A SEQUENTIAL DECISION METHOD FOR HOMOGENEOUS BATCH DISPOSITION IN THE PRESENCE OF MEASUREMENT ERROR

By

RAEANNE LAMBERT

Bachelor of Science
The University of Tulsa
Tulsa, Oklahoma
1982

Master of Business Administration
Eastern New Mexico University
Portales, New Mexico
1985

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

OKLAHOMA STATE UNIVERSITY

A SEQUENTIAL DECISION METHOD FOR

HOMOGENEOUS BATCH DISPOSITION

IN THE PRESENCE OF

MEASUREMENT

ERROR

Thesis Approved:

Kenneth E Case
Thesis Adviser

Michael H Branson

Manjunath Kamath

W M Nanty

David B. Pratt

Thomas C. Collins
Dean of the Graduate College

PREFACE

The problem addressed in this research paper involves the impact of measurement error on the disposition of a homogeneous batch of product from a continuous process. Disposition is based on a single variable characteristic relative to a single specification limit. A newly-developed sequential sampling procedure is proposed which minimizes the expected total cost of sampling for a maximum number of observations. Motivation for this research is provided by industry situations in which a single observation of a quality characteristic is utilized to determine conformance of a homogenous batch of product. This topic has not previously been explored in the quality control literature.

Sequential sampling models and procedures are developed based on, alternatively, statistical and economic principles. The proposed economic model provides the optimal sequential sampling plan as determined by the minimum expected total cost. A comprehensive computer program is presented which implements both the statistical and economic models. The alternative approaches are compared on the basis of expected costs through the use of a computer simulator.

I would like to express my sincere appreciation to my major advisor, Dr. Kenneth E. Case, for his assistance, encouragement and support throughout my association with Oklahoma

State University. He is an outstanding educator and advisor whose guidance and influence is reflected in this research writing. Additionally, Dr. Case has earned my respect as a person, providing humor and insight when my rose-colored glasses fogged over. I would also like to express my gratitude to Dr. David Pratt, Dr. Manjunath Kamath, Dr. Michael Branson and Dr. Bill Warde for their advice and encouragement throughout my graduate study and research. It has been a pleasure and privilege to associate with such an outstanding doctoral committee.

I was (relatively) young and single when I arrived in Stillwater, but now I find myself surrounded with moral support. I thank my husband, Mike Thornbrugh, for his encouragement and understanding as I pursued my goals. My son, Adam, and daughter, Emily, have yet to learn of the sacrifices they have made. My mother, Joanne Todd, has been an inspiration throughout my educational endeavors; I thank her for her support and assistance when the deadlines closed-in.

Special thanks are extended to my in-laws, Paul and Freda Thornbrugh, for the encouragement and many hours of baby-sitting which I can never repay.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

FIGURE PAGE

CHAPTER 1

THE RESEARCH PROBLEM

1.1 INTRODUCTION

Overview

The subject of this research paper is the development of a decision method for disposition of a single homogeneous batch from a continuous process by variables characteristics in the presence of measurement error. The decision method dictates acceptance, rejection or re-measurement of the batch; disposition is based on a cumulative sequence of measurements. The study considers consequences and costs of measurement and conformance misclassification as decision criteria. A comparison of economic criteria with statistical criteria is performed. Cases of known and unknown measurement error distribution variance are treated, as well as measurement systems with and without bias. Practical implementation of the decision method is accomplished through a comprehensive computer program, relieving the operator of complex mathematical calculations and decisions. Effectiveness of the proposed system is demonstrated through simulation.

General Discussion

Measurement system error has received widespread attention in consideration of impact on lot-by-lot acceptance sampling and control charting. However, literature is scarce with regard to quality dispositions which are based on a single dimensional measurement which is subject to inherent measurement system errors. Such single-measurement dispositions are performed frequently in industrial situations of batch inspection of a continuous process. Single batches are viewed independently of others, and decisions are made based on a single measurement iteration without regard to inherent inspection system errors.

Errors of measurement are unfamiliar to, and largely ignored by, the majority of shop-floor personnel, yet misclassification as a result of such errors involves real costs to the organization and its customers. Previous research has focused on compensating adjustment of specification limits to provide a cushion for measurement uncertainty and associated misclassification risks. However, establishment of such statistical decision limits creates inconsistencies among design documents (reflecting functional tolerances) and quality criteria documents (showing decision limits).

Non-formalized decision procedures which exist in industry utilize arbitrary decision limits (not specification limits) in conjunction with unsophisticated decision criteria dictating final disposition or re-measurement. As a fundamental example, an unsatisfactory initial reading may simply prompt re-measurement of the characteristic until the desired result is obtained. More formalized (yet still, arbitrary) systems may call for such criteria as a majority of five measurements, two out of three, etc. Such systems have received little attention in the literature. If the ultimate decision outcome dictates acceptance, no consideration is given to previously-observed measurements which fell beyond tolerance bounds.

This research utilizes functional tolerances in the development of statistical and economic decision criteria for homogeneous batch disposition. Existing literature on measurement error, variables sampling in the presence of measurement uncertainty, sequential analysis and quality cost modeling provides a firm base for approaching the problem and developing a sound, practical operations tool.

1.2 STATEMENT OF THE PROBLEM

Introduction

As presented in the previous section, this research effort focuses on quality dispositions which are based on a single measurement of a variable characteristic. Single measurement dispositions are common in bulk release of product from continuous processes. Neglecting to account for measurement system errors in such an inspection configuration may lead to inappropriate disposition and unnecessary quality costs incurred by the producer. No decision methodology has been developed which specifically addresses the problem of single batch disposition based on a variable characteristic subject to measurement error. Economic modeling of this type of quality disposition has not been treated.

The problem is illustrated in Figure 1.1. Two batches are shown, with the measurement error distribution (normal, no bias) centered on the actual, unknown value of the variable characteristic being measured. Although the true value of the characteristic for each batch lies below the upper specification shown, the tail area of the measurement error distribution around Batch B falling above the tolerance is, obviously, greater than that of Batch A. On a single measurement iteration, the probability of observing an out-of-tolerance reading is greater for Batch B than for Batch A. In any case, a non-conforming reading and subsequent rejection of the batch for either A or B would be erroneous.

The following discussion highlights areas of quality control and sampling inspection theory which are pertinent to the research problem.

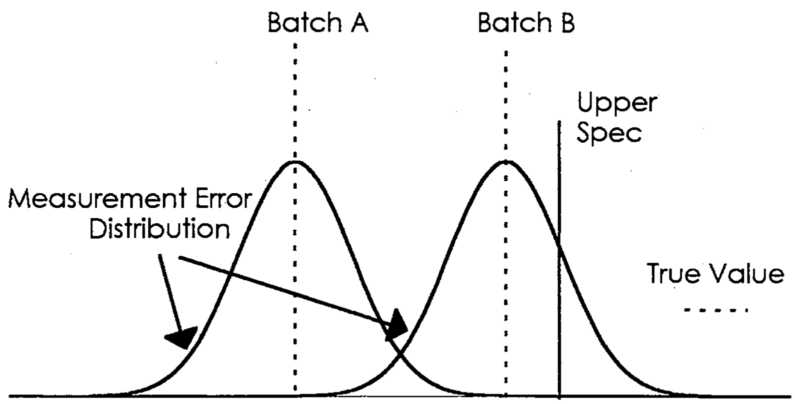

Figure 1.1 Illustration of the Problem.

Measurement System Error

Measurement system error is inherent in all industrial quality control/assurance operations. In the case of variable characteristics, the true measurement value is often confounded by human error and/or instrument test error [37]. This measurement error may be characterized in terms of bias and imprecision. Bias is the difference between the true value of a product characteristic and the average of a series of repeated measures on that characteristic using a fixed inspection system (same gage, operator, etc.). Bias appears as a fixed displacement from the true value, either in the positive (observed greater than true) or negative (observed less than true) direction. Imprecision is the inability to repeat observed results of measuring a characteristic using a fixed inspection system in a series of measurements. Imprecision can be expressed as the standard deviation of the measurements, giving a picture of the measurement dispersion around the averaged measured value of the characteristic.

In attribute inspections, variability between inspectors contributes to inconsistencies in part quality assessment. Additionally, variation is evidenced in judgments made by the same inspector on similar parts at different times and under varying inspection conditions.

Such inconsistencies are often dealt with by introducing visual standards for non-conformities to reduce subjectivity of the inspection process.

In the case of a variable part characteristic inspection, a specific characteristic is physically measured with a mechanical gage; the resultant gage reading is then compared to dimensional tolerances specified on part drawings or in quality assurance documents. The introduction of standard measurement tools removes much of the subjectivity from the process. However, measurement system error remains, as evidenced by an inability to reproduce gage readings in successive measurements, for various operators, different measurement tools, etc. The physical characteristic, measurement gage, inspector, physical environment and inspection technique all contribute to the numerical value assigned to the batch/part which serves to characterize the product. The true value of the characteristic is unknown and unknowable, yet, tolerances must be met in order to ensure a functional end-product.

For variable measurements, inspection system error can be characterized through gage repeatability and reproducibility studies. Such studies involve repeat measurements on a single product for various inspectors (no change in physical characteristic). Study results are analyzed through Analysis of Variance techniques to discover the sources of variation in the measurement process. When the measurement system is in control, variation is due only to random errors and the measurement distribution approximates a normal curve.

Acceptance Sampling

In lot acceptance sampling schemes, a group (lot) of parts is subjected to a cumulative decision criterion. The observations obtained from a random sample of the lot provide an estimate of the population parameters. By characterizing the lot through sampling, judgments can be made concerning the conformity of the lot to dimensional specifications and percent defective requirements.

Variables sampling plans may be one-sided or two-sided with respect to specification limits. One-sided plans are exact in the treatment of single specification limits. One-sided plans may also be used for double specification limits, but effectively treat each limit separately in estimating the proportion of product which is non-conforming. Two-sided plans address both tail areas jointly in judging conformance to the predetermined acceptable level of proportion non-conforming.

The most important consideration, as identified by Schilling [50] in applying variables sampling plans is the requirement that the underlying distribution is known and stable. Well-known variables sampling plans, such as MIL-STD-414, assume normality of the population.

When the standard deviation is known, an estimate of the population proportion non-conforming is straightforward using the normal distribution. The case of unknown standard deviation may be solved using a normal approximation, as suggested by Wallis [60]; exact solution involves the non-central t-distribution. In sampling from a normal population, the t statistic, $t = \frac{\sqrt{N}(U - \overline{X})}{s}$, follows a Student's t-distribution only for 50% non-conforming. In general (all values of proportion non-conforming), t follows the non-central t-distribution.

MIL-STD-414, Sampling Procedures and Tables for Inspection by Variables for Percent Defective [38], presents two methods for treating the case of unknown standard deviation. The "s-method" utilizes the sample standard deviation as an estimate of the population parameter. Alternatively, a procedure is provided which estimates the standard deviation from a sample value for the average range ("R-method"). However, both methods are based on the non-central t-distribution in estimating the proportion of product non-conforming.

In consideration of inherent inspection system error, lot-by-lot acceptance sampling plans may be adjusted in terms of acceptance criteria. Such compensating adjustments have been treated in the literature. To allow for variation in measurements, percentage non-conforming limits are moved to ensure that lot quality requirements are not violated. This, effectively, tightens the quality requirements for the lot. Obviously, such plan compensations motivate attempts to reduce inspection error.

Although the relevance of variables acceptance sampling to the problem being addressed may not be readily apparent, the two topics are closely related. Just as sampling inspection attempts inferences about a lot (population), so does single characteristic measurement support inferences about the unknown true value of the characteristic. The assumption of normality in variables sampling plans (such as MIL-STD-414) is comparable to an assumption of normally distributed measurement imprecision in the problem at hand.

Single Measurement Disposition

Single batch inspections have received little attention in the literature, yet are prevalent in practical application. In a broad sense, even lot-by-lot acceptance inspections by variables involve a series of single part inspections in which each part is numerically characterized based on a single measurement iteration. Some techniques developed to address measurement error associated with lot-by-lot plans do address the use of multiple measurement of parts (primarily to estimate the measurement system variance).

However, single iteration inspections are not always associated with lot acceptance plans. Bulk release of a quantity of product from a continuous process is often based on a single measurement of a quality characteristic. In single trial inspection, a single characteristic is inspected independently of all other products and characteristics for the purpose of classification against dimensional inspection criteria.

It is important to note that, in a single measurement iteration, the true value of a measured characteristic is impossible to separate from measurement system error [23]. In recognition of measurement error, non-formalized decision systems (not, necessarily, firmly based on statistical and/or economic criteria) have emerged in practice. Such systems frequently call for repeated measurement iterations to verify or discount original observations. Upon observing an undesirable outcome on the first measurement, the inspector may simply measure again (perhaps, more carefully) to verify the previous result. If the second outcome is desirable, yet contradicts the first, very often it alone will dictate part disposition. More dramatically, a characteristic may be measured and re-measured until the desired outcome is achieved and all prior observations are discarded. Clearly, such procedures do not properly address the problem nor provide an objective system for batch or part classification.

Multiple Measurement Sampling

Repeating measurements on a characteristic provides a better estimate of the true value of the quality characteristic than a single measurement iteration. Through multiple measurements, it is possible to approximately distinguish the true value from the inherent measurement error. Indeed, Gage Repeatability and Reproducibility studies which serve to characterize the measurement error distribution are based on multiple part measurements.

If information is available on the parameters of the measurement error distribution, it is possible to specify a fixed iteration number which will adequately compensate for measurement imprecision. The average of the multiple measurements then serves as the estimator of the true value of the characteristic used for product disposition. However, a fixed iteration number may impose unnecessary inspection costs; all product, regardless of its "goodness" relative to the quality specification of interest, is subjected to the same number of measurements (hence, the same inspection costs). If measurement iterations are costly and/or time-consuming, the inspection "overkill" incurred by product lying well-within specifications can represent a substantial expense to the producer.

An alternative to multiple-iteration (fixed) measurement is sequential sampling. The literature on sequential techniques is broad and deep. In sequential sampling, observations are made only until enough evidence exists to decide in favor of the specified null or alternative hypothesis. Such systems have been successfully applied to lot-by-lot acceptance sampling and parameter estimation. However, sequential statistical principles have not been applied to the problem of measuring a single variable characteristic in the presence of measurement error. Predictably, practical implementation of sequential techniques is more complicated than its fixed-iteration counterpart. All available data is examined following each iteration and subjected to the decision criteria.

The difficulty of designing a sequential plan for measurement of a single variable characteristic depends heavily on prior knowledge of the measurement error distribution. Under the assumption of a normal error distribution function, a sample of observations taken from a single part will represent the non-central t-distribution. If a variance estimator is available (from a repeatability and reproducibility study) the problem represents an application of a likelihood ratio test for simple hypotheses. A lack of knowledge about the variance makes the problem one of composite hypotheses and complicates the application of a likelihood ratio test [59]. On the shop-floor, such sequential plans are best-implemented through the use of a computer program.

Quality Costs

Any incorrect decision in a quality inspection carries an associated quality cost. The cost of misclassifying a conforming product as unacceptable may simply be the cost of scrapping or reworking the batch. Alternatively, non-conforming product, characterized as acceptable, travels further through the process and gains value. If the misclassification is discovered before leaving the producer, the ultimate scrapping of the product carries a larger cost than it would have had it occurred earlier in the production process. If the non-conforming batch leaves the producer and is ultimately discovered by the customer, the potential for additional (often prohibitive) costs is varied. These costs are often difficult to quantify, but may represent the greatest potential costs to the producer. A non-conforming product which reaches the customer may perpetuate costs due to return, replacement, warranty, recall, injury, loss of life, etc.

There are also real costs associated with the physical inspection system. Fixed costs of inspection include paperwork, product handling and inspection setup. In a multiple-iteration inspection scheme (fixed or sequential), there are also costs associated with each measurement iteration, such as labor and gage depreciation.

Product disposition following inspection may involve fixed and variable costs, depending on the plant layout and disposition procedures. Material handling costs may be quantity-dependent. Paper-handling costs may be dependent on the material disposition (additional documentation may be required for a rejected product).

Cost models continue to appear in the literature treating lot-by-lot acceptance sampling, both by attributes and variables [6, 7, 8, 9, 11, 37, 51, 52]. Measurement system error has been explored in this area [6, 11, 17, 37, 58]. However, the case of sequential single product inspection, the focus of this research problem, has not been addressed.

Problem Summary

Given a single, independent variable batch characteristic with costs of associated classification/disposition, the problem becomes one of improving the disposition decision method. By integrating the principles of sequential sampling with the economics of quality costing, a method can be developed for product disposition based on a single variable characteristic. Multiple measurements improve the estimate of the true dimension; by sequentially analyzing observations, excess sampling (hence, cost) is not incurred.

This research paper addresses the problem of dispositioning a batch of product from a continuous process in the presence of measurement system error. Conditions of known and unknown measurement error variance are considered. The research addresses the problem from, alternatively, economic and statistical standpoints.

1.3 OBJECTIVES OF THE RESEARCH

The outcome of the research is a comprehensive decision system to ensure statistically and economically correct batch disposition based on a single variable characteristic in the presence of measurement error. Models are developed and presented for both statistical and economically optimal sequential sampling procedures. Practical implementation of the sampling theory is facilitated through a comprehensive computer program.

Consideration is given to:

i) Measurement variation with and without bias.

ii) Known and unknown measurement system variance.

iii) Economic decisions versus statistical decisions.

Research efforts are based on the following assumptions:

1) The measurement error is normally distributed.
2) Disposition is based on a single variable quality characteristic.
3) The quality characteristic is judged relative to a single specification limit, beyond which the product is considered non-conforming.
4) Dimensional Tolerances are given; all assessments are made based on specified tolerances.
5) Decision risks (α and β) are given (statistical case).
6) Quality costs of misclassification are known.
7) The batch is homogeneous with respect to the characteristic being measured.

