf(f_outel,"Error11: i2<0 & i1<0 no cf possible!\n");
```

```
         exit(1);
        }
      if (flag==f13)
        fval=Retrieve(&s_info[w],qvar*13+2,i);
      else
        fval=Retrieve(&s_info[w],qvar*13+7,i);
     }
    else
    if (i2 != Abs(i1))
     {
      if (indc==cf)
       {
        fprintf(f_outel,"Error12: i2!= Abs(i1) no cf possible!\n");
        exit(1);
       }
     if (((i2 > 0.0) && (i1 > 0.0)) || ((i2 < 0.0) && (i1 < 0.0)))
      {
      fprintf(f_outel,"Error13: in Sumnorm in the limits i1=%lf and i2=%lf\n",i1,i2);
      exit(1);
      }
     }
    prob=QW(i1,i2,i)*Normden(v,m,Q(i1,i2,i));
    if ((fval < 0.0) || (prob < 0.0) || (prob > 1.0))
      {
       fprintf(f_outel,"1: prob=%lf QW=%lf Normden=%lf\n",prob,QW(i1,i2,i),Normden(v,m,Q(i1,i2,i)));
       fprintf(f_outel,"fval=%lf\n",fval);  fprintf(f_outel,"v=%lf m=%lf i1=%lf i2=%lf i=%d\n",v,m,i1,i2,i);
       exit(1);
      }
     sum += QW(i1,i2,i)*fval*Normden(v,m,Q(i1,i2,i));
   }
 return(sum);
}

/* function Suminf1 - to find the sum3a in the functions1,3 using var1- if flag=f13 then used in funcs. f1
and f3 and if flag=f24 then used in funcs. f2 and f4   */

double Suminf1(int w, double i1, double i2, int qvar, int flag, int indc)
{
  int i;
  double sum=0.0, fval1=-1.0, fval0=-1.0, prob=0.0, wt=-1.0, norm=-1.0;

  for (i=0; i < numquadpts; ++i)
  {
  if ((flag==f13) && (indc==cf))
   {
    if (i==0)
     fval0 = Retrieve(&s_info[w],qvar*13+11,0);
    else
     fval1 = Retrieve(&s_info[w],qvar*13+11,i);
   }
  else
  if ((flag==f13) && (indc==ncf))
   {
    if (i==0)
     fval0 = Retrieve(&s_info[w],qvar*13+3,0);
    else
     fval1 = Retrieve(&s_info[w],qvar*13+3,i);
   }
  else
  if ((flag==f24) && (indc==ncf))
   {
    if (i==0)
     fval0 = Retrieve(&s_info[w],qvar*13+8,0);
    else
     fval1 = Retrieve(&s_info[w],qvar*13+8,i);
   }
  else
   {
    fprintf(f_outel,"Error20:   In   Suminfl   combo   of   (flag,indc)   does   not   exist!   flag=%d
indc=%d\n",flag,indc);
    exit(1);
    }
 if (((i != 0) && (fval1 != fval0)) || (fval0 < 0.0))
  {
   fprintf(f_outel,"Error21: fval1=%lf fval0=%lf i=%d\n",fval1,fval0,i);
   exit(1);
  }
}
 if ((i1 == 0.0) && (i2 == 0.0))
  {
   sum = 0.5*fval0;
   return(sum);
  }
 else
 if ((i1 == 0.0) && (i2 > 0.0))
  {
   if(i2 > 8.0)
    return(0.0);
   else
    {
    for (i=0; i < numquadpts; ++i)
    {
    wt = QW(i1,i2,i); norm = Normden(1.0,0.0,Q(i1,i2,i)); prob += wt*norm;
    if ((wt < 0.0) || (norm < 0.0) || (prob > 0.5))
     {
      fprintf(f_outel,"Error22: In Suminfl i1=%lf i2=%lf i=%d\n",i1,i2,i);
      fprintf(f_outel,"fval=%lf\n",fval0);
      fprintf(f_outel,"prob=%lf QW=%lf Normden=%lf\n",prob,wt,norm);
      exit(1);
     }
    }
    sum = (0.5 - prob)*fval0;
```