The research effort may be logically subdivided into the following sub-objectives:

1) Known Measurement Variance Sequential Model (Statistical)

    a) Without bias

    b) With bias

Given the measurement distribution parameters, tolerance limit and acceptable risk levels, the model accepts observed measurements, sequentially, and recommends acceptance, rejection or re-measurement of the characteristic. The basis of this sub-section is the Wald [59] Sequential Probability Ratio Test (SPRT) for simple hypotheses.

2) Unknown Measurement Variance Sequential Model (Statistical)

   a) Without bias

   b) With bias

Given the measurement distribution mean (bias), tolerance limit and acceptable risk levels, the model accepts observed measurements, sequentially, and recommends acceptance, rejection or re-measurement of the characteristic. The methodology follows that of Rushton [49] in treating tests of composite hypotheses using likelihood ratios (ratio of two non-central t-distributions). Because the measurement error variance is unknown and must be estimated with sample data, no decisions are recommended prior to the second sample measurement.

3) Develop Cost Model

A cost model appropriate to batch measurement of a characteristic from a continuous process, subject to measurement error, is developed. For total cost modeling, a prior normal distribution of batch measurements is assumed. Costs included are measurement iteration costs, costs incurred due to acceptance of a non-conforming batch and costs associated with rejection of a batch which is acceptable. The case of known measurement system variance is treated.

The sequential economic model is based on decision cutoff values which are iteration-specific and are compared with the average of the sequential observations. The cost model is also a function of the maximum number of allowable observations taken for batch disposition.

4) Optimize Cost Model

Optimization of the cost model developed in item (3) integrates prior knowledge of the process distribution in minimizing the expected total cost of the sequential sampling plan. The expected cost associated with an additional measurement iteration involves subsequent costs of acceptance, rejection, and re-measurement in (possible) future iterations, up to and including n_{max} (the maximum number of observations allowed).

5) Computer Modeling of Decision Theory

The decision methods as specified in items (1)-(4) are coded in the FORTRAN computer language in a comprehensive program. The user of the program is presented the option of utilizing the statistical procedure or optimizing the sequential plan based on cost parameters. A module is also provided which accepts user-specified economic decision cutoff values and estimates the expected cost of the plan.

The statistical module prompts for and accepts required parameters of the inspection plan (specification, risk level, costs, etc.) and the observation sequence commences. At each sequential iteration, the program prompts the user for the observed measurement, and recommends either acceptance, rejection or additional measurement. Truncation of the sequential sampling plan is performed at the n_{max} value which is specified by the user.

The economic optimization module prompts for and accepts parameters of the inspection plan, including sampling costs. The optimal decision cutoff values are presented as output. Values of n_{max} in the range 1 to 3 are allowed by the computer program.

6) Comparison of Models Through Simulation

A computer simulator is written in FORTRAN for comparing the sequential and economic models. Various runs are conducted using data which simulates inspection data obtained in the presence of measurement error. Product dispositions dictated by the decision methods for the simulated data are compared to the desired (correct) result.

The statistical decisions which are dictated in sections (1) and (2) are evaluated based on the total cost of the sampling plans which is developed. The costs specified as input to the optimal economic plan are used to assess the performance of the statistical plans.

1.4 SUMMARY AND DOCUMENT ORGANIZATION

The research addresses the problem of homogeneous batch disposition based on a single variable characteristic relative to a specification limit in the presence of measurement error. Product disposition is often based on a single measurement observation. Sequential plans are proposed which implement statistical and economic theory in improving the disposition process. The optimal economic model provides the minimal expected total cost of sampling, and is implemented for maximum observation values of 1 through 3 through a comprehensive computer program; the program also allows product inspection using the statistical theory which is presented. The upper limit of three observations is widely applicable in industrial situations which historically allow only a single measurement observation.

Chapter 1 presents the research problem and a discussion of the objectives of the research. Existing literature which relates to the research problem is highlighted in Chapter 2. Pertinent topics which are presented are: 1) Measurement Bias and Imprecision, 2) Variables Sampling and the Non-Central t-Distribution, 3) Economic Analysis of Variables Acceptance Schemes, and 4) Sequential Analysis.

The statistical theory and proposed solution to the problem are developed in Chapter 3, including the known and unknown variance cases. The statistical cases utilize sequential probability ratio theory in model development. Chapter 3 also contains FORTRAN program subroutine summaries. A user guide to program operation for the statistical solutions are given in Chapter 4.

Development of the economic model for the known variance case is presented in Chapter 5. The logic behind the economic computer program operation is presented, as are summaries of the pertinent subroutines utilized in the FORTRAN program. Chapter 6 contains the user guide to the economic computer program operation.

Results of the research, including comparison of the optimal economic model with the statistical approach, are given in Chapter 7. The computer simulation program used for model comparisons is also presented in this chapter. Chapter 8 gives a summary of research findings and contributions.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

The following chapter contains a formal discussion of the literature related to variables inspection in the presence of measurement error. The review is sub-divided into four primary areas:

Measurement Bias and Imprecision
Variables Acceptance Sampling and Applications of the Non-Central t-Distribution
Sequential Analysis
Economic Analysis of Variables Acceptance Schemes

Each of the above topics has been covered extensively in available literature. An attempt is made in this chapter to present only those works which have relevance to this research effort.

The section covering economic analysis also cites several sources which address Bayesian and decision theoretic analyses. The economic decision system proposed in this research effort requires application of Bayesian and decision theoretic techniques. The literature which is reviewed is limited to that which is relevant to economic analysis of the research problem.

2.2 MEASUREMENT BIAS AND IMPRECISION

Several early articles focus on inspection of individual parts in the presence of measurement error. More recent works address the effects of measurement bias and imprecision on variables acceptance sampling plans.

Grubbs [23] first identifies a measurement as being composed of two components: one being the true (unknown) value of the characteristic, and the other an error in measurement. In a single measurement, these two components are inseparable. He develops a statistical method for estimating and comparing product variation and errors of measurement. The analysis assumes independence and normality of the true characteristic and measurement variation.

As an allowance for the inherent errors of measurement as identified by Grubbs, Eagle [21] creates "test specifications" set inside functional tolerances by an amount sufficient to ensure that non-conforming product is not accepted due to measurement error. The test limits are based on tradeoffs between producer and consumer risks. Error system estimates are obtained through the method of Grubbs [23]. Eagle identifies the two types of errors which are possible when measuring in the presence of inspection error: Consumer's Loss (CL), which is the probability that non-conforming product units will be accepted and Producer's Loss (PL), which is the probability that conforming product will be rejected.

In consideration of the test specifications proposed by Eagle, Grubbs and Coon [24] explore the proper placement of these limits. The authors deal with three criteria for placement: 1) Ensuring that producer and consumer risks are equal, 2) Minimizing the sum of producer and consumer risks and, 3) Minimizing the cost of making wrong decisions. The authors also address the use of multiple measures (constant n) on each sample part in order to ensure that risks are kept to some arbitrary minimum.

Lotti [33] discusses the tradeoffs involved with the placement of test limits relative to functional specifications. He concentrates on the concepts of False Acceptance Rates (FAR) and False Rejection Rates (FRR) in determining the consequences of locating "pseudo" tolerance limits.

Consideration of the effects of inspection error on variables acceptance sampling schemes is first presented by David, Fay and Walsh [17]. The authors propose a variables acceptance sampling plan which compensates for inherent errors of measurement. They deal with the case of no bias and a product distribution which is centered relative to specifications. The compensating plan, which considers only one of the tolerance limits, utilizes a constant number of repeat measurement made on each part in the sample.

Owen and Weisen [46] extend the method of David et al [17]. by considering cases of one-sided specification limits and non-centering of the product distribution. They utilize the bivariate normal distribution to develop inspection criteria which are based on inspection costs. Bias, as well as imprecision in measurement, is considered.

In a technical note, Diviney and David [20] summarize the problems introduced by measurement error when attempting to properly disposition product through variables acceptance sampling. They illustrate the performance of an acceptance sampling plan by variables in the presence of inspection error by imposing a "shadow" Operating Characteristic (OC) curve reflecting true plan performance on the OC curve defined by the inspection plan.

To compensate for the shift in OC curve due to measurement bias and imprecision, Mei, Case and Schmidt [37] propose adjustments in sample size and acceptance criteria. The result is a sampling plan which overlays the desired OC curve without inspection error. The authors give excellent definitions of measurement bias and imprecision. Bias is given as the "difference between the true dimension of a product and the average of a long series of repeated measurements on that unit." Imprecision is the "dispersion of repeated measurements on the same unit of product."

Hahn [25] presents a practical, numerical example of assessing the percentage of product conforming to a single specification in the presence of measurement and process variability. The first method presented places binomial confidence bounds on the observed percentage non-conforming. This is, essentially, the treatment of variables data as attributes data, representing an undesirable loss of information. The second method utilizes the estimators $\hat{\mu}_{meas}$ and $\hat{\sigma}_{meas}$ and the assumption of normality of the measured values in placing confidence limits on the percentage non-conforming. The third method is approximate and places confidence bounds on the percentage of actual values which satisfies specification limits. It utilizes an estimate of $\hat{\sigma}_{actual}$ and uses tabled values of the non-central t-distribution in placing bounds on the percentage of non-conforming product. This method treats the actual standard deviation of the product as if it were observed in obtaining degrees of freedom for bounding the percentage.

A paper by Jaech [27] extends the practical example presented by Hahn in making statistical statements about the lot quality. The author examines the third method by Hahn and the appropriateness of neglecting sampling error when constructing confidence intervals utilizing small sample sizes. The proposed solution given by Jaech utilizes an approximation to the degrees of freedom for $\hat{\sigma}_{actual}$. A simulation study supports the method proposed by the author.

Owen and Chou [45] examine the effect of measurement error on one-sided variables acceptance plans such as given in MIL-STD-414. They show effects on the plan OC curve in terms of the ratio of the standard deviation of the measuring instrument to the standard deviation of the object being measured. They also give error effects on the producer's and consumer's risks of a specified plan.

Again, extending the problem presented by Hahn, Mee [35] proposes a solution to the problem of placing confidence bounds on percentage non-conforming by utilizing existing tables for the tail area of a normal distribution. The three cases examined are: 1) Ratio of Variances ($R = \sigma^2_{meas} / \sigma^2_{actual}$) known, 2) Measurement error unknown, and 3) Variance Ratio Estimated from Repeated Measurements. In case 3, a fixed number of measurements is performed on each of the sample parts.

To supplement the research performed by Hahn, Jaech and Mee, a paper by Mee, Owen and Shyu [36] gives methods for computing confidence bounds on the proportion of product that is accepted through acceptance sampling by variables but actually fails to meet the performance specification. Again, the true product values and measurement errors are assumed to be independently distributed normal variates. Procedures are given for both known and unknown measurement variance. In the case of unknown measurement variance, repeated measurements are taken in order to obtain an estimate. The authors utilize existing tables of the non-central t-distribution and retabulate the tail area as a function of the ratio of the measurement and observed sample standard deviations.

Tang and Schneider [58] treat the effects of inspection error on a complete inspection plan. In complete inspection, each incoming item is inspected for conformance and reworked to a target value if found non-conforming. The authors use the Taguchi loss function in three cases for treating quality costs: 1) No inspection error, 2) Non-conforming product is reworked, then perfect inspection is performed, and 3) Non-conforming product is reworked, and no further inspection occurs.

2.3 VARIABLES SAMPLING AND THE NON-CENTRAL t-DISTRIBUTION.

Johnson and Welch [29] present practical applications of the non-central t-distribution. One application of this distribution is the case in which objects are classified as effective or defective according to whether values of a characteristic exceed or fall short of a fixed standard. The parent population of the objects must be normal. When the standard deviation is unknown, the non-central t-distribution provides information about the proportion of product falling beyond the standard. This distribution differs from the familiar Student's t-distribution in the additional non-centrality parameter, δ, representing the offset of the population mean from the fixed standard. In the case of Student's t-distribution, $\delta=0.0$, representing equal division between effective and defective product. The authors provide tables of the non-central t-distribution in a form useful for solving practical problems.

The first formalized plans for acceptance sampling by variables characteristics are presented by Lieberman and Resnikoff [32]. They present plans for a single quality characteristic, measurements of which are independent, identically distributed normal random variables. The plans are one-sided, in that tails of the distribution are controlled independently. The plans are indexed by code letter and AQL, each combination of which represents an OC curve in the collection. The probability of acceptance at the AQL varies from 0.89 to 0.99, following the practice of the published MIL-STD-105A (acceptance sampling by attributes, now in revision E). Plans are presented for known standard deviation, sample standard deviation used to estimate unknown standard deviation (s-method) and average range used to estimate unknown standard deviation (R-method).

The presentation by Lieberman and Resnikoff is the basis of MIL-STD-414 Sampling Procedures and Tables for Inspection by Variables for Percent Defective, published by the U.S. Department of Defense [38]. The form of this document closely follows MIL-STD-105x; the variables plan was issued to take advantage of the considerable savings in sample size realized by utilizing a variables plan rather than an attribute plan. The statistical principles underlying MIL-STD-414 and the Lieberman and Resnikoff paper are presented in a reference document published as a U.S. DOD Technical Report [34].

Tables of the non-central t-distribution require a triple entry since the distribution depends on the degrees of freedom (f) and the non-centrality parameter (δ); tables are found in various forms throughout the literature. Early tables presented by Johnson and Welch [29] do not deal directly with the probability integral, nor is it possible to obtain from them values of the density function. Resnikoff and Lieberman [48] use as an argument $x = \sqrt{\frac{t}{f}}$ to make the tables more compact. The authors also address the use of the non-central t-distribution in applying the WAGR sequential test for variables measurements. This sequential procedure is discussed in more detail in a later section of this document.

Owen [43] identifies a bivariate non-central t-distribution which may be utilized to model two-sided tolerance limits and two-sided acceptance sampling plans in which the tail proportions are controlled. The joint probability of interest is that the mean sample measurement is above the lower specification and below the upper specification limit. The authors present tables of the constants required to specify parameters of the sampling schemes.

Owen [44] also documents a method of addressing two-sided variables sampling plans which utilize the univariate non-central t-distribution. The proposed method controls the sum of the tail probabilities by reducing the problem to the two extreme cases of a band of OC curves (maximum at one-half of non-conformities in each tail, minimum at all of the non-conformities in a single tail). The two-sided case presented is for unknown mean and standard deviation of a normal population. Corrections are also given to previous works by Owen which describe application of the two-sided tables to the one-sided case.

Kirkpatrick [30] examines the problem of placing confidence limits on percent non-conforming in a one-sided sampling plan by variables characteristics. His work closely follows the statistical principles used by Lieberman and Resnikoff and MIL-STD-414 in obtaining point estimates of the percent non-conforming in a single tail of a normal distribution. Given the point estimates, tables are provided which bound the proportion with 90, 95 and 99 percent confidence limits from the non-central t-distribution. The procedure given is exact for one-sided plans; for two-sided plans, tabular values are approximate.

In a departure from the exact solution of unknown standard deviation methods (s-method) for variables sampling using the non-central t-distribution, Hamaker [26] proposes that the normal approximation is adequate for OC curve derivation. His objective is to compute variables sampling plans which are equivalent to well-known attributes plans (MIL-STD-105x). The author presents straightforward adjustments to apply to all cases of attributes acceptance sampling plans (standard deviation known, s-method, R-method) in order to achieve equivalent variables plans. Consideration is also given to the setting of fictitious limits in order to reduce sample size while maintaining OC curve performance.

Hamaker also makes an interesting point concerning the practicality of known standard deviation sampling plans. The author notes that, even if a large sample estimate of the standard deviation is available, it must still be demonstrated that σ does not vary from lot to lot. If σ is constant, but unknown, application of the appropriate s-method may require 3 to 4 times the sample size dictated by the known σ method.

Weingarten [61] also proposes a normal approximation for the case of unknown variance. The author gives a general procedure for obtaining confidence limits on percent non-conforming which is applicable for any sample size and confidence level in one-sided sampling plans. The method is based on a procedure due to Duncan which is based on a normal approximation instead of the appropriate non-central t-distribution. The author indicates that the approximation is excellent as long as the sample size exceeds ten.

The problem of constructing confidence limits on simultaneous (two-sided) sampling plans by variables is addressed by Chou and Owen [10]. The authors propose a method for bounding the percentage non-conforming for the unknown standard deviation case (s-method). Tables are given for various values of sample size, normalized specification limits and confidence level. The method is exact in utilizing the bivariate non-central t-distribution.

In general consideration of placing confidence bounds on an underlying normal distribution, Odeh, Chou and Owen [41] examine the effects of sample size. The authors look at two types of confidence intervals: 1) The β-expectation tolerance interval which contains $100\beta\%$ of the underlying distribution, and 2) The β-content tolerance interval which contains at least $100\beta\%$ of the population with confidence level γ. The general problem is in defining a Student's t-distribution confidence interval.

2.4 ECONOMIC ANALYSIS OF VARIABLES ACCEPTANCE SCHEMES

Wetherill and Campling [63] examine both attributes and variables acceptance sampling inspection in terms of decision theory. Decision theory considers the consequences of decisions in assessing the appropriateness of specific sampling plans. For the variables case, the authors assume a normally distributed process with a constant, known variance. In their opinion, the most difficult utility to estimate in evaluating consequences is the profit accruing from accepting conforming items. The authors investigate the effects of errors in formulation of the sampling model and errors in estimating parameters of the model. They also investigate improvement in utility using double and sequential sampling plans, rather than single sampling.

Schmidt, Case and Bennett [52] assess the use of economic criteria in selecting a variables sampling plan. The authors develop a total cost model which considers fixed and variable costs associated with inspection, acceptance, screening and scrapping of a lot of product. The model developed is distributionally general (assuming known variance of the product distribution) with an example presented which is specific to a normal product distribution.

In contrasting Bayesian and Decision Theoretic methods, Barnett [3] notes that both activities aim to extend the concept of "relevant information" beyond that obtained from sampling. Bayesian techniques augment sample data with prior information about the situation under study. Decision theory recognizes that actions imply consequences, and qualitatively combines assessments of these consequences with sample results to arrive at a sensible choice of action. Each action is assigned a particular loss (or cost) so that a loss function may be defined over the entire realm of possible actions. By combining prior process knowledge, classical statistics and consequences of any action taken, it is possible to arrive at an informed decision concerning the sample at hand. In one hypothetical example from industry, Barnett illustrates the Bayesian approach using a normal prior distribution. The normal distribution is a conjugate prior, in that it yields a posterior within the same (normal) family.

Ladany [31] examines the effects of changing economic conditions on a Bayesian acceptance plan for attributes. The model utilizes a prior binomial distribution, subsequently approximated by a normal distribution for ease of use. The author performs a "reverse analysis" by modifying economic conditions and then examining the implications of the change on the statistical parameters specifying the sampling plan. The model equates the expected cost of lot acceptance with the expected cost of lot rejection using Bayesian techniques, then finds the corresponding value of lot percent defective for a single sample acceptance plan.

By integrating variables acceptance sampling, measurement error and economic considerations, Case and Bennett [7] illustrate the adverse monetary effects of imperfect measurement in variables acceptance plans. The authors assume normal distribution of the measurement error, with mean (bias) and variance (imprecision) known. Additionally, lot and measurement error distributions are assumed independent. Cost concepts are developed generally, with no assumption of product distribution. The authors note that high bias and/or imprecision cause the cost model to be dominated by specific terms and take the cost of the plan to some upper limit.

In illustrating the unfavorable cost consequences associated with errors of measurement in an acceptance sampling plan by attributes, Collins, Case and Bennett [11] note that any error at all results in economic loss. They use a basic Guthrie-Johns cost model and show incremental costs which result from neglecting inspection error in selecting an acceptance sampling plan. The authors present arguments with which to convince practicing Quality

Control managers of the significant costs which may be incurred by ignoring inspection error.

Schmidt, Bennett and Case [51] present a three-action cost model which may be used in selecting appropriate lot disposition in acceptance sampling by variables. The decision criteria given in the model may call for lot acceptance, lot screening or lot scrapping. The model considers the estimated quality of the lot in determination of the most cost effective method of lot disposition following sampling results. The authors present an optimal solution and an approximation which is dependent on sample size in approaching the optimal solution.

Chen [9] examines economically-based acceptance double sampling by attributes. The author redevelops the Guthrie Johns model for single sampling into a model appropriate for double sampling. Fixed costs associated with inspection, acceptance and rejection are also included in the economic model. The double sampling plan considered by Chen is Bayesian in nature; in the case in which two samples are required, information obtained from the first sample is combined with results of the second sample to make inferences about the lot quality.

Using a decision theoretic approach to the variables acceptance sampling problem, Fertig and Mann [22] explore the savings in sample size achievable by accounting for finite lot size. The economic cost of making a disposition decision is measured by a loss function. The model assumes that any non-conforming items found in the sample are replaced by conforming items and that rejected lots are screened for non-conforming items which are replaced by conforming ones. The accept/reject costs are balanced by assuming that there are average costs that the producer is willing to incur if the process is operating at either extreme of the OC curve (acceptable quality level and rejectable quality level). The authors assume a normally distributed process with mean and standard deviation unknown.

Boucher and Jafari [6] derive an economic optimal solution to the problem of selecting a process level when subjecting lots to variables acceptance sampling per MIL-STD-414. The authors treat only the case in which process variance is known. The cost model developed assumes that a variable cost per unit relative to the unit measure of the quality characteristic is incurred in addition to a fixed unit cost. Conversely, rejected lots represent a fixed unit revenue and a penalty cost proportional to the deficit in the quality characteristic.

Moskowitz and Tang [39] utilize the three-action cost structure as proposed by Schmidt et al. [52] in performing Bayesian analysis of known variance acceptance sampling by variables. The authors recognize three ways of obtaining a prior sampling distribution: empirically (past events), subjectively, or some combination of these two. They consider a prior normal distribution, with the performance variable subjected to two-sided requirements. Both the quadratic and step-loss functions are examined in reaching the optimal total cost model. The Bayes optimal sampling plan, as obtained by the authors, is robust with respect to the form of the prior distribution, as well as to mis-specification of the mean and variance, as long as the tail specification reasonably approximates that of a normal distribution.

2.5 SEQUENTIAL ANALYSIS

In his definitive work on sequential analysis, Wald [59] explains that the number of observations required by the sequential procedure depends on the observations, and is not predetermined, but a random variable. The Sequential Probability Ratio Test (SPRT) for testing the mean of a normal distribution with known variance, credited to Wald, utilizes a likelihood ratio which is computed following each sequential sample. The likelihood ratio used in the Wald SPRT is simply the probability that the alternative hypothesis is true (given the current sample outcome) over the probability that the null hypothesis is true (given the current sample). Based on the value of the likelihood ratio, one of three possible decisions is made following each sample taken: 1) the hypothesis is not rejected, 2) the hypothesis is rejected, or 3) another sample is drawn.

In the case of simple hypotheses, the Wald sequential test effects the greatest possible savings in the average number of observations over other sequential and non-sequential tests. A hypothesis is said to be simple if it determines, uniquely, the values of all unknown parameters of the distribution. This is the case in tests concerning the mean of a normal distribution with known variance. Wald gives the hypotheses of this test as,

$$H_0: \theta = \theta_0 \qquad \text{and} \qquad H_1: \theta = \theta_1$$

where θ is the unknown mean of the distribution, θ_0 is a value of the mean below which rejection of the lot is considered to be an error of practical consequence and θ_1 is a value of the mean above which rejection of the lot is considered to be a practical error. The author suggests that the interval between θ_0 and θ_1 is a zone of indifference, in which mean values occur for which there is no particular preference between decisions to accept and reject the lot.

For the case of composite hypotheses, as in a normal distribution with unknown variance, Wald proposes a system of weight functions by which the composite hypotheses (dependent of the variance) are transformed to simple hypotheses (independent of the variance). The composite nature of the hypotheses arises due to the fact that the variance is a nuisance parameter in assessing hypotheses concerning the parameter of interest, the mean. Unknown variance problems, also termed sequential t-tests, typically perform a transformation to eliminate the nuisance parameter from the likelihood ratio. However, a problem arises in calculating the OC curve and the Average Sample Number (ASN) in the composite case. Wald proposes that the Average Sample Number (ASN) for the unknown variance case is bounded (on the low side) by the ASN for the known variance case.

In a classic book compiled by the Columbia Statistical Research Group, Wallis [60] notes that sequential is superior to non-sequential analysis whenever 1) the data becomes available serially and 2) the cost of the data is approximately proportional to the amount of data. He defines superiority in terns of minimizing the set of quantities (N, α, β), where N is the number of observations, α is the risk of erroneously rejecting the hypothesis and β is the risk of erroneously accepting the hypothesis.

Sobel and Wald [54] extend the known variance sequential problem to a multi-decisional case. The problem is to choose one of three mutually exclusive hypotheses:

$$H_1{:}\theta < a_1 \qquad H_2{:}a_1 \le \theta \le a_2 \qquad H_3{:}\theta > a_2,$$

where θ is the unknown mean of the distribution and a_1 and a_2 are the lower and upper specifications, respectively. In order to deal with this problem, the authors divide the parameter space into five mutually exclusive and exhaustive zones. They define indifference zones around a_1 in which there is no strong preference between H_1 and H_2, and around a_2 in which there is no real preference between H_2 and H_3. The problem then proceeds as the Wald SPRT for simple hypotheses in treating a_1 and a_2 simultaneously.

A publication by the National Bureau of Standards [57] provides tables with which to perform sequential t-tests (unknown variance). The tables make use of the confluent hypergeometric function, F(n/2, 1/2;x). The likelihood ratio targeted by the tables is that of a non-central t-distribution to a t-distribution (Student's). This method is appropriate when the null hypothesis is such that it specifies the offset of the distribution mean relative to some fixed specification as zero, rather than a non-zero offset. The more general case, as described in the next paragraph, is less restrictive in utilizing a likelihood ratio of two non-central t-distributions.

A sequential t-test (unknown variance) given by Rushton [49] specifies a likelihood ratio of two non-central t-distributions. The null hypothesis tested states that X is normally distributed with unspecified standard deviation σ and mean $\mu=\delta\sigma$, δ being specified; the alternative hypothesis states that X is normally distributed with unspecified standard deviation σ and mean $\mu=\delta'\sigma$, δ' being specified. The author presents the exact formula for the likelihood ratio which involves the Hh function tabulated by Airey [1]. Rushton gives an approximate solution which he states is satisfactory except in the case that hypotheses are far apart and the sample size is in consequence small. He notes that, due to non-linearity of the test, it is not possible to calculate an estimate of the ASN. Rushton suggests that the Wald ASN approximation for unit variance may be appropriate for sample size greater than 30.

Cox [16] develops a method analogous to Rushton's in treating tests of composite hypotheses other than that for the mean of a normal distribution, variance unknown. The procedure may be used in many problems in which a jointly sufficient set of estimators can be found for the unknown parameters. The author presents examples for sequential test of variance (normal distribution, mean unknown), sequential analysis of variance, variance ratio test and test for correlation coefficient. Cox also illustrates that all methods he presents (and Rushton's procedure) can be obtained by Wald's method of weight functions in treating composite hypotheses.

David and Kruskal [18] provide a proof that the WAGR sequential t-test (known variance) terminates with probability one. The test, named for Wald, Arnold, Goldberg and Rushton, tests that the proportion of a normal population greater than a given constant is p_0 (given) versus p_1 (given). The author notes that this probability condition must be met in order to apply the standard Wald SPRT in testing the two hypothesis points.

The Wald SPRT procedure for simple hypotheses (known variance) is further explored by DeGroot and Nadler [19]. They indicate that the optimal properties of the OC and ASN make the test very appealing. However, due to the need to know the population variance exactly, the SPRT procedure has limited applicability in practice. The authors indicate that, often, the variance is approximated in order to utilize the straightforward Wald SPRT procedure. They look at problems of testing means and proportions defective; additionally, they examine the sensitivity of the SPRT to departures of the variance from its assumed value. A procedure is given for an SPRT test when the variance can be restricted to a finite interval a priori.

A comprehensive review of literature relevant to sequential analysis is presented by Johnson [28]. He indicates that part of the appeal in the standard Wald SPRT is in that it does not require special tables for application. In regards to tests of composite hypotheses, the author notes that there are no approximate formulae for the operating characteristic or ASN for the test. He also discusses sequential estimation, curtailed sampling and two-sample procedures as presented in the literature.

Schneiderman and Armitage [53] present approximate procedures for use in sequential t-tests (variance unknown). The methods are modeled after so-called "wedge plans" for testing the mean of a population with known variance. Wedge plans provide a bridge between open (unlimited) plans such as Wald and restricted (truncated) plans. The authors' approximation utilizes boundaries derived for known variance problems based on studentization of the t-distribution for large sample sizes. They note that the approximating procedure is necessarily somewhat arbitrary and the argument heuristic. The authors term the argument the PVK ("pseudo-variance known") conjecture.

A series of FORTRAN subroutines is provided by Cooper [12, 13, 14, 15] for use in calculating tail integrals of the normal, Student's t- and non-central t-distributions. The numerical method of the non-central t-distribution subroutine closely follows that given by Owen [43].

Billard and Vagholkar [5] provide an alternative procedure for the multi-decision method as provided by Sobel and Wald. For the known variance case, their method provides an ASN function in addition to the OC function which was previously derived. The new procedure specifies that a "few" observations be taken before any serious comparison of the hypotheses is undertaken. This is justified due to the authors' observation that few experimenters would be content to terminate the testing process very early (say, n=2), especially when the difference between the values of the mean for the null and alternative hypotheses is very small. Following the initial sample, the Wald SPRT procedure is followed.

Wetherill [62] makes the distinction between two kinds of composite hypotheses: 1) those involving ranges of the parameters of interest, and 2) those involving nuisance parameters, as in the sequential t-test. In the second case, the author indicates that it is generally desirable to use methods given by Cox or Rushton rather than the weight functions given by Wald. Cox and Rushton utilize methods which construct a test statistic having a distribution not dependent on the nuisance parameter; Wald's method of weighting essentially integrates out the nuisance parameter with a seemingly arbitrary function.

2.6 SUMMARY

The four areas which are covered in the literature review all have relevance to the research topic. It is widely acknowledged that measurement error can distort product conformance to engineering specifications. Methods for measurement error compensation have been applied in the areas of acceptance sampling and control charting. However, the problem of accounting for measurement error in single batch (or item) disposition has not been treated.

Sequential sampling techniques have been extensively developed and are well-known in application to lot-by-lot acceptance plans by attributes and by variables. The problem of single item disposition in the presence of normally distributed measurement error bears a resemblance to lot sampling by variables from a normal population. The application of sequential methods in the single item situation, the subject of the research effort, has received little attention in the literature.

The decision system which is proposed utilizes accepted techniques of sequential sampling in addressing the problem of single item disposition by variables in the presence of measurement error. Consideration of economic inspection criteria in assessing performance of the decision system develops an additional area which has not been addressed in the literature.

CHAPTER 3

THEORETICAL DEVELOPMENT
OF THE STATISTICAL SOLUTION TO THE PROBLEM

3.1 INTRODUCTION

This chapter addresses the statistical solution of the problem of homogeneous batch disposition on the basis of a single variable characteristic subject to measurement error. Solutions are presented for the cases of known and unknown measurement error variance. Tolerable risks of errors in acceptance and rejection are considered in the statistical model; explicit economic consequences of disposition errors are neglected in reaching a decision for batch disposition.

The case of known measurement error variance implements the Sequential Probability Ratio Test (SPRT) first introduced by Wald [59]. This sequential theory has not previously been applied to the problem of homogeneous batch disposition subject to measurement error.

Solution of the problem of unknown measurement system variance involves the non-central t-distribution. Problems of this nature are known as sequential t-tests. Previous solutions to these problems utilizing SPRT theory have involved ratios of non-central to central t-distributions and require extensive table searches for application. Alternatively, approximations have been developed which treat sequential t-tests. The solution presented in this chapter applies SPRT theory in the exact solution of the sequential t-test problem, utilizing the ratio of two non-central t-distributions. In addition to this development, this sequential theory has not previously been applied to the problem of homogeneous batch disposition subject to measurement error.

3.2 SEQUENTIAL SAMPLING

A sequential test of statistical hypothesis involves making a calculation following each sequential observation (or group of observations) and determining a course of action. Rather than making a determination on a random sample of n observations, as in non-sequential methods, the sample size is a random variable and unknown prior to beginning inspection. After any single observation (or group of observations), the hypothesis may be rejected, the hypothesis may fail to be rejected or data collection may continue. The criteria for decision-making are pre-determined and a decision is reached as soon as enough data is available to satisfy specified risk levels.

Sequential tests in statistics offer several advantages over non-sequential tests. Sequential methods prove superior when, 1) data becomes available serially, and 2) the cost of the data is proportional to the amount of data. When data becomes available in fixed quantity and the cost of collection is fixed (overhead), non-sequential methods prove superior in minimizing the parameters of interest N, α and β (respectively, the sample size, the tolerable risk of rejection when the null hypothesis is true, and the tolerable risk of acceptance when the null hypothesis is false). Unlike non-sequential tests, the number of sequential observations, N, is variable; for fixed α and β, the average number of observations ($\overline{N}$) is minimized.

The fundamental quantity computed after each observation in a sequential test is the "likelihood ratio". Given all observations available, probabilities of observing the accumulated data (composed of n observations) are calculated assuming the null hypothesis is true (p_{0n}) and the alternative hypothesis is true (p_{1n}). The ratio of these two probabilities, p_{1n} / p_{0n}, is the likelihood ratio (λ_n) at that point in the sampling sequence. If this ratio ever exceeds a certain level, A, the testing stops and the null hypothesis is rejected . If the likelihood ratio falls below a given level, B, data observation stops and the null hypothesis is not rejected. In the case that the calculated ratio falls between the two values, A and B, evidence is insufficient to support either hypothesis based on predetermined risk levels (α and β) and the experiment continues. The testing thresholds, A and B, are completely determined from the specified risk levels.

Assuming that successive observations are independent, the likelihood ratio for a set of observations can be found by taking the likelihood ratio of the most current observation and simply multiplying it by the ratio obtained from all preceding observations. By utilizing the natural logarithm of the likelihood ratio, the sequential mathematics reduces to simple addition and subtraction.

When first presented, practical sequential testing procedures relied heavily on graphical procedures. In the graphical analysis, parallel lines are calculated and plotted on a graph of Σx vs. n which define the acceptance, rejection and continuation regions for each sequential n. These lines incorporate the natural logarithms necessary for carrying out the testing and remove any heavy mathematics from the operator. With the graphical testing description in-hand, the operator need only measure the characteristic of interest and calculate the summation of the measurements (over n). The operator then plots this quantity on the chart and determines the proper course of action based on the region of the chart in which the value falls. Details of the graphical procedure are provided in Wald's book on Sequential Analysis [59].

The use of computers in testing and measurement has further removed underlying mathematics from the responsibility and view of the inspector. Complex calculations are carried out behind-the-scenes, and the need for the simplifying (but, still, time-consuming) graphical procedures is eliminated. Due to the widespread availability of computers at the time of this writing, graphical procedures as they relate to the research topic are omitted from this research.

Although the SPRT terminates with probability one, in any single experiment the number of required observations may be very large. In many cases, it may be desired to establish a maximum number of samples to be taken in the experiment. Truncation of the SPRT changes the risk probabilities (α and β) associated with the procedure. This research addresses the truncated SPRT for simple hypotheses (applicable to the problem of known measurement error variance) in order to provide a valid basis for comparison with the economic sequential problem, as developed.

3.3 TESTS OF SIMPLE HYPOTHESES

Sequential Probability Ratio Test

In the case of testing a simple hypothesis H_0 against a single alternative H_1, Wald [59] defines the Sequential Probability Ratio Test (SPRT) for application following each observation. A hypothesis is said to be simple if it determines, uniquely, the values of all unknown parameters of the subject distribution. Parts of the following discussion are taken from Wald [59].