```
      return(sum);
     }
    }
   else
   if ((i2 == 0.0) && (i1 < 0.0))
    {
     if(Abs(i1) > 8.0)
      return(fval0);
     else
      {
      for (i=0; i < numquadpts; ++i)
      {
       wt = QW(i1,i2,i); norm = Normden(1.0,0.0,Q(i1,i2,i)); prob += wt*norm;
       if ((wt < 0.0) || (norm < 0.0) || (prob > 0.5))
        {
         fprintf(f_outel,"Error23: In Suminf1 i1=%lf i2=%lf i=%d\n",i1,i2,i);
         fprintf(f_outel,"fval=%lf\n",fval0);
         fprintf(f_outel,"prob=%lf QW=%lf Normden=%lf\n",prob,wt,norm);
         exit(1);
        }
      }
       sum = (0.5 + prob)*fval0;
       return(sum);
      }
    }
   else
    {
     fprintf(f_outel,"Something wrong in Suminf1\n");
     return(-100000000.0);
    }
}
```

/* function Suminf2 - to find the sum3b in the functions1,3 using var1-if flag=f13 then used in funcs. f1 and f3 and if flag=f24 then used in funcs. f2 and f4 */

```
double Suminf2(int w, double i1, double i2, int qvar, int flag, int indc)
{
 int i;
 double sum=0.0, fval0=-1.0, fval1=-1.0, prob=0.0, wt=-1.0, norm=-1.0;

 for (i=0; i < numquadpts; ++i)
  {
   if ((flag==f13) && (indc==cf))
    {
     if (i==0)
      fval0 = Retrieve(&s_info[w],qvar*13+12,0);
    else
      fval1 = Retrieve(&s_info[w],qvar*13+12,i);
    }
 else
 if ((flag==f13) && (indc==ncf))
```

```
     {
      if (i==0)
       fval0 = Retrieve(&s_info[w],qvar*13+4,0);
     else
       fval1 = Retrieve(&s_info[w],qvar*13+4,i);
     }
  else
  if ((flag==f24) && (indc==ncf))
     {
      if (i==0)
       fval0 = Retrieve(&s_info[w],qvar*13+9,0);
     else
       fval1 = Retrieve(&s_info[w],qvar*13+9,i);
     }
  else
   {
    fprintf(f_outel,"Error30:In Suminf2 combo of (flag,indc) doesn't exist! flag=%d indc=%d\n",flag,indc);
    exit(1);
   }
 if (((i != 0) && (fval1 != fval0)) || (fval0 < 0.0))
  {
   fprintf(f_outel,"Error31: fval1=%lf fval0=%lf i=%d\n",fval1,fval0,i);
   exit(1);
  }
}
 if ((i1 == 0.0) && (i2 == 0.0))
  {
   sum = 0.5*fval0;
   return(sum);
  }
 else
 if ((i1 == 0.0) && (i2 > 0.0))
  {
   if(i2 > 8.0)
     return(fval0);
   else
    {
     for (i=0; i < numquadpts; ++i)
     {
     wt=QW(i1,i2,i); norm=Normden(1.0,0.0,Q(i1,i2,i)); prob += wt*norm;
     if ((wt < 0.0) || (norm < 0.0) || (prob > 0.5))
      {
       fprintf(f_outel,"Error32: In Suminf2 i1=%lf i2=%lf i=%d\n",i1,i2,i);
       fprintf(f_outel,"fval=%lf\n",fval0);
       fprintf(f_outel,"prob=%lf QW=%lf Normden=%lf\n",prob,wt,norm);
       exit(1);
      }
     }
     sum = (0.5 + prob)*fval0;
     return(sum);
    }
   }
  else
  if ((i2 == 0.0) && (i1 < 0.0))
   {
   if(Abs(i1) > 8.0)
     return(0.0);
   else
    {
     for (i=0; i < numquadpts; ++i)
     {
     wt=QW(i1,i2,i); norm=Normden(1.0,0.0,Q(i1,i2,i)); prob += wt*norm;
     if ((wt < 0.0) || (norm < 0.0) || (prob > 0.5))
      {
       fprintf(f_outel,"Error33: In Suminf2 i1=%lf i2=%lf i=%d\n",i1,i2,i);
       fprintf(f_outel,"fval=%lf\n",fval0);
       fprintf(f_outel,"prob=%lf QW=%lf Normden=%lf\n",prob,wt,norm);
       exit(1);
      }
     }
    sum = (0.5 - prob)*fval0;
    return(sum);
    }
  }
  else
  {
   fprintf(f_outel,"Something wrong in Suminf2\n");
   return(-100000000.0);
  }
}

/* function to find the pt.of intersection for any 2 funcs. */

double Find_Inter(double (*g1)(int,double,double,int), double (*g2)(int,double,double,int), double x1,
double xn, double step,int IT,double h, int indc)
{
 int sign,oldsign;
 double diff=0.0,i;

 for (i=x1; i >= (xn-step); i -= step)
  {
   diff = (g1)(IT,i,h,indc) - (g2)(IT,i,h,indc);
   if (Abs(diff) < TOLERANCE)
     return(i);
   sign = ((diff < 0.0) ? -1 : 1);
   if (i == x1)
     oldsign = sign;
   if ((i==xn) && (oldsign==sign))
     return(-1.0);
   if (sign != oldsign)
     break;
  }
  if (i < (xn-step))
   {
    fprintf(f_outel,"Error in Find_Inter!\n");
    fprintf(f_outel,"h=%lf i=%3.20lf indc=%d IT=%d\n", h, i, indc, IT+1);
    fprintf(f_outel,"xn=%lf step=%lf xn+step=%3.20lf\n", xn, step, xn+step);
    exit(0);
    return(-1.0);
   }
 return(Find_Inter(g1,g2,i+step,i,step/H,IT,h,indc));
}

/*  function - to find the 4 pts. of intersection */

int Four_Inter(int IT, double *lt1, double *lt2, double var,int indc)
{
 double xval1=-1.0, xval2=-1.0, xval3=-1.0, max12, f1_val, f2_val, f3_val, f4_val;
 double (*minconstf)(int, double,double,int)=&F_err;

  f1_val=f1(IT,0.0,var,indc);  f2_val=f2(IT,0.0,var,indc);
  f3_val=f3(IT,0.0,var,indc);  f4_val=f4(IT,0.0,var,indc);
  if (f1_val < f2_val)
    minconstf = funcs[0];
  else
  if (f1_val > f2_val)
    minconstf = funcs[1];
  else
   {
    fprintf(f_outel,"constf1=constf2\n");
    exit(1);
   }
  if (max(f3_val,f4_val) < mum(f1_val,f2_val))
   {
    xval1=Find_Inter(funcs[2],minconstf,H,0.0,1.0,IT,var,indc);
    xval2=Find_Inter(funcs[3],minconstf,H,0.0,1.0,IT,var,indc);
    xval3=Find_Inter(funcs[2],funcs[3],H,0.0,1.0,IT,var,indc);
    max12=max(xval1,xval2);
    if (xval3 < 0.0)
    {
     *lt1=*lt2=max12;
     return(10);
    }
   if (xval3 > max12)
    {
     *lt1=*lt2=max12;
     return(10);
    }
   if (xval3 < mum(xval1,xval2))
    {
     *lt1=xval3; *lt2=max12;
     return(10);
    }
   }
  if (f3_val > mum(f1_val,f2_val))
   {
    xval2=Find_Inter(funcs[3],minconstf,H,0.0,1.0,IT,var,indc);
    *lt1=*lt2=xval2;
    return(10);
   }
  if (f4_val > mum(f1_val,f2_val))
   {
    xval1=Find_Inter(funcs[2],minconstf,H,0.0,1.0,IT,var,indc);
    *lt1=*lt2=xval1;
    return(10);
   }
   fprintf(f_outel,"ERROR: No such case in Four_Inter\n");
   exit(1);
return(-100);
}

/* sort function that sorts array ra and makes the corresponding rearrangement to array rb */

void sort2(int n,double ra[],double rb[])
{
  int l,j,ir,i;
  double rrb,rra;

  l=(n >> 1)+1;  ir=n-1;
  for (;;)
   {
    if (l > 0)
     {
      rra=ra[--l]; rrb=rb[l];
     }
    else
     {
      rra=ra[ir];   rrb=rb[ir];
      ra[ir]=ra[0]; rb[ir]=rb[0];
      if (--ir == 0)
       {
        ra[0]=rra;  rb[0]=rrb;
        return;
       }
     }
   i=l;  j=(l << 1)+1;
   while (j <= ir)
    {
     if (j < ir && ra[j] < ra[j+1]) ++j;
      if (rra < ra[j])
       {
        ra[i]=ra[j];  rb[i]=rb[j];
        j += (i=j);
       }
      else j=ir+1;
    }
   ra[i]=rra; rb[i]=rrb;
  }
}
/* to get the total exp. loss for Crowder's case of sc=0 */