Let $f(X,\theta)$ represent the distribution of the random variable X under consideration, with θ the only unknown parameter of the distribution. Let H_0 be the hypothesis that $\theta=\theta_0$ and H_1 the hypothesis that $\theta=\theta_1$. The Sequential Probability Ratio Test for testing H_0 against

H_1 proceeds as follows: Positive constants A and B (B<A) are chosen. After each (nth) trial of the experiment the likelihood ratio is computed as

$$\frac{p_{1n}}{p_{0n}} = \frac{f(x_1,\theta_1)\ldots f(x_n,\theta_1)}{f(x_1,\theta_0)\ldots f(x_n,\theta_0)}.$$

If

$$B < \frac{p_{1n}}{p_{0n}} < A,$$

another sample is drawn and the experiment continues. If

$$\frac{p_{1n}}{p_{0n}} \geq A \tag{1}$$

the process terminates with the rejection of H_0. If

$$\frac{p_{1n}}{p_{0n}} \leq B \tag{2}$$

the process terminates with the failure to reject H_0.

The sequential nature of the computations makes it practically convenient to compute the natural logarithm of the likelihood ratio p_{1n}/p_{0n} rather than the raw ratio. This is because $\ln(p_{1n}/p_{0n})$ can be written as the sum of n terms:

$$\ln\frac{p_{1n}}{p_{0n}} = \ln\frac{f(x_1,\theta_1)}{f(x_1,\theta_0)} + \cdots + \ln\frac{f(x_n,\theta_1)}{f(x_n,\theta_0)}.$$

In practice, the expression $z_i = \ln\frac{f(x_i,\theta_1)}{f(x_i,\theta_0)}$ is calculated following each sequential iteration. If

$$\ln B < \sum_{i=1}^{n} z_i < \ln A$$

the testing continues with an additional observation. If

$$\sum_{i=1}^{n} z_i \geq \ln A$$

the process terminates with the rejection of H_0. If

$$\sum_{i=1}^{n} z_i \leq \ln B$$

the process terminates with the failure to reject H_0.

The constants A and B required for the described testing are determined such that the specified test has prescribed strength (α, β). Given inequality (1) which dictates rejection of H_0, the probability of obtaining a sample which meets the specified criterion is clearly at least A times as large under hypothesis H_1 as under hypothesis H_0. Thus, the probability measure of the totality of all such samples is also at least A times as large under H_1 as under hypothesis H_0. The probability measure of the totality of all samples meeting this criterion is the same as the probability that the sequential test will terminate with the failure to reject H_0. But this latter probability is equal to α when H_0 is true and to 1-β when H_1 is true. This yields $1-\beta \geq A\alpha$, also written as

$$A \leq \frac{1-\beta}{\alpha}.$$

A lower limit for B is derived in a similar manner utilizing equation (2). The probability of failing to reject H_0 is at most B times as large when H_1 is true as when H_0 is true. The probability of failing to reject H_0 is 1-α when H_0 is true and β when H_1 is true, yielding $\beta \leq (1-\alpha)B$, also written as

$$B \geq \frac{\beta}{1-\alpha}.$$

Thus, $\beta/(1-\alpha)$ is a lower limit for B.

These limiting equalities for A and B are derived under the assumption that successive observations are independent observations of X.

Truncation of the SPRT involves establishment of a definite upper limit, n_{max}, for the number of allowed observations. A straightforward, general rule for truncation of the sequential test is proposed by Wald [59] and verified by Baker [2]. It proceeds as follows: If the SPRT does not lead to a final decision for any $n<n_{max}$, fail to reject H_0 on the n_{max}th trial when

$$\ln B < \sum_{i=1}^{n} z_i \leq 0,$$

and reject H_0 on the n_{max}th trial when

$$0 < \sum_{i=1}^{n} z_i < \ln A.$$

Truncation of the process at n_{max} affects the probabilities of types I and II errors. The relative effect of truncation on these error probabilities depends on the value of n_{max}; the larger n_{max}, the smaller is the effect of truncation on α and β. Upper bounds on the error risks, given by Wald [59], assume that n_{max} is large enough such that $z_1, \ldots, z_{n_{max}}$ can be regarded as normally distributed. Bounding the error risks requires consideration of cases in which the truncated and non-truncated process lead to conflicting conclusions about the null hypothesis.

Known Measurement Distribution Variance

The situation of known measurement system variance follows directly from Wald's treatment of simple hypotheses using SPRT methods. SPRT theory has not previously been applied to the problem of homogeneous batch disposition based on a single variable characteristic subject to measurement error. The true value of the variable characteristic being measured is the only unknown parameter; for the case of no measurement bias, the (normal) measurement error distribution is centered about the unknown mean. In the case of a single, upper specification limit, the hypothesis may be stated as H_0: The unknown batch characteristic, μ, is less than or equal to the specified upper limit, U. Alternatively, H_1 states that the unknown parameter exceeds the specification, U (the batch is non-conforming).

If μ is equal to U, the upper specification limit, there is an indifference as to the disposition of the batch. As μ becomes increasingly greater than U, the preference favors rejection of the batch; as μ decreases from U, it is preferred to accept the batch based on the variable characteristic being measured. The relative closeness of μ to U greatly influences the degree of preference for each alternative disposition option. Generally, it is possible to define some "indifference" limits, U_0 and U_1 ($U_0 < U < U_1$), about U such that rejection of the batch is an error of practical consequence (as judged by the experimenter) when $\mu \le U_0$ and acceptance of the batch is a practically significant error when $\mu \ge U_1$. The range of values which fall between U_0 and U_1 defines the region of indifference for batch disposition. Selection of this zone is not a statistical problem; the indifference region is selected on the basis of practical considerations concerning the consequences of a wrong decision.

The risks to be tolerated, α and β, are chosen after definition of the indifference limits, U_0 and U_1 and relate to these limits, rather than the specification.

Let X_i denote the ith observed measurement on the current batch. It is assumed that X_i is a random variable given as

$$X_i = \mu + \varepsilon_i,$$

where μ is the true value of the batch characteristic and ε_i denotes a random measurement error component. The measurement errors are assumed to be distributed independently as $N(\mu_{me},\sigma_{me}^2)$, so that the observations are distributed normally as $N(\mu+\mu_{me},\sigma_{me}^2)$. The mean of the measurement error distribution, μ_{me}, is the bias of the measurement system, considered to be known and fixed.

Given:

X_i=random variable of observation on μ, trial i

μ=true value of the batch characteristic, unknown

x_i=observed value of μ, sequential trial i (i=1,...,n)

μ_{me}=mean of the measurement error distribution (bias)

σ_{me}=standard deviation of the measurement error distribution

U=upper specification limit for the variable characteristic of interest

U_0=lower indifference limit for the variable characteristic of interest

U_1=upper indifference limit for the variable characteristic of interest

β=tolerable risk of acceptance when the true value is greater than U_1

α=tolerable risk of rejection when the true value is less than U_0

A=lower decision limit for the likelihood ratio, given approximately as $(1-\beta)/\alpha$

B=upper decision limit for the likelihood ratio, given approximately as $\beta/(1-\alpha)$

For initial development of the SPRT procedure as it applies to the research problem, measurement system bias, μ_{me}, is assumed to be zero. The case of non-zero bias is considered following the zero bias case.

In applying the SPRT procedure, successive observations are made on μ, the variable batch characteristic of interest. If $\mu = U_0$, the probability density of the sample of observations $(x_1, \ldots, x_n)$ is given by

$$p_{0n} = \frac{1}{(2\pi)^{\frac{n}{2}} \sigma_{me}{}^{n}} e^{\frac{-1}{2\sigma_{me}{}^{2}} \sum_{i=1}^{n} (x_i - U_0)^2},$$

and if $\mu = U_1$, the probability density function (p.d.f.) is given by

$$p_{1n} = \frac{1}{(2\pi)^{\frac{n}{2}} \sigma_{me}{}^{2}} e^{\frac{-1}{2\sigma_{me}{}^{2}} \sum_{i=1}^{n} (x_i - U_1)^2}.$$

The probability ratio p_{1n}/p_{0n} is calculated following each sequential observation. The inspection continues as long as

$$B < \frac{p_{1n}}{p_{0n}} = \frac{e^{\frac{-1}{2\sigma_{me}{}^{2}} \sum (x_i - U_1)^2}}{e^{\frac{-1}{2\sigma_{me}{}^{2}} \sum (x_i - U_0)^2}} < A. \tag{3}$$

The batch is accepted if $p_{1n}/p_{0n} \leq B$; the procedure terminates in rejection of the batch if the ratio $\geq A$.

As previously suggested, practical implementation of the decision procedure is facilitated by taking advantage of natural logarithms of the inequalities. Simplifying, equation (3) may be rewritten as

$$\ln\left(\frac{\beta}{1-\alpha}\right) < \frac{U_1 - U_0}{\sigma_{me}{}^{2}} \sum_{i=1}^{n} x_i + \frac{n}{2\sigma_{me}{}^{2}} (U_0{}^2 - U_1{}^2) < \ln\left(\frac{1-\beta}{\alpha}\right). \tag{4}$$

The decision test limits of inequality (4) may be calculated prior to beginning the inspection procedure. The current observation is then simply added to the sum of all previous observations and the log of the likelihood ratio is tested against the decision limits. Decision criteria for disposition remain as previously indicated in equation (3).

In consideration of a measurement error distribution with non-zero mean (bias), the mean is simply subtracted from the observed value of the characteristic of interest. Recall that positive bias implies observed value greater than true value; negative bias produces an observation which is less than the true value of the characteristic. Rewriting equation (4) in consideration of non-zero bias gives

$$\ln\left(\frac{\beta}{1-\alpha}\right) < \frac{U_1 - U_0}{\sigma_{me}^2} \sum_{i=1}^{n} (x_i - \mu_{me}) + \frac{n}{2\sigma_{me}^2}(U_0^2 - U_1^2) < \ln\left(\frac{1-\beta}{\alpha}\right). \qquad (5)$$

Equation (5) is general and applies for both zero and non-zero measurement system bias.

The testing theory for the specific situation involving a single, upper variable specification extends directly to the case of a lower specification, L. Let the null hypothesis, H_0, be expressed as: The unknown batch characteristic, μ, is greater than or equal to the specified lower limit, L. Alternatively, H_1 states that the unknown parameter falls short of the specification, L (the batch is non-conforming). The indifference limits may then be given as L_0, the cutoff for batch acceptance and, L_1, the lower indifference limit beyond which the preference is for batch rejection.

With these parameter definitions, the theory previously developed is directly applied to the lower specification limit case. The inequality expression, including non-zero measurement error bias, is expressed as

$$\ln\left(\frac{\beta}{1-\alpha}\right) < \frac{L_1 - L_0}{\sigma_{me}^2} \sum_{i=1}^{n} (x_i - \mu_{me}) + \frac{n}{2\sigma_{me}^2}(L_0^2 - L_1^2) < \ln\left(\frac{1-\beta}{\alpha}\right).$$

The testing procedure for the case of known measurement system variance (upper specification limit) is shown in flowchart form in Figure 3.1. This is the logic utilized for practical solution of the problem via the FORTRAN program discussed in the following chapter. The case of a single lower specification limit is solved using the exact same logic with the appropriate substitution of limit (specification and indifference) variables.

3.4 TESTS OF COMPOSITE HYPOTHESES

General Discussion

A dominant, attractive feature of SPRT theory is its generality. However, application of the SPRT procedure is limited to those cases in which H_0 and H_1 are simple hypotheses. Many practical statistical procedures, such as t-tests and most analysis of variance tests, involve composite hypotheses in which the values of certain parameters are not completely specified. Tests of the mean of a normal distribution with unknown variance, also called sequential t-tests, involve such composite hypotheses. For practical application of SPRT theory in approaching the complex, composite problem, Wald [59] proposes a method of weighting the simple hypotheses included in a given composite hypothesis by defining prior distributions for the undefined (nuisance) parameters.

When published, Wald's system of weighting was not felt to be unequivocally satisfactory [28] and received little practical application to the problems of sequential t-tests. For a period of time, the only practical way in which sequential analysis could be applied to these problems was by replacing the composite hypotheses with simple hypotheses, thereby neglecting some available information. In the case of testing a variable dimension against one or more specification limits, this could be accomplished by approaching the

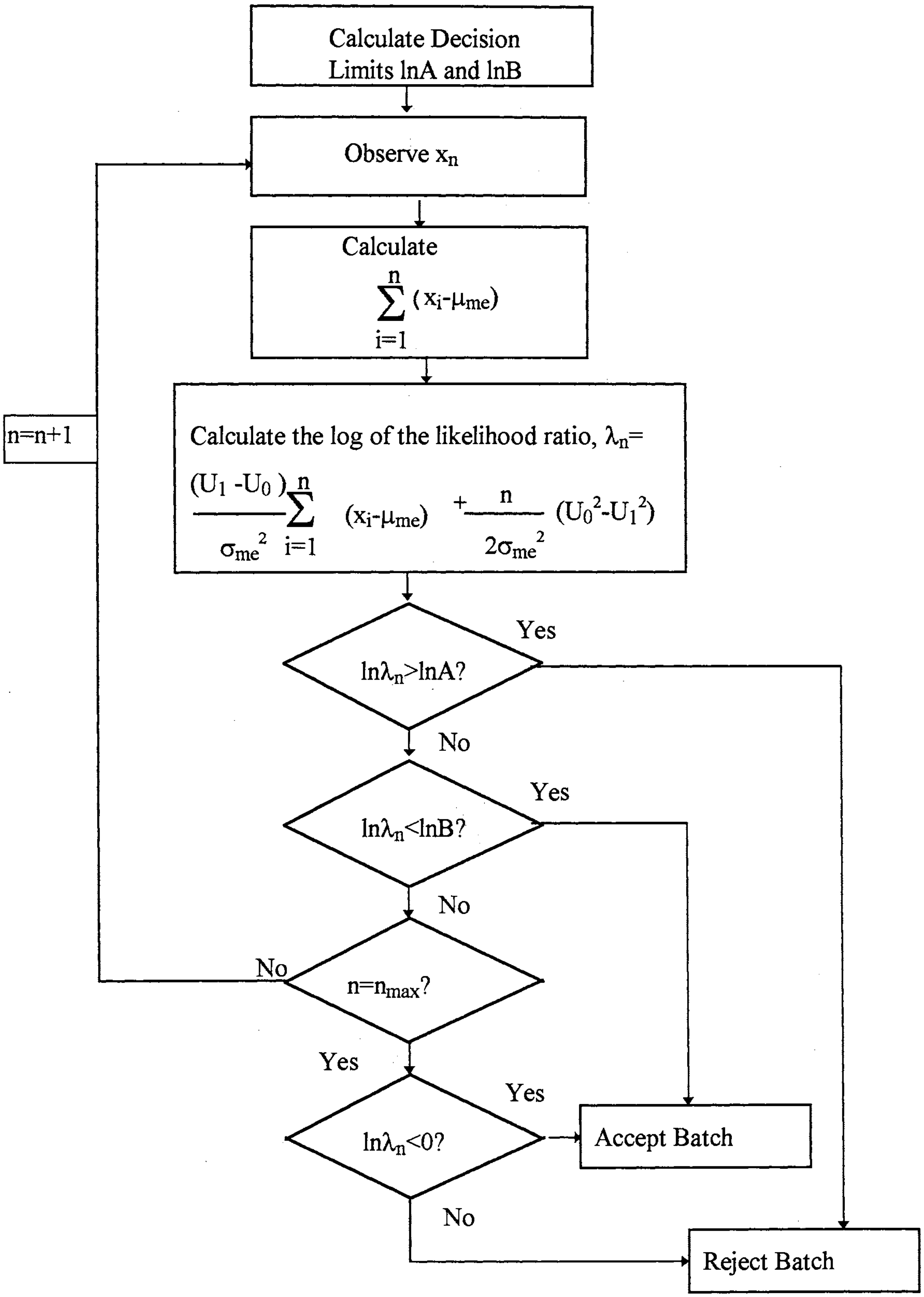

Figure 3.1. Procedure Flowchart for the Case of Known Measurement System Variance, Upper Specification.

problem in terms of the percentage of defective product. Wald [59] presents the practical procedure of such a problem.

An acceptable extension of the SPRT problem to the sequential t-test utilizes a sufficient statistic for the unknown mean (θ) which depends only on that mean [see references 16, 49 and 57]. Essentially, given the statistic T_n ($x_1,\ldots,x_n$), which depends only on θ, it is acceptable to use the ratio $p(T_n|\theta_1)/p(T_n|\theta_0)$ in testing sequentially. The testing procedure must terminate with probability one in order to be valid.

The composite problem treated in this research is the sequential t-test, in which the mean of a normal distribution with unknown variance is tested against a fixed limit. Details of the sequential solution of this problem (which integrates SPRT theory and treatment of composite hypotheses) are presented in the following section of this paper.

Unknown Measurement Distribution Variance

Exact solution of the sequential problem with unknown variance through application of likelihood ratios involves the non-central t-distribution. Procedures exist which solve, through tabular methods, the problem involving a ratio of a non-central to a central t-distribution. However, no such methods exist for solving the general case represented by the ratio of two non-central t-distributions. The following procedure utilizes SPRT theory in solving the exact, general case of the sequential t-test. In addition to this contribution, application of this theory to the problem of homogeneous batch disposition by a single variable characteristic in the presence of measurement error has not previously been explored.

Application of Wald's SPRT theory allows the variable nature of the characteristic to be retained. The general theory of sequential tests, as previously described, dictates to take observations sequentially and calculate, at each stage, the likelihood ratio λ_n (p_{1n}/p_{0n}). If

$$B < \frac{p_{1n}}{p_{0n}} < A,$$

another sample is taken and the experiment continues. Failure of the inequality on the right leads to rejection of H_0; failure on the left leads to a failure to reject H_0.

Rushton [49] presents the sequential solution for the composite case of testing the mean of a normal distribution with unknown variance against a specified upper limit. However, he stops short of practical implementation of the exact solution and presents an approximation which utilizes the Hh function tabled by Airey [1]. Because existing tables of the Hh function are practically limiting, Rushton checks his approximation by utilizing the confluent hypergeometric function which is closely related to the Hh function. The following procedure represents exact solution of the sequential t-test problem by implementing SPRT theory.