double Tot_Loss(int itr, double lim1, double lim2, double wvr, double qvr, int indc)
{
  int i;
  double tot_el1=0.0, tot_el2=0.0, tot_el3=0.0, total=0.0, xpt, i1, i2;
  double (*mf1)(int,double,double,int)=&F_err, (*mf2)(int,double,double,int)=&F_err;
  double (*mf3)(int,double,double,int)=&F_err;

   if ((indc==cf) && (lim1 != lim2))
    {
     fprintf(f_outel,"In Tot_Loss indc=cf, lim1 != lim2 ERROR\n");
     fprintf(f_outel,"indc=%d and lim1=%lf lim2=%lf\n",indc,lim1,lim2);
     exit(1);
    }
   mf1=minfunc(itr,0.0,qvr,indc);
   if (lim1 != lim2)
     mf2=minfunc(itr,(lim1+lim2)*0.5,qvr,indc);
   mf3=minfunc(itr,lim2+1.0,qvr,indc);
   i1=-lim1/sqrt(wvr); i2=lim1/sqrt(wvr);
   for (i=0; i < numquadpts; ++i)
    {
     xpt=Q(i1,i2,i)*sqrt(wvr);
     tot_el1 = tot_el1+(mf1)(itr,xpt,qvr,indc)*QW(i1,i2,i)*Normden(1.0,0.0,Q(i1,i2,i));
    }
   if (lim1 == lim2)
    tot_el2=0.0;
   else
    {
     i1=lim1/sqrt(wvr); i2=lim2/sqrt(wvr);
     for (i=0; i < numquadpts; ++i)
      {
       xpt=Q(i1,i2,i)*sqrt(wvr);
       tot_el2 = tot_el2+(mf2)(itr,xpt,qvr,indc)*QW(i1,i2,i)*Normden(1.0,0.0,Q(i1,i2,i));
      }
    }
     tot_el2=tot_el2*2.0;
   i1=0.0; i2=lim2/sqrt(wvr);
   if (i2 > 8.0)
    tot_el3=0.5;
   else
    {
     for (i=0; i < numquadpts; ++i)
       tot_el3 = tot_el3 + QW(i1,i2,i)*Normden(1.0,0.0,Q(i1,i2,i));
    }
   tot_el3 = (0.5-tot_el3)*(mf3)(itr,0.0,qvr,indc)*2.0;
```

```
    total=tot_el1+tot_el2+tot_el3;
  return(total);
 }

/* to get the intersection for Crowder's case of sc=0 */

void Inter_Crow(double qv, int iter_n)
{
  int i;
  double ptx, pty, lim=-1.0, i1, i2;
  double (*minfc1)(int,double,double,int)=&F_err;
  double (*minfc3)(int,double,double,int)=&F_err;
  char CR;

  funcs[0]=f1; funcs[2]=f3; funcs[1]=funcs[3]=NULL;
  lim=Find_Inter(f1,f3,H,0.0,1.0,IT,qv,cf);
  Store(&s_info[iter_n],(qct_13)*13+12,numquadpts,lim);
  minfc1=minfunc(IT,0.0,qv,cf);
  i1=-lim; i2=lim;
  for (i=0; i < numquadpts; ++i)
      Store(&s_info[iter_n],(qct_13)*13+10,i,(minfc1)(IT,Q(i1,i2,i),qv,cf));

  minfc3=minfunc(IT,lim+1.0,qv,cf);
  i1=0.0; i2=lim;
  for (i=0; i < numquadpts; ++i)
    Store(&s_info[iter_n],(qct_13)*13+11,i,(minfc3)(IT,Q(i1,i2,i),qv,cf));
  i1=-lim; i2=0.0;
  for (i=0; i < numquadpts; ++i)
    Store(&s_info[iter_n],(qct_13)*13+12,i,(minfc3)(IT,Q(i1,i2,i),qv,cf));
}
```

Vita

Shanthi Sethuraman was born on Novenber 15, 1963 in Madras, India. She attended Stella Maris College, India, receiving a Bachelor of Science Degree in Mathematics in 1984. In pursuit of further education in the United States, she attended Virginia Polytechnic Institute and State University, receiving a Master of Science Degree in Mathematics in 1989, a Master of Science Degree in Statistics in 1991 and a Doctor of Philosophy Degree in Statistics in 1995.

Shanthi Sethuraman is married to Mark R. Olin and she will begin her career as a Senior Statistician in the Statistical and Mathematical Sciences Division at Eli Lilly and Co., Tippecanoe Laboratories at Lafayette, Indiana.

Shanthi Sethuraman...