The solution procedure for the case of unknown variance is general as to the nature of the single specification limit (upper or lower); the FORTRAN computer program which is subsequently presented accommodates both situations. As in the case of known variance, the testing theory is first developed for the case of zero bias (μ_{me}=0) and extended to cover the general bias case. Let

X_i=random variable of observation on μ, trial i

μ=true value of the batch characteristic

x_i=observation on μ, trial i (i=1,...,n)

$\bar{x}_n$=average of n observations, $\frac{1}{n}\sum_{i=1}^{n} x_i$

s_i=standard deviation of i observations, i=1,...,n

μ_{me}=known mean of the measurement error distribution (bias)

σ_{me}^2=unknown true variance of the measurement error distribution

δ_0=null hypothesis ratio, μ_0/σ_{0me}

δ_1=alternative hypothesis ratio, μ_1/σ_{1me}

U=upper specification limit for the variable characteristic of interest

L=lower specification limit for the variable characteristic of interest

α=tolerable risk of error when H_0 is true

β=tolerable risk of error when H_1 is true

p_{0n}=probability of observing accumulated n observations assuming H_0 is true

p_{1n}=probability of observing accumulated n observations assuming H_1 is true

λ_n=likelihood ratio, p_{1n}/p_{0n}

Prior to making a single observation, the probability of interest may be expressed as

$$P(X_i < U) = \Phi\left(\frac{(U-\mu)}{\sigma_{me}}\right) \qquad \text{(Upper Specification)}$$

or

$$P(X_i > L) = \Phi\left(\frac{(L-\mu)}{\sigma_{me}}\right) \qquad \text{(Lower Specification)}$$

where $\Phi(x)$ is the cumulative distribution function for the standard normal distribution. Shifting the origin of all measurements by subtracting the specification limit allows the probabilities to be written as

$$P(X_i<0)=\Phi(-\delta) \qquad \text{(Upper Specification)}$$

and

$$P(X_i>0)=\Phi(-\delta) \qquad \text{(Lower Specification)}$$

where $\delta=\mu/\sigma_{me}$. Note that the probabilities shown are general for both lower and upper specification limits.

For a single trial, H_0 states that X_i is normally distributed with unspecified standard deviation σ_{me} and mean $\mu=\delta_0\sigma_{me}$ (δ_0 specified) and H_1 states that X_i is normally distributed with unspecified standard deviation σ_{me} and mean $\mu=\delta_1\sigma_{me}$ (δ_1 specified). By considering $p_0=\Phi(-\delta_0)$ and $p_1=\Phi(-\delta_1)$, the question becomes whether $\mu/\sigma=\delta_0$ or $\mu/\sigma=\delta_1$. In practice, the equalities shown in the hypothesized equations behave as inequalities in defining a band of indifference for the location of X_i (just as do the upper and lower indifference limits in the known variance case). Because it is not strictly defined which value of hypothesized δ is greater (or less) than the other, the hypotheses are shown as equalities for the general case.

If n observations x_i (i=1,2,...,n) have been taken, a sequential test of H_0 against H_1 may be obtained by considering only the distribution of the ratio $t=\frac{\bar{x}_n\sqrt{n}}{s_n}$. For H_0, t has the non-central t-distribution with (n-1) degrees of freedom and parameter δ_0, the probability density function [29] being

$$\phi(t|\delta_0,n)=\frac{\Gamma(n)\exp\left[\frac{1}{2}n(n-1)\delta_0{}^2\ \left(n-1+t^2\right)\right]}{2^{\frac{1}{2}(n-2)}\Gamma\left(\frac{1}{2}(n-1)\right)\sqrt{\pi(n-1)}}\left(\frac{n-1}{n-1+t^2}\right)^{\frac{1}{2}n}Hh_{n-1}\left(-\delta_0 u\right)$$

where $u=t\sqrt{n/(n-1+t^2)}$, and

$$Hh_n(x)=\int_0^{\infty}\left(\frac{z^n}{n!}\right)\exp\left[\frac{1}{2}(z+x)^2\right]dz$$

is the Hh-function tabulated by Airey [1].

For H_1, t has the probability density function $\phi(t|\delta_1,n)$, so that the likelihood ratio is given as

$$\lambda_n(t|\delta_0,\delta_1) = \frac{\phi(t|\delta_1,n)}{\phi(t|\delta_0,n)}.$$

As before, it is easier to run the sequential test procedure in terms of the natural logarithm of the likelihood ratio. Thus, the quantity $\ln\lambda_n(t|\delta_0,\delta_1)$ is calculated following each observation, and if

$$\ln\left(\frac{\beta}{1-\alpha}\right) < \ln\lambda_n(t|\delta_0,\delta_1) < \ln\left(\frac{1-\beta}{\alpha}\right) \tag{6}$$

another observation is taken. If the right inequality is broken, H_0 is rejected; if the left inequality is broken, H_0 is not rejected.

In consideration of the research problem, δ_0 and δ_1 may be viewed in terms of the indifference limits discussed in the case of known measurement distribution variance. That is, each delta may be considered a hypothesized distance, μ/σ, from the upper specification, representing a limit of "practical significance" for the batch disposition problem. It is assumed that the smaller of the two ratios represents a lower indifference limit, below which acceptance of the batch should occur. Conversely, the greater ratio represents the upper limit, beyond which the batch should be rejected. For solution of the problem, δ_0 and δ_1 are unrestricted in value. This implies that the null hypothesis may be either that the actual value of the batch is above the specification or that the value is below the specification.

In practice, it is easier to work with u than with t, since

$$u_n = \frac{\sum_{i=1}^{n}(x_i - \text{Spec.})}{\sqrt{\sum_{i=1}^{n}(x_i - \text{Spec.})^2}}.$$

This is easily calculable at each step of the sequential procedure by keeping track of the cumulative sum and cumulative sum of squares.

Making the substitution of u for t in calculating the log of the likelihood ratio gives

$$\ln\lambda_n = g_n(\delta_1 u_n) - g_n(\delta_0 u_n) - 0.5(\delta_1^2 - \delta_0^2)$$

where $$g_n(x) = 0.5x^2 + \ln[y_n(x)]$$

and $$y_n(x) = Hh_{n-1}(-x)/Hh_{n-1}(0).$$

As an alternative to the Hh function, the confluent hypergeometric function may be used in calculating $g_n(x)$. This substitution is presented in a practical procedure at the end of this section.

When the measurement error distribution mean is zero (no bias), the above equations may be used directly as presented. When the mean, μ_{me}, is non-zero, it is simply subtracted from each observation prior to the calculation of u. That is, generally, u is written as

$$u_n = \frac{\sum_{i=1}^{n}(x_i - \mu_{me} - \text{Spec.})}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_{me} - \text{Spec.})^2}}.$$

This is, of course, valid for any measurement error distribution mean and either specification limit. Given this general ratio, the sequential test proceeds using the Hh function or confluent hypergeometric function to calculate the likelihood ratio following each observation. The likelihood ratio is tested against the probability bounds as shown in inequality (6).

In practice, the non-central t-distribution probabilities which are required for the likelihood ratios may be calculated using the Hh function shown, above, or the confluent hypergeometric function. The Hh function tables of Airey [1] are limiting in sample size (n) and achievement of precision in interpolation due to the range of table values. A procedure which utilizes the confluent hypergeometric function [56] in treating the unknown variance case follows. This is the logic which is implemented in the FORTRAN computer program which is discussed in Chapter Four and presented in Appendix A.

Unknown Variance Procedure

1) Calculate decision limits $\ln A = \frac{1-\beta}{\alpha}$ and $\ln B = \frac{\beta}{1-\alpha}$

2) Observe x_n.

3) Calculate

$$u_n = \frac{\sum_{i=1}^{n}(x_i - \mu_{me} - \text{Spec.})}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_{me} - \text{Spec.})^2}}$$

4) Calculate the log of the likelihood ratio,

$$\ln\lambda_n = g_n(\delta_1 u_n) - g_n(\delta_0 u_n) - 0.5(\delta_1^2 - \delta_0^2)$$

where

$$g_n(x) = \ln\left\{M\left(\frac{n}{2},\frac{1}{2},\frac{1}{2}x^2\right) + \sqrt{2}xM\left(\frac{1}{2}(n+1),\frac{3}{2},\frac{1}{2}x^2\right)\frac{\Gamma\left(\frac{1}{2}(n+1)\right)}{\Gamma\left(\frac{1}{2}n\right)}\right.$$

and $M(\alpha,\gamma,x)=\sum_{j=0}^{\infty}\frac{\Gamma(\gamma)\Gamma(\alpha+j)x^j}{\Gamma(\alpha)\Gamma(\gamma+j)j!}$ is the confluent hypergeometric which is closely related to the Hh function.

5) Compare the log of the likelihood ratio to the decision limits found in step 1. If $\ln\lambda_n<\ln B$, don't reject H_0. If $\delta_0<\delta_1$, this leads to acceptance of the batch; if $\delta_1<\delta_0$, it leads to batch rejection. If $\ln\lambda_n>\ln A$, reject H_0. If $\delta_0<\delta_1$, exceeding the upper limit leads to rejection of the batch; if $\delta_1<\delta_0$, it leads to acceptance. If the sequential ratio falls between the two decision limits, repeat steps 2-5.

3.5 PROGRAM DESCRIPTION

The comprehensive FORTRAN program which implements the statistical and economic theories as presented in the current chapter and Chapter 5 is presented in Appendix A. Both models are accessible from a common main module (shown below as MAIN). The FORTRAN subroutines which implement the statistical sequential theory as discussed in this chapter are summarized in the following paragraphs. Each heading represents the actual subroutine name as given in the FORTRAN code (without arguments). Further information on program operation and a description of subroutine arguments is contained within the body of the code (Appendix A) in the form of program comments.

The general hierarchy of the program modules is shown in Figure 3.2. There are five subroutine branches which are accessible from the main program module. Subroutine names are shown in parentheses in the figure. The interaction of these subroutines with the remainder of the program code is further-detailed in the subroutine summaries which follow (statistical branch) and in Chapter 5 (economic branch).

Verification of computer program logic is accomplished through redundant runs using hand calculations, FORTRAN simulators (Appendix D), Microsoft Excel 5.0[1] and Mathcad 4.0.[2]

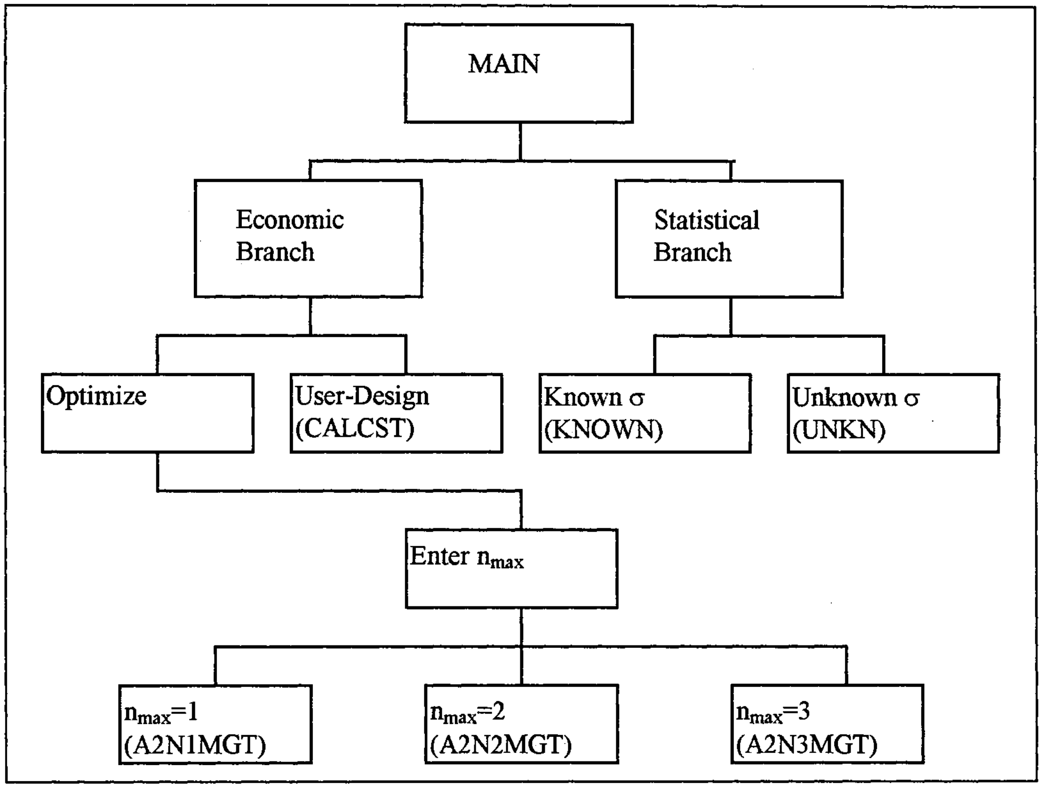

Figure 3.2 Hierarchy of Subroutines, Comprehensive FORTRAN Program.

MAIN

Provides access to the statistical and economic sequential models through menu options. Calls subroutines: CALCST, A2N1MGT, A2N2MGT, A2N3MGT, KNOWN, UNKN.

---

[1] *Microsoft Office Professional v. 4.3* (1993), Microsoft Corporation, USA.

[2] *Mathcad 4.0 User's Guide Windows Version* (1993), Mathsoft, Inc, Cambridge, MA.

KNOWN

Implements the known variance statistical sequential model as described in Section 3.3. Accepts inspection information and measurement observations and makes a disposition decision based on Wald SPRT theory. Calls subroutines: none. Called from: MAIN.

UNKN

Implements the unknown variance statistical sequential model as described in Section 3.4. Accepts inspection information and measurement observations and makes a disposition decision based on likelihood ratio theory. Calls subroutines: SUMCH, GAMN. Called from: MAIN.

SUMCH

Evaluates the confluent hypergeometric function ($M(\alpha,\gamma,x) = \sum_{j=0}^{\infty} \frac{\Gamma(\gamma)\Gamma(\alpha+j)x^j}{\Gamma(\alpha)\Gamma(\gamma+j)j!}$) for the purpose of finding the non-central t-distribution likelihood ratio. Calls subroutines: none. Called from: UNKN.

GAMN

Evaluates the gamma function for use in the likelihood ratio calculation (unknown variance case). Calls subroutines: none. Called from: UNKN.

3.6 SUMMARY

The statistical treatment of the problem of homogeneous batch disposition based on a single variable characteristic subject to measurement error involves application of sequential statistics. The Sequential Probability Ratio Test (SPRT) requires that observations are accumulated until enough evidence is available to reach a decision based on pre-specified risk tolerances, α and β. The decision ratio which is calculated following each observation is the likelihood ratio. Economic consequences of the disposition decision are not explicitly considered.

The case of known measurement system variance (distribution normal) can be treated directly with Wald's SPRT for simple hypotheses, previously not applied to the research problem. The simple case requires the definition of indifference limits around the specification limit. Beyond these indifference limits, decision errors are determined (by the inspector or plan designer) to be of practical consequence. Observations are repeated until the likelihood ratio falls outside the statistical decision limits.

When the measurement system variance is unknown (normality assumed), the problem is one of composite hypotheses and is termed a sequential t-test. Previous attempts to apply simple hypothesis SPRT theory to the case of unknown variance have used tables and approximations in order to handle the non-central t-distribution probabilities which compose the likelihood ratio. The research solution implements simple hypothesis SPRT theory in exact treatment of the problem; the non-central t-probability likelihood ratios are calculated directly using a computer solution presented later in this paper. The solution treats cases of both upper and lower specification limits.

Practical implementation of the case of unknown variance requires calculation of non-central t-distribution probabilities in order to obtain the likelihood ratio after each iteration. These probabilities are complicated and require the Hh function, tabled by Airey [1], or the confluent hypergeometric function [56]. As in the simple hypothesis situation, the likelihood ratio is tested against statistical decision limits following each observation. The unknown variance case is general as to the nature of the specification limit (upper or lower).

CHAPTER 4

OPERATION OF THE INTERACTIVE COMPUTER ROUTINES FOR THE STATISTICAL SOLUTION

4.1 INTRODUCTION

This chapter details the operation of the interactive computer program modules which implement the solutions of the known and unknown measurement variance statistical problems. The solution methodology is presented in the preceding chapter. The actual FORTRAN program containing these modules (composed and executed on an IBM-compatible personal computer using Microsoft FORTRAN version 5.1) appears in Appendix A.

The computer routines are interactive and prompt the user for the required input parameters. Sequential batch data is entered as it becomes available via the computer keyboard (code may be easily modified to accept entry directly from a measurement gage through a computer input port). Prior to beginning data entry, the operator is asked to specify a maximum number of measurement iterations; if a statistical decision is not determined prior to reaching this operator-designated maximum, the current log of the likelihood ratio is displayed and data entry stops. If this maximum is reached, it is left to the user to determine batch disposition. A general rule for making a decision upon reaching the maximum number of iterations is provided in references [28] and [59]. This sampling situation, known as the truncated SPRT, is described in Chapter 3. The disposition decision recommended by the truncated plan is displayed for the user's information.

Error checks are performed for user-provided input parameters. All input values are presented for operator verification prior to beginning execution of the sequential data collection.

4.2 KNOWN MEASUREMENT VARIANCE

Program Operation

As with other program modules presented in later chapters, access to the statistical routines is provided through a common main menu interface. The main menu appears as follows:

```
Sequential Testing Program

Please select one of the following options:

  1  Economic Testing
      *  Plan Optimization
      *  Expected Cost Calculation

  2  Statistical Testing
      *  Known Measurement Error Variance
      *  Unknown Measurement Error Variance

  3  Exit Program

2
```

The user has entered a “2” in order to access the statistical routines. If anything other than the valid options of 1-3 is entered in response to the menu, the following message appears:

```
**** Invalid Entry.  Please Reenter. ****
```

In response to the selection of the second option from the main menu, the statistical plan menu is presented:

```
Statistical Testing Plans

Please select one of the following options:

  1  Known Measurement Error Variance
      Sequential Data Entry and Batch Disposition

  2  Unknown Measurement Error Variance
      Sequential Data Entry and Batch Disposition

  3  Return to Main Menu

1
```

The user entry of “1” begins execution of the known variance routine for statistical testing. An invalid entry in response to this menu brings up the error message which was previously presented.

The known measurement variance option of the FORTRAN program begins by requesting the iteration maximum. This is the number of individual measurements which the program user is willing to make in order to reach a disposition decision (a decision may be reached prior to this maximum).

```
What is the maximum number of iterations which
you wish to make (cannot exceed 50)?

6
```

The user has entered a "6", indicating that the measurement process should not exceed 6 iterations. A response that does not fall in the range of 1-50 produces an error message and the entry prompt reappears on the screen. This maximum iteration value is utilized to set up a DO LOOP for data entry. If a statistical decision is reached prior to reaching the maximum, the user is informed of the appropriate batch disposition and data entry stops.

The first inspection system parameter which is entered is the measurement error standard deviation:

```
Enter the standard deviation of the measurement
error distribution.

.5
```

The value of the standard deviation may not be less than zero.

The module next requests the value of the measurement error distribution mean. This is often referred to as the measurement bias and follows the following sign convention: if the observed reading is greater than the true value by the fixed error mean, the bias is positive; if the observed reading is consistently less than the actual value by the amount of the bias, it is considered negative. The prompt appears with a reminder of this convention:

```
Enter the measurement error bias.
Sign Convention: If the instrument reads higher
than the true value, this bias should be positive.

1.
```

The user has entered a value of "1.". The program routine will subtract this fixed bias from each observation entered prior to calculating probabilities for the likelihood ratio.

The next three pieces of information specify the inspection system decision limits. In addition to the specification limit (upper or lower), the user enters the upper and lower indifference limits (previously explained) around the specification limit. Either one of these limits may coincide with the actual specification limit, but all three limits cannot be the same. An error check is performed to ensure that the indifference limits are appropriately located relative to the specification limit. The first prompt is:

```
Enter the Specification Limit.

102
```

Because it is the first limit which is entered, the value of the specification is unrestricted. The user has entered a value of "102". In addition to specifying the parameter value, the user must also indicate if this specification is an upper or lower limit. The program prompt is

```
Is this an Upper (1) or Lower (2) Spec?

Enter 1 or 2.

1
```

The user has entered a "1" for an upper specification. As explained, the indifference limits must be located in proper relation to this specification.

Following entry of the specification, the next prompt appears:

```
Enter the Acceptance Indifference Limit.
(Beyond which acceptance is preferred)

99
```

In the above example, the user has entered a value of "99". An entered value greater than the upper specification produces an error message and a request for reentry. The final limit prompt then appears:

```
Enter the Rejection Indifference Limit.
(Beyond which rejection is preferred)

102
```

In this case, an entered value less than the upper specification brings up an error and the user is prompted for reentry of all three of the limits (specification, acceptance indifference and rejection indifference). The user has entered "102" (also the upper specification limit) indicating an aversion to false acceptance.

The next items of information required by the program module are the statistical risks of incorrect batch disposition which the program user is willing to incur. In coming to a decision concerning the conformity of the batch being inspected, the possibility exists for two types of error. A Type I error is said to have occurred if the lot is rejected as non-conforming when it actually is acceptable in regard to the characteristic of interest. The acceptable risk level associated with a Type I error is given as alpha (α). The other error, Type II, involves acceptance of the batch when the batch characteristic does not conform to the specification. Beta (β) represents the acceptable risk level associated with the occurrence of a Type II error. The first prompt given is

```
Enter Alpha, Type I Error Probability (0 to 1).

.1
```

An entry value which is out of range brings an error message. The request for Beta follows alpha entry:

```
Enter Beta, Type II Error Probability (0 to 1).
.01
```

Again, an entry that is not within the range 0.0-1.0 produces an error message and a prompt for reentry of the beta level. The relatively low value entered by the user in the illustration (".01") indicates that the penalty for a false batch acceptance is more severe than that for a false batch rejection.

Following this last parameter entry, the program values are displayed for review. The user is given the opportunity to change any of the parameters, although only one parameter may be changed at a time. In the following illustration, the user takes the opportunity to modify the acceptance indifference limit.

```
1 Error Standard          .50
  Deviation=
2 Upper                 102.0
  Specification=            0
3 Accept                99.00
  Indifference
  Limit=
4 Reject                102.0
  Indifference              0
  Limit=
5 Alpha=                  .10
6 Beta=                   .01

Is the above information correct?
Enter to accept, or # of parameter to reenter.

3
```

The program then prompts for reentry of the parameter specified by the user:

```
Enter Accept Indifference Limit
```

```
100
```

The user has entered "100" as a correction to the previously entered value of the acceptance indifference limit. The same error checks which were previously described are also performed for any parameter modifications. Following any corrections, the parameters are displayed in summary form and corrections may again be made.

Once all of the parameter information is entered correctly (per the user), data entry begins:

```
Enter measurement observation #     1
102.
Enter measurement observation #     2
101.2
```

Observation entry continues until a disposition decision is reached or the maximum number of iterations expires. After entry of the second observation in the example, above, the following message appears:

```
Ln of likelihood ratio,     -6.4000, less than
ln of B,      -4.4998.
********  Accept Batch  ********
```

In this particular example, an "accept" disposition decision was reached in two iterations. A similar message appears with the corresponding lower decision limit (ln of A) if the data leads to a "reject" decision within the maximum iterations allowed.

In the event that the user-specified maximum number of iterations is reached without leading to an appropriate disposition decision, the following message appears (values shown correspond to entry values of 102, 102, 101.9, 102, 101.8, 102.1):

```
Maximum number of iterations reached.
Log of likelihood ratio = -1.600000
Acceptance limit = -4.499810
Rejection Limit = 2.292535
Wald Truncation Rule calls for Acceptance.
```

This is the truncated SPRT decision rule as proposed by Wald [59] and discussed in Chapter 3. If the data suggests that the batch should be rejected based on the relative location of the log of the likelihood ratio (following maximum observations), a similar message appears so-indicating.

Display of these values with the decision dictated by the Wald truncation rule is intended to give the user the relative location of the likelihood ratio for an informed disposition decision. However, this message indicates that the program was unable to reach a batch disposition decision based on the Wald SPRT theory within the desired number of measurement iterations.

To return to the main menu from the statistical testing menu, the third option ("3 Return to Main Menu") is selected. An entry of "3" at the main menu prompt causes termination of the entire sequential testing program.

4.3 UNKNOWN MEASUREMENT VARIANCE

Program Operation

As in the known variance case discussed in section 4.2, the program module for the statistical solution case of unknown measurement error variance is accessed from the main menu. At the secondary menu, option number "2" is selected by the user:

```
Statistical Testing Plans

Please select one of the following options:

  1  Known Measurement Error Variance
      Sequential Data Entry and Batch Disposition

  2  Unknown Measurement Error Variance
      Sequential Data Entry and Batch Disposition

  3  Return to Main Menu

2
```

This begins execution of the unknown variance statistical routine. The first input requested by this program option is the maximum number of iterations (observations) which the program user is willing to make.

```
What is the maximum number of iterations which
you wish to make?  (cannot exceed 50)

4
```

In this case, the user has indicated that a disposition decision is desired within four iterations. If the number entered is less than two or greater than fifty, the entry is flagged as invalid and the user must try again. Unlike the known variance decision method, this scenario does not allow a statistical disposition decision with a single trial.

The routine then begins a series of prompts which request entry of measurement system parameters. The first parameter required is the specification limit:

```
Enter the Specification Limit.
```

```
102
```

In this example, the user has entered a value of "102" for the specification. This program module does not require that the user distinguish between upper and lower specification limits. This is because the solution is general and everything is left in terms of testing a null hypothesis against an alternative, regardless of relative location to the specification limit. Relation to the specification is indicated by the signs of the respective ratios (mean to standard deviation) which constitute the tested hypotheses. The program prompts for the null hypothesis ratio:

```
Enter the ratio of mean/standard deviation to be
tested for the null hypothesis.
```

```
0
```

The above prompt refers to the ratio δ_0 as defined in the previous chapter. The value entered in the example, "0", indicates that the hypothesized value includes the upper specification as the mean. There is no sign restriction on this value; a negative value is used for a hypothesized value which is to the left of the specification limit. A similar prompt then appears which addresses the mean to standard deviation ratio to be tested for the alternative hypothesis:

```
Enter the ratio of mean/standard deviation to be
tested for the alternative hypothesis.
```

```
-1
```

In terms of the theoretical description presented in the previous chapter, the ratio shown above is δ_1. Because the example entry is negative ("-1"), it defines a hypothesized position below the specification limit and the previously defined δ_0.

```
Enter Alpha, acceptable Type I error probability,
associated with a true null hypothesis
(0 to 1).

.1
```

Alpha, the type I error probability, is as previously defined and must be between 0.0 and 1.0. The last parameter entry follows:

```
Enter Beta, acceptable Type II error probability,
associated with a true alternative hypothesis
(0 to 1).

.01
```

This is the acceptable Type II error risk. As in the known variance example, the user has entered a beta value of ".01", indicating a relatively strong aversion to a false batch rejection.

The module next requests the value of the measurement error distribution mean (bias). The sign convention described in section 4.2 applies in this case, and is displayed for information purposes:

```
Enter the measurement error bias.
Sign Convention: If the instrument reads higher
than the true value, this bias should be positive.

0
```

The user-entered value of "0" indicates that there is no consistent measurement offset which must be accounted for in the inspection system.

After all parameters are entered, they are displayed for user review and verification:

```
1 Specification Limit=       102.
                               00
2 Null Hyp. Ratio of          0.00
  Mean/Std. Dev.=
3 Alt. Hyp. Ratio of             -
  Mean/Std. Dev.=             1.00
4 Alpha=                       .10
5 Beta=                        .01
6 Measurement Bias=           0.00

Is the above information correct?
Enter to accept, or # of parameter to reenter.
```

By pressing the carriage return (ENTER), the user indicates that all of the information displayed is correct. If any of the information is changed, it is subjected to the original error checks and all data is, again, displayed for user approval.

Upon acceptance of all input parameters, data entry begins. Recall that, in the example, the user has indicated that no more than four batch measurement iterations are allowed.

```
Enter measurement observation #    1

102.6

Enter measurement observation #    2

102.2

Enter measurement observation #    3

102.5

Enter measurement observation #    4

102.6
```

Observation entry continues until a disposition decision is reached or the maximum number of iterations is attained. In the example, a statistical decision is achieved after the fourth observation, and appears with the following message:

```
Ln of Likelihood ratio,     -4.86, less than
ln of B,     -4.50.
************************************
****** Reject Null Hypothesis ******
************************************
```

In the event that the maximum number of iterations is reached without coming to a decision concerning the batch, the statistical information is displayed for the user's consideration:

```
No decision reached.
Log of likelihood ratio=        1.397299
Accept limit=         2.292535
Reject limit=         -4.499810
```

The example information shown, above, is consistent with the following sequential data entry: 102.6, 100.4, 100.1, 100.4. If the user applies the truncation rule for simple hypotheses which was previously presented, the null hypothesis will fail to be rejected.

To return to the main menu from the statistical testing menu, the third option ("3 Return to Main Menu") is selected. An entry of "3" at the main menu prompt causes termination of the entire sequential testing program.

4.4 SUMMARY

The interactive FORTRAN program routines for the known variance cases of the economic solution proceed in similar manners. Both program solutions are accessed from the main menu which manages the statistical and economic solutions. In each statistical case, the user provides inspection system parameters and specifies the maximum number of measurement iterations which are to be taken. The program then prompts sequentially for measurement data. If a statistical disposition decision is not achieved within the user-specified number of iterations, the current decision ratio and decision limits are provided for the user's consideration. The Wald SPRT solution, used in the case of known measurement system variance, uses a truncated SPRT rule in making a disposition recommendation upon reaching the observation maximum. Error checks are performed on all input parameters to ensure consistency with theoretical solution constraints.

The procedures of the computer program are consistent with the sequential statistical solutions as presented in the previous chapter.

CHAPTER 5

THEORETICAL DEVELOPMENT OF THE ECONOMIC SOLUTION TO THE PROBLEM

5.1 INTRODUCTION

The statistical sequential assessment of a batch of known measurement error variance subjectively incorporates the consequences involved with the incorrect disposition of product. Risks are included in the statistical model through the quantification of alpha and beta and through the establishment of a subjective indifference zone for batch assessment. All of these parameters, which attempt to quantify the risks involved with incorrect dispositions, are strictly subjective as established by the plan designer or inspector. Although cost may be a consideration in designating the risk levels associated with the statistical case, cost components are implicit and are not required for development and solution of the problem.

The economic sequential assessment of a homogeneous batch of product involves objective quantification of the costs associated with the inspection and disposition of the product. These costs are explicit and reflect the true risks and cost consequences associated with an incorrect batch disposition decision. Predictably, these costs of quality are not typically known and may be very difficult to estimate. The economic model requires that these costs be assessed and used as the basis for the inspection system design.

There are two costs to consider when executing an acceptance sampling plan for lot or batch disposition. These costs are identified and discussed in a writing by Case and Keats [8] on attributes sampling plans. The first cost is that incurred while gathering data by which to make a decision. In the case of a sequential plan, this cost may be viewed as an iteration cost, involving operator labor, gage depreciation, gage cleaning or resetting, and any other actions associated with performing a single measurement of the batch characteristic. The second cost is incurred through disposition of the lot or batch, as indicated by the sampling plan. In the current problem, only two possible disposition decisions are considered: acceptance of the batch and rejection of the batch.

When subjecting a batch to any acceptance sampling plan, two errors are possible in the disposition of a specific batch of product. A Type I error is committed when a conforming batch is classified as non-conforming and rejected as unsuitable for use. In the specific problem treated in this paper, a Type I error occurs when the true batch value does not exceed the upper specification limit, U, and the batch is erroneously rejected. A Type II error is committed when a non-conforming batch is classified as conforming and accepted for use. Specifically, a Type II error occurs when the true batch value exceeds U, and the batch is wrongfully deemed acceptable.

Both of these errors, Types I and II, have associated costs. Typically, costs incurred through the commission of a false acceptance far exceed those associated with a false rejection. In the event that a conforming batch of product is rejected (Type I error), the cost incurred is only the cost of the batch of product. That is, upon rejection, the batch of product is scrapped at current worth and no other costs are incurred. If the process is such that the batch of product is reworked, there may also be additional costs added to the product prior to being submitted to further testing. In the case of a Type II error and the associated acceptance of a non-conforming batch of product, costs are often greater by orders of magnitude, but are also very difficult to identify and quantify. Costs to consider are those due to loss of customer goodwill, warranties, returns, repairs, lawsuits resulting from non-conformities, loss of return customers and, in the extreme case, loss of life due to the non-conformity. Non-conformities passed on through the commission of a Type II error continue through the manufacturing process and run the ultimate risk of reaching the customer if not detected prior to final inspection.

In assessing the expected total cost associated with a particular sequential inspection plan, consideration is given to the history of similar batches which have been inspected prior to the current batch. Prior history of batch inspection is incorporated into the economic model through Bayesian decision theory methods. Past batch history is used to predict batch quality prior to making observations on the current batch. Each measurement iteration performed on the batch is used to update the past history and make further predictions about the batch conformity to specifications and corresponding expected costs associated with erroneous rejection and acceptance.

The model which is developed assumes that the batch inspector (or designing engineer) perceives a practical limit to the number of measurement iterations which he/she is willing to conduct. This is an assumption based on the practical aspects of product inspection and the understandable limits of patience and perseverance of a product inspector. In practical applications which normally utilize a single measurement observation for purposes of homogeneous batch disposition, an upper limit of three iterations seems reasonable and realistic. An explicit maximum for the number of observations made is also assumed in the statistical solution to the problem and implemented through truncation of the SPRT (see Chapter 3).

5.2 NOTATION

In order to facilitate model development, the following notation is defined:

n_{max} = the maximum number of iterations that the inspector is willing to conduct on a given batch

x_i = the ith observation/iteration on the batch characteristic, μ ($i \le n_{max}$)

$\bar{x}_n$=average of n observations, $\frac{1}{n}\sum_{i=1}^{n} x_i$

U = upper specification, differentiating acceptable and non-conforming product

L = lower specification, differentiating acceptable and non-conforming product

$C_{n,L}$ = the lower cutoff limit for use after n iterations

$C_{n,H}$ = the upper cutoff limit for use after n iterations

$C_{n_{max}}$ = the single cutoff limit for n_{max} ($C_{n_{max}} = C_{n_{max},L} = C_{n_{max},H}$)

S = the cost per iteration for measurement inspection

A = the cost of accepting a batch which is actually non-conforming

R = the cost of rejecting a batch which is actually conforming

μ = unknown value of the batch characteristic of interest

τ_0 = the standard deviation of the prior distribution of batch values

τ_i = the standard deviation of the updated prior distribution of batch values prior to the i+1 iteration

θ_0 = the mean of the prior distribution of batch values

θ_i = the mean of the updated distribution of batch values prior to the i+1 iteration

σ_{me} = the standard deviation of the measurement error distribution

μ_{me} = the mean of the measurement error distribution (bias)

$f(\mu)$ = continuous prior distribution of batch values ~ $N(\theta_0,\tau_0^2)$

$l(x_i|\mu)$ = sampling distribution describing the probability of observing a measurement, x_i, given an actual value, $\mu \sim N(\mu+\mu_{me},\sigma_e^2)$

$g(x_1)$ = marginal (or unconditional) distribution describing the probability of observing an observation, x_1, on the first iteration $\sim N(\theta_0+\mu_{me},\tau_0^2+\sigma_e^2)$

$g(x_n|x_1,\ldots,x_{n-1})$ = conditional distribution of x_n, describing the probability of drawing an observation x_n, given all prior observations on the current batch $\sim N(\theta_{n-1}+\mu_{me},\tau_{n-1}^2+\sigma_e^2)$

$h(\mu|x_1,\ldots,x_n)$ = posterior distribution describing the probability of the batch having an actual value, μ, given that observations x_1 through x_n have been observed $\sim N(\theta_n,\tau_n^2)$

5.3 ECONOMIC MODEL DEVELOPMENT

Sampling Procedure

The economic model is based on inspection parameters which dictate the subsequent action following a given measurement iteration. At any iteration prior to the designated maximum in the sequential measurement procedure, any of three actions may be taken. These actions, as described previously in the development of the statistical decision case, are: 1) Batch acceptance, 2) Batch rejection and, 3) Continuation of the sequential measurement procedure. Whereas the statistical model provides decision limits which are based on subjective risk levels defined by the inspector or plan designer, the economic model optimizes the plan decision limits based on explicit costs associated with disposition decision and prior batch history. The decision limits which are utilized in the economic analysis are termed cutoff values as defined in the following discussion.

The model which is developed is general for the case of a single upper specification limit. Extension to the situation of a lower specification follows readily from the upper specification model development; a change in signs for the inequalities when testing the observation mean against the appropriate cutoff values represents the only procedural change in executing the inspection plan. Practical solution of the lower limit problem is easily inferred through symmetry from a complementary upper solution. The computer solution which is subsequently presented accepts either an upper or lower limit in designing an optimum economic solution to the single specification problem.

The optimal inspection plan is dependent on the maximum number of measurement iterations, n_{max}, designated by the inspector. If the measurement sequence is carried to n_{max} (implying all previous measurements dictated an additional iteration), a disposition decision is required upon the n_{max}th iteration. Because a decision will be made based on the single iteration, a cutoff value of $C_{n_{max}} = C_{n_{max},L} = C_{n_{max},H}$ is defined as the only decision limit. At n_{max}, if

$$\bar{x}_{n_{max}} > C_{n_{max}}$$

then the batch is rejected. Alternatively, if

$$\bar{x}_{n_{max}} \leq C_{n_{max}}$$

the batch is accepted. The cutoff value $C_{n_{max}}$ is determined such that the total cost equation is minimized.

Following any n iterations, prior to n_{max}, $\bar{x}_n$ may dictate acceptance, rejection or disposition deference through continuation of the iteration process. In this case, two cutoff values ($C_{n,L}$ and $C_{n,H}$) are specified, such that, for

$$\bar{x}_n \leq C_{n,L}$$

the batch is accepted following the nth iteration. If

$$\bar{x}_n > C_{n,H}$$

the batch is rejected following the nth iteration. In the case that

$$C_{i,L} < \bar{x}_i \leq C_{i,H}$$

no disposition decision is made following the nth iteration and the procedure continues with an additional iteration (at n+1).

The sampling procedure for any $n_{max} \geq 1$ in testing against an upper specification limit is summarized in Figure 5.1. Implementation of the logic requires specification of all known measurement system parameters and the set of decision cutoff values $(C_{1,L}, C_{1,H}, \ldots, C_{n_{max}})$ to be utilized for batch disposition.

In the case of a single, lower specification limit, only the directions of the inequalities change in testing the observation mean against the cutoff limits for the plan. That is, an $\bar{x}_i$ which falls under the lower cutoff limit dictates rejection, rather than acceptance of the batch. Acceptance is indicated if the observation mean exceeds the upper cutoff limit.

For any designated n_{max}, the total cost of the inspection sequence is a function of several variables:

$$TC(n_{max}; \mu, x_1, \ldots, x_{n_{max}}; C_{1,L}, C_{1,H}, \ldots, C_{n_{max}-1,L}, C_{n_{max}-1,H}; C_{n_{max}}; \text{U or L})$$

Some of these parameters (n_{max} and cutoff values) are decision variables under the control of the user or designer. Others are random variables $(\mu; x_1, \ldots, x_{n_{max}})$ over which the user has no control. As shown in subsequent total cost equation developments, for any designated n_{max}, the total cost equation may be expressed as the sum of $(n_{max})*3$ terms.

Impact of Measurement Bias on Plan Design

The effect of bias on the sampling plan design depends on the nature of the prior distribution. It is assumed that all historical data represents actual batch values and all

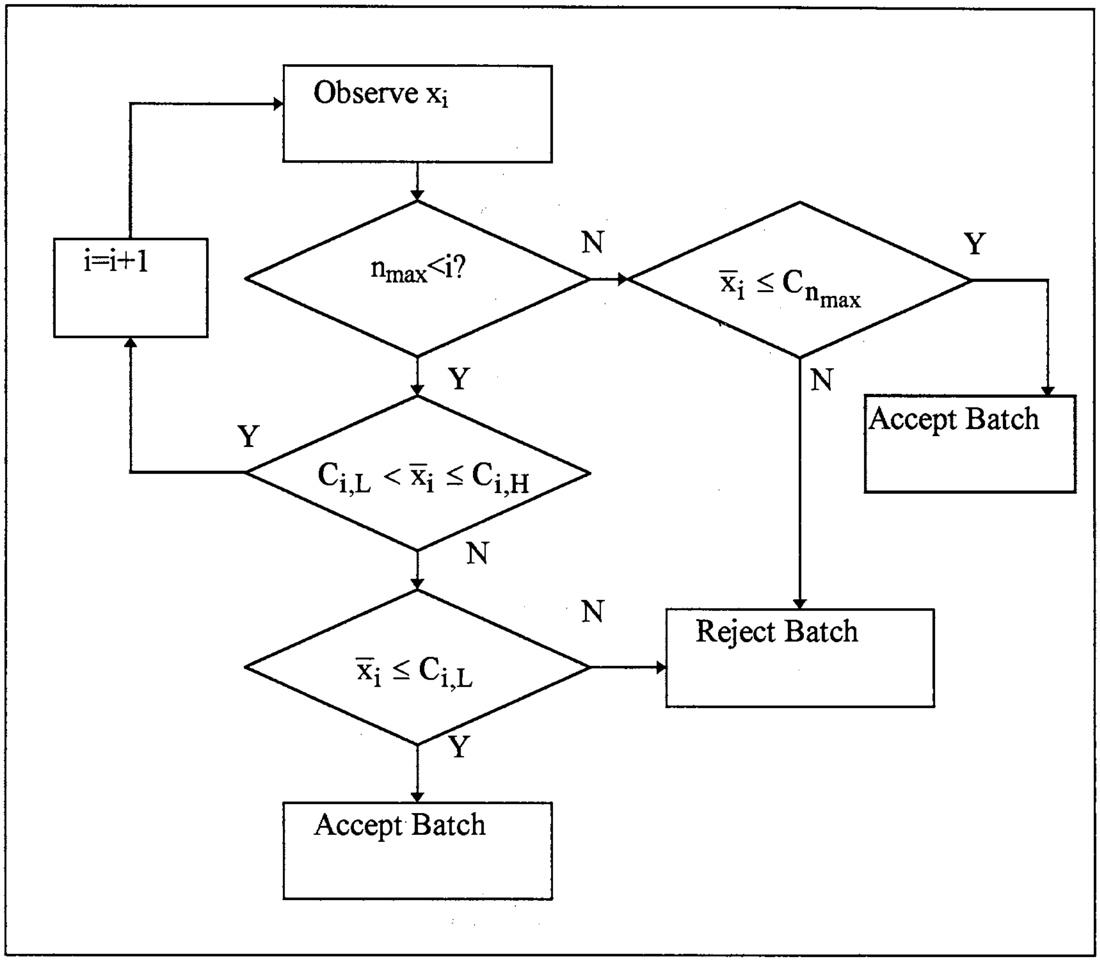

Figure 5.1. Flowchart of Economic Sequential Sampling Plan, Upper Specification Limit.

measurement system bias has been removed from the prior distribution of values. Under this assumption, compensation for measurement system bias must be made in one of two ways:

1) The cutoff values are chosen based on observations with zero bias and each measurement, x_i, is bias-adjusted prior to using the decision system.
2) The cutoff values are chosen to include the bias adjustment and each observation, x_i, is used directly in cutoff comparisons to determine batch disposition.

To free the system operator of any extraneous calculations, the second compensation method is most desirable. That is, the operator wishes to use the observations directly in calculating the mean value for comparison to the applicable cutoff value(s). In order to build the bias into the decision cutoff limits, it is not necessary to carry the measurement error mean completely through the economic optimization and system design. Because the bias is a constant offset from the actual value, the system may be designed with zero bias, and the measurement error mean simply added to the chosen cutoff limits following system design. This is the approach of the theoretical development included in this chapter.

The FORTRAN computer routines which make application of the theoretical developments also make bias adjustments following a zero-bias system design. The zero-bias cutoffs are provided as program output in addition to the cutoffs which include the bias as provided as input by the operator. By providing the zero-bias system parameters, any change in the measurement error offset is easily incorporated into the sampling plan design without running the optimization program. The computer code for the FORTRAN programs described in this chapter is provided in Appendices A and B.

Distributional Properties

It is assumed that the unknown batch characteristic, μ, follows a normal distribution according to historical batch information available. In terms of the prior distribution parameters, $f(\mu)$ is distributed as Normal with mean θ_0 and variance τ_0^2. Additionally, the measurement error is assumed to be distributed as Normal with mean μ_{me} and variance σ_{me}^2. It follows that the sampling distribution of the first observation on μ, $l(x_1|\mu)$, is centered at mean $(\mu+\mu_{me})$ with variance σ_{me}^2. In order to evaluate the other distributions involved in the sequential sampling plan, the following Bayesian equation is given:

$J(x_1,\mu)=l(x_1|\mu)f(\mu)=g(x_1)h(\mu|x_1)$

or

Joint Dist.	= Sampling Dist.	x	Prior	= Marginal Dist. x Dist.	Posterior Dist.

To facilitate the distributional development, the measurement error bias is neglected (assumed zero). Because each measurement observation is offset from the actual batch value by this constant, known value, the bias may be omitted and reintroduced at a later stage without loss of model integrity.

The joint distribution (μ_{me}=0) is given as

$$J(x_1,\mu)=\frac{1}{2\pi\cdot\sigma_{me}\tau_0}\exp\left\{-\frac{1}{2}\left[\frac{\left(\mu-\theta_0\right)^2}{\tau_0{}^2}+\frac{\left(x_1-\mu\right)^2}{\sigma_{me}{}^2}\right]\right\}.$$

This is a bivariate normal distribution. To find the marginal distribution of x_1, shown above as $g(x_1)$, first define

$$\rho=\frac{1}{\tau_0{}^2}+\frac{1}{\sigma_{me}{}^2}=\frac{\tau_0{}^2+\sigma_{me}{}^2}{\tau_0{}^2\cdot\sigma_{me}{}^2}$$

and complete the squares for the exponential portion of the joint distribution as follows [8]:

$$\begin{aligned}\frac{1}{2}\left[\frac{(\mu-\theta_0)^2}{\tau_0{}^2}+\frac{(x_1-\mu)^2}{\sigma_{me}{}^2}\right]&\\
&=\frac{1}{2}\left[\left(\frac{1}{\tau_0{}^2}+\frac{1}{\sigma_{me}{}^2}\right)\mu^2-2\left(\frac{\theta_0}{\tau_0{}^2}+\frac{x_1}{\sigma_{me}{}^2}\right)\mu+\left(\frac{\theta_0{}^2}{\tau_0{}^2}+\frac{x_1{}^2}{\sigma_{me}{}^2}\right)\right]\\
&=\frac{1}{2}\rho\left[\mu^2-\frac{2}{\rho}\left(\frac{\theta_0}{\tau_0{}^2}+\frac{x_1}{\sigma_{me}{}^2}\right)\mu\right]+\frac{1}{2}\left(\frac{\theta_0{}^2}{\tau_0{}^2}+\frac{x_1{}^2}{\sigma_{me}{}^2}\right)\end{aligned}$$

$$=\frac{1}{2}\rho\left[\mu-\frac{1}{\rho}\left(\frac{\theta_0}{\tau_0^2}+\frac{x_1}{\sigma_{me}^2}\right)\right]^2-\frac{1}{2\rho}\left(\frac{\theta_0}{\tau_0^2}+\frac{x_1}{\sigma_{me}^2}\right)^2+\frac{1}{2}\left(\frac{\theta_0^2}{\tau_0^2}+\frac{x_1^2}{\sigma_{me}^2}\right)$$

$$=\frac{1}{2}\rho\left[\mu-\frac{1}{\rho}\left(\frac{\theta_0}{\tau_0^2}+\frac{x_1}{\sigma_{me}^2}\right)\right]^2+\frac{(\theta_0-x_1)^2}{2\left(\tau_0^2+\sigma_{me}^2\right)}.$$

Therefore,

$$J(x_1,\mu)=\frac{1}{2\pi\cdot\sigma_{me}\tau_0}\exp\left\{-\frac{1}{2}\rho\left[\mu-\frac{1}{\rho}\left(\frac{\theta_0}{\tau_0^2}+\frac{x_1}{\sigma_{me}^2}\right)\right]^2\right\}\exp\left\{-\frac{(\theta_0-x_1)^2}{2\left(\tau_0^2+\sigma_{me}^2\right)}\right\}.$$

The marginal distribution of x_1, shown above as $g(x_1)$, is found by the definition:

$$g(x_1)=\int_{-\infty}^{\infty}J(x_1,\mu)d\mu=\frac{1}{\sqrt{2\pi\rho}\sigma_{me}\tau_0}\exp\left\{-\frac{(\theta_0-x_1)^2}{2\left(\tau_0^2+\sigma_{me}^2\right)}\right\}.$$

By examination, the marginal distribution of x_1 is $N(\theta_0,\tau_0^2+\sigma_{me}^2)$.

The posterior distribution, $h(\mu|x_1)$, is given as

$$h(\mu|x_1)=\frac{J(x_1,\mu)}{g(x_1)}=\sqrt{\frac{\rho}{2\pi}}\exp\left\{-\frac{1}{2}\rho\left[\mu-\frac{1}{\rho}\left(\frac{\theta_0}{\tau_0^2}+\frac{x_1}{\sigma_{me}^2}\right)\right]^2\right\},$$

and is distributed as Normal with mean

$$\theta_1=\frac{1}{\rho}\left(\frac{\theta_0}{\tau_0^2}+\frac{x_1}{\sigma_{me}^2}\right)=\frac{\sigma_{me}^2}{\sigma_{me}^2+\tau_0^2}\theta_0+\frac{\tau_0^2}{\sigma_{me}^2+\tau_0^2}x_1$$

and variance

$$\tau_1=\frac{1}{\rho}=\frac{\tau_0^2\sigma_{me}^2}{\sigma_{me}^2+\tau_0^2}.$$

Note that all distributions involved in the Bayesian equation are Normally distributed. This is because the class of normal priors is a conjugate family for the class of normal densities. That is, because the prior distribution and the measurement error density are both normal, the resulting posterior is also normal.

Subsequent observations on μ ($x_2,\ldots, x_{n_{max}}$, if applicable) yield similar results in terms of the properties of the conditional and posterior distributions.

Generally, for any iteration i, the posterior distribution is Normal with mean

$$\theta_i = \frac{1}{\rho}\left(\frac{\theta_{i-1}}{\tau_0^2} + \frac{x_i}{\sigma_{me}^2}\right) = \frac{\sigma_{me}^2}{\sigma_{me}^2 + \tau_{i-1}^2}\theta_{i-1} + \frac{\tau_{i-1}^2}{\sigma_{me}^2 + \tau_{i-1}^2}x_i = \frac{\dfrac{x_i}{\sigma_{me}^2} + \dfrac{\theta_{i-1}}{\tau_{i-1}^2}}{\left(\dfrac{1}{\sigma_{me}^2} + \dfrac{1}{\tau_{i-1}^2}\right)}$$

and variance

$$\tau_i^2 = \frac{1}{\rho} = \frac{\tau_{i-1}^2\,\sigma_{me}^2}{\sigma_{me}^2 + \tau_{i-1}^2} = \frac{1}{\left(\dfrac{1}{\sigma_{me}^2} + \dfrac{1}{\tau_{i-1}^2}\right)}.$$

Specific distributional parameters of subsequent iterations are supplied and discussed in later sections of this paper.

5.4 APPLICATION OF THE MODEL

n_{max}=1 Development

For the case of n_{max}=1, it is desired to make an economic disposition decision based on a single measurement observation. Recall that at n_{max}, $C_{n_{max},L} = C_{n_{max},H} = C_{n_{max}}$ (in this case, = C_1) is the only cutoff value utilized for the decision. This simple case of a single observation may be compared to a fixed sample size of 1. If it were desired to fix the sample size at a single measurement, the same economic method could be used to set a "sampling specification limit" for decision-making. This decision limit would be identical to the maximum economic sequential cutoff value, C_1.

The batch is either accepted or rejected based on the single observation x_1 and its relation to the designated cutoff, C_1. For a single upper specification limit, if

$$x_1 \le C_1$$

the process terminates with batch acceptance; if

$$x_1 > C_1$$

the batch is rejected. The appropriateness of this disposition decision depends on the actual value (unknown) of the batch characteristic being examined. The four possible total cost outcomes associated with disposition decisions for the $n_{max} = 1$ case can be described, as follows:

$$
\begin{aligned}
TC(n_{max}=1,\mu,U,x_1,C_1) \\
&= S && \text{if} \quad x_1 \le C_1; \mu \le U \\
& && \text{(batch appropriately accepted)} \\
&= S && \text{if} \quad x_1 > C_1; \mu > U \\
& && \text{(batch appropriately rejected)} \\
&= S + A && \text{if} \quad x_1 \le C_1; \mu > U \\
& && \text{(batch erroneously accepted)} \\
&= S + R && \text{if} \quad x_1 > C_1; \mu \le U \\
& && \text{(batch erroneously rejected).}
\end{aligned}
$$

All four of these possible outcomes must be considered in the determination of the optimal cutoff, C_1, which minimizes the expected total cost of the procedure. Because μ is not a decision variable and is out of the user's control, it may be expected out of the cost equation. Additionally, U is assumed fixed and is dropped from the variable list. This will reduce the four cost outcomes shown, above, to two cost equations, as follows:

$$E(TC(n_{max}=1,x_1,C_1))$$

$$= S + AP(\mu > U) \qquad \text{if} \qquad x_1 \le C_1$$

$$= S + RP(\mu \le U) \qquad \text{if} \qquad x_1 > C_1$$

It is also desirable to remove the observation variable, x_1, from the expected total cost equation. The two possible disposition outcomes can now be combined into an expected total cost equation which considers the probable locations of μ and x_1:

$$E(TC(n_{max} = 1, C_1) = S + AP(\mu > U, x_1 \le C_1) + RP(\mu \le U, x_1 > C_1)$$

The two probability terms in the expected total cost equation involve the joint probability distribution function of μ and x_1. The joint distribution is bivariate normal and was previously shown in the Bayesian development of the posterior distribution as $J(x_1,\mu)$. This joint probability term leads to two possible Bayesian approaches for the determination of the decision limit, C_1. As previously shown in the discussion of Bayesian decision theory:

$$J(x_1,\mu)=l(x_1|\mu)f(\mu)=g(x_1)h(\mu|x_1)$$

or

Joint Dist.	= Sampling Dist.	x	Prior Dist.	= Marginal Dist.	x Posterior Dist.

The middle product term deals with the prior distribution and the rightmost product term involves the posterior distribution. The use of either of the product terms in place of the joint probability will lead to the exact same expected total cost. In the first approach, the expected cost of inspection is derived from the product expression which involves the prior distribution and may be considered to be a prior cost; the second approach utilizes the posterior distribution and determines a posterior cost based on the observation of x_1.

n_{max}=1: Approach 1 (Prior Costing)

This approach assesses the expected total cost of the sampling sequence based on the expected cost prior to observing x_1. The expected total cost for the procedure of $n_{max}=1$ is given as

$$E(TC(n_{max}=1, C_1) = S + AP(\mu > U, x_1 \le C_1) + RP(\mu \le U, x_1 > C_1)$$
$$= S + A\int_{-\infty}^{C_1}\int_{U}^{\infty} J(\mu, x_1)d\mu dx_1 + R\int_{C_1}^{\infty}\int_{-\infty}^{U} J(\mu, x_1)d\mu dx_1 .$$

In this solution, the product of the prior and sampling distributions is substituted for the joint distribution in the above equation.

$$E(TC(n_{max}=1, C_1))$$
$$= S + A\int_{-\infty}^{C_1}\int_{U}^{\infty} l(x_1|\mu)f(\mu)d\mu dx_1 + R\int_{C_1}^{\infty}\int_{-\infty}^{U} l(x_1|\mu)f(\mu)d\mu dx_1$$
$$= S + A\int_{-\infty}^{C_1}\int_{U}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{me}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\mu}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi}\tau_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_0}{\tau_0}\right)^2\right]d\mu dx_1$$
$$+ R\int_{C_1}^{\infty}\int_{-\infty}^{U} \frac{1}{\sqrt{2\pi}\sigma_{me}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\mu}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi}\tau_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_0}{\tau_0}\right)^2\right]d\mu dx_1$$

This is the expected total cost equation to be minimized through optimization of the single unknown C_1. The optimal decision cutoff (appearing only in the integral limits) may be found by using a unidimensional search procedure. The FORTRAN optimization program which is a product of the research utilizes a step search (on x_1) in locating the C_1 value which results in minimum expected total cost. Note that the cost equation is expressed as the sum of (n_{max}=1)*3 = 3 terms.

n_{max}=1: Approach 2 (Posterior Costing)

This solution method examines the expected total posterior cost based on an observed value of x_1. Recall that this approach makes a substitution for the joint distribution function which involves the posterior (updated) distribution of batch values. Again, the expected total cost equation may be shown in terms of the joint function:

$$E(TC(n_{max}=1,C_1))$$

$$= S + AP(\mu > U, x_1 \le C_1) + RP(\mu \le U, x_1 > C_1)$$

$$= S + A \int_{-\infty}^{C_1} \int_{U}^{\infty} J(\mu, x_1) d\mu dx_1 + R \int_{C_1}^{\infty} \int_{-\infty}^{U} J(\mu, x_1) d\mu dx_1 .$$

Making the substitution of the product of posterior and marginal distributions gives

$$E(TC(n_{max}=1,C_1))$$

$$= S + A \int_{-\infty}^{C_1} \int_{U}^{\infty} h(\mu|x_1) g(x_1) d\mu dx_1 + R \int_{C_1}^{\infty} \int_{-\infty}^{U} h(\mu|x_1) g(x_1) d\mu dx_1$$

$$=S$$

$$+A \int_{-\infty}^{C_1} \int_{U}^{\infty} \frac{1}{\sqrt{2\pi}\tau_0} \exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\sigma_{me}}\right)^2\right] \frac{1}{\sqrt{2\pi\left(\tau_0{}^2+\sigma_{me}{}^2\right)}} \exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0{}^2+\sigma_{me}{}^2\right)}}\right)^2\right] d\mu dx_1$$

$$+R \int_{C_1}^{\infty} \int_{-\infty}^{U} \frac{1}{\sqrt{2\pi}\tau_0} \exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\sigma_{me}}\right)^2\right] \frac{1}{\sqrt{2\pi\left(\tau_0{}^2+\sigma_{me}{}^2\right)}} \exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0{}^2+\sigma_{me}{}^2\right)}}\right)^2\right] d\mu dx_1$$

The posterior parameters θ_1 and τ_1 are as given in the development of the posterior distribution of batch values. The expected posterior cost equation shown, above, is similar in form to the previous equation for expected prior cost. However, the fact that only one of the exponential terms shown in the posterior cost involves the unknown actual value, μ, provides the opportunity for a more straightforward solution to cost minimization. The alternative and more practicable approach is described in the following section.

In assessing the location of x_1 relative to the $n_{max}=1$ cutoff value, two possible expected total costs are considered:

$$
\begin{aligned}
&E(TC(n_{max}=1, x_1, C_1)) \\
&\qquad = S + AP(\mu > U) &&\text{if} \quad x_1 \leq C_1 \\
&\qquad = S + RP(\mu \leq U) &&\text{if} \quad x_1 > C_1
\end{aligned}
$$

Note that the first expected cost is incurred when x_1 dictates batch acceptance and the second expected cost is a consequence of batch rejection. When dealing with the posterior cost of sampling, these two cost consequences may be rewritten as

$$
\begin{aligned}
&\text{Acceptance Cost } = S + AP(\mu > U|x_1) &&\text{if} \quad x_1 \leq C_1\text{, and} \\
&\text{Rejection Cost } = S + RP(\mu \leq U|x_1) &&\text{if} \quad x_1 > C_1.
\end{aligned}
$$

It is logical to assume that, when $x_1=C_1$, there is a disposition indifference as to acceptance and rejection. When considering the equality condition as a point of indifference, it is appropriate to set the acceptance and rejection costs equal and solve for the conditional probability, $P(\mu > U|x_1)$.

Acceptance Cost = Rejection Cost

$$AP(\mu > U|x_1) = RP(\mu \leq U|x_1)$$

$$AP(\mu > U|x_1) = R(1\text{-}P(\mu > U|x_1))$$

$$AP(\mu > U|x_1) = R\text{-}RP(\mu > U|x_1)$$

$$P(\mu > U|x_1) = \frac{R}{R+A}$$

This cost ratio represents the point of indifference for the accept/reject decision. The sampling procedure dictates batch acceptance when

$$P(\mu > U|x_1) \leq \frac{R}{R+A}$$

and rejection if this condition is not met. The x_1 value at which the inequality just holds represents the optimal value of C_1 which minimizes the expected total cost of the single measurement iteration procedure. This value of C_1 may be found through a unidimensional search procedure.

The probability $P(\mu > U|x_1)$ is found by integration of the posterior distribution, $h(\mu|x_1)$. As previously shown, the posterior distribution is distributed as Normal with mean θ_1 and variance τ_1^2. Explicitly,

$$P(\mu > U|x_1) = \int_U^\infty h(\mu|x_1)d\mu .$$

The optimal value of C_1 is found through solution of this integral for values of C_1=x_1, searching for the cutoff value which yields a probability of meeting the cost ratio condition as described. Specifically,

$$P(\mu > U|x_1) = \int_U^\infty \frac{1}{\sqrt{2\pi}\tau_1} \exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\tau_1}\right)^2\right] d\mu ,$$

and with the substitution of C_1 for the first observation, the equation may be written

$$P(\mu > U|x_1{=}C_1) = \int_U^{\infty} \frac{1}{\sqrt{2\pi\tau_1}} \exp\left[-\frac{1}{2}\left(\frac{\mu - \dfrac{\dfrac{C_1}{\sigma_{me}^2}+\dfrac{\theta_0}{\tau_0^2}}{\left(\dfrac{1}{\sigma_{me}^2}+\dfrac{1}{\tau_0^2}\right)}}{\tau_1}\right)^2\right] d\mu .$$

The search for optimal C_1 continues until this probability falls just short of the cost ratio, R/(A+R).

Modification of the Model for Lower Specification

As noted, the only inspection procedural difference when dealing with a lower limit involves a change in inequality direction when assessing the observation mean relative to the appropriate cutoff value. Modifying the expected total cost equation to reflect a lower limit involves only the limits of the integrals of μ and x_i, indicating a change in criteria in assessing the conformity of the batch and the proximity of the observation mean relative to the cutoff, $C_{i,L}$ or $C_{i,H}$. In explanation, the model for $n_{max}=1$ in the case of a lower specification limit is given as

$$E(TC(n_{max} = 1, C_1) = S + AP(\mu < L, x_1 \geq C_1) + RP(\mu \geq L, x_1 < C_1).$$

Assessing this expected total cost by both prior and posterior costing approaches requires only a change in integral limits for the upper specification equations presented in the previous section. Specifically, the prior costing equation becomes

$$E(TC(n_{max}{=}1, C_1))$$
$$= S + A\int_{C_1}^{\infty}\int_{-\infty}^{L} l(x_1|\mu)f(\mu)d\mu dx_1 + R\int_{-\infty}^{C_1}\int_{L}^{\infty} l(x_1|\mu)f(\mu)d\mu dx_1$$

$$= S + A\int_{C_1}^{\infty}\int_{-\infty}^{L}\frac{1}{\sqrt{2\pi}\sigma_{me}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\mu}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi}\tau_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_0}{\tau_0}\right)^2\right]d\mu dx_1$$

$$+R\int_{-\infty}^{C_1}\int_{L}^{\infty}\frac{1}{\sqrt{2\pi}\sigma_{me}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\mu}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi}\tau_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_0}{\tau_0}\right)^2\right]d\mu dx_1.$$

Note that the distributions involved are exactly the same as those presented in the upper limit case. Similarly, the posterior costing approach is given as

$$E(TC(n_{max}=1,C_1))$$

$$=S + A\int_{C_1}^{\infty}\int_{-\infty}^{L}h(\mu|x_1)g(x_1)d\mu dx_1 + R\int_{-\infty}^{C_1}\int_{L}^{\infty}h(\mu|x_1)g(x_1)d\mu dx_1$$

$$=S$$

$$+A\int_{C_1}^{\infty}\int_{-\infty}^{L}\frac{1}{\sqrt{2\pi}\tau_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi\left(\tau_0^2+\sigma_{me}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0^2+\sigma_{me}^2\right)}}\right)^2\right]d\mu dx_1$$

$$+R\int_{-\infty}^{C_1}\int_{L}^{\infty}\frac{1}{\sqrt{2\pi}\tau_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi\left(\tau_0^2+\sigma_{me}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0^2+\sigma_{me}^2\right)}}\right)^2\right]d\mu dx_1.$$

For $n_{max}>1$, modifications for the lower specification case follow logically from the $n_{max}=1$ situation.

$n_{max}=2$ Development

The case for $n_{max}=2$ (and, indeed, for any $n_{max}>1$) builds upon the development presented for $n_{max}=1$. As was indicated, the expected total cost term is logically expressed as the sum of $(n_{max}=2)*3 = 6$ terms. The cost components for the $n_{max}=1$ procedure which involve the single C_1 value now require either $C_{1,L}$ or $C_{1,H}$ as a decision limit at iteration

$n=1$. The region between $C_{1,L}$ and $C_{1,H}$ defines the continuation interval which dictates observation of the second measurement. If the second observation is made, the single limit $C_{n_{max}} = C_2$ provides the disposition decision limit for comparison with the average of x_1 and x_2, designated $\overline{x}_2$. If $\overline{x}_2 > C_2$, the batch is rejected; otherwise, the batch is accepted.

As in the development of $n_{max}=1$, it is informative to show the possible cost consequences of the sequential measurement process for $n_{max}=2$:

$TC(n_{max}=2,\mu,U,x_1,x_2,C_{1,L},C_{1,H},C_2)$

$= S$	if	$x_1 \le C_{1,L}; \mu \le U$
		(batch appropriately accepted on n=1)
$= S$	if	$x_1 > C_{1,H}; \mu > U$
		(batch appropriately rejected on n=1)
$= S+S$	if	$C_{1,L} < x_1 \le C_{1,H}; \overline{x}_2 \le C_2; \mu \le U$
		(batch appropriately accepted on n=2)
$= S+S$	if	$C_{1,L} < x_1 \le C_{1,H}; \overline{x}_2 > C_2; \mu > U$
		(batch appropriately rejected on n=2)
$= S+A$	if	$x_1 \le C_{1,L}; \mu > U$
		(batch erroneously accepted on n=1)
$= S+R$	if	$x_1 > C_{1,H}; \mu \le U$
		(batch erroneously rejected on n=1)
$= S+S+A$	if	$C_{1,L} < x_1 \le C_{1,H}; \overline{x}_2 \le C_2; \mu > U$
		(batch erroneously accepted on n=2)
$= S+S+R$	if	$C_{1,L} < x_1 \le C_{1,H}; \overline{x}_2 > C_2; \mu \le U$
		(batch erroneously rejected on n=2)

By combining terms, as before, the costs may be consolidated into a comprehensive equation for the expected total cost of the $n_{max}=2$ procedure:

$$
\begin{aligned}
&E(TC(n_{max}=2, C_{1,L}, C_{1,H}, C_2)) \\
&\qquad = S + AP(\mu > U; x_1 \le C_{1,L}) + RP(\mu \le U; x_1 > C_{1,H}) \\
&\qquad\qquad + SP(C_{1,L} < x_1 \le C_{1,H}) \\
&\qquad\qquad + AP(\mu > U; C_{1,L} < x_1 \le C_{1,H}; \bar{x}_2 \le C_2) \\
&\qquad\qquad + RP(\mu \le U; C_{1,L} < x_1 \le C_{1,H}; \bar{x}_2 > C_2)
\end{aligned}
$$

The expected total cost equation is logically given in terms of six cost components (two each of iteration, false acceptance and false rejection).

If the outcome of observation x_1 dictates a second iteration on μ, the posterior distribution of μ given x_1 becomes the prior distribution for x_2. The Bayesian decision theory relationship which was previously specified for the case of $n_{max}=1$ may be modified to reflect this situation, as shown below. Note that both sides of the Bayesian equation must be multiplied by the marginal distribution of x_1 to maintain the integrity of the relationships.

$$J(x_1, x_2, \mu) = l(x_2|\mu)h(\mu|x_1)g(x_1) = g(x_2)g(x_1)h(\mu|x_1, x_2)$$

As before, the $n_{max}=2$ situation may be solved through Approach 1 (prior costs) or Approach 2 (posterior costs). Examination of the prior and posterior equivalents of the joint distribution of x_1, x_2 and μ indicates that the solution will involve triple integrals. The posterior approach (2) provides the more straightforward solution, as explained in a subsequent section of this paper.

n_{max}=2: Approach 1 (Prior Costing)

Solution of the n_{max}=2 problem using the prior costing approach requires a multivariate search for the three required disposition cutoff values ($C_{1,L}$, $C_{1,H}$ and C_2).

Substitutions for the joint distributions in the expected total cost equation are made from the left side of the Bayesian decision theory equation, as presented. Note that the second and third terms in the total cost equation involve only the joint distribution between μ and x_1. These terms are very similar to the terms involving the single cutoff, C_1, found in the expected cost for the case of n_{max}=1. The only difference in these costs at n=2 associated with incorrect disposition on the first iteration is in the existence of the interval between $C_{1,L}$ and $C_{1,H}$ which allows for the possibility of continuation of the measurement procedure. Restating the cost equation using the prior equations yields

$$
\begin{aligned}
&E(TC(n_{max}=2, C_{1,L}, C_{1,H}, C_2)) \\
&= S + AP(\mu > U; x_1 \le C_{1,L}) + RP(\mu \le U; x_1 > C_{1,H}) \\
&\quad + SP(C_{1,L} < x_1 \le C_{1,H}) \\
&\quad + AP(\mu > U; C_{1,L} < x_1 \le C_{1,H}; \bar{x}_2 \le C_2) \\
&\quad + RP(\mu \le U; C_{1,L} < x_1 \le C_{1,H}; \bar{x}_2 > C_2) \\
&= S + A \int_{-\infty}^{C_{1,L}} \int_{U}^{\infty} J(\mu, x_1) d\mu dx_1 + R \int_{C_{1,H}}^{\infty} \int_{-\infty}^{U} J(\mu, x_1) d\mu dx_1 \\
&\quad + S \int_{C_{1,L}}^{C_{1,H}} g(x_1) dx_1 \\
&\quad + A \int_{C_{1,L}}^{C_{1,H}} \int_{-\infty}^{2C_2 - x_1} \int_{U}^{\infty} J(\mu, x_1, x_2) d\mu dx_2 dx_1 \\
&\quad + R \int_{C_{1,L}}^{C_{1,H}} \int_{2C_2 - x_1}^{\infty} \int_{-\infty}^{U} J(\mu, x_1, x_2) d\mu dx_2 dx_1
\end{aligned}
$$

In the prior costing approach, substitution for the joint distribution is made from the left side of the Bayesian decision relationships which were previously presented.

$$
\begin{aligned}
&E(TC(n_{max}=2, C_{1,L}, C_{1,H}, C_2)) \\
&\quad = S + A \int_{-\infty}^{C_{1,L}} \int_{U}^{\infty} l(x_1|\mu) f(\mu)\, d\mu\, dx_1 + R \int_{C_{1,H}}^{\infty} \int_{-\infty}^{U} l(x_1|\mu) f(\mu)\, d\mu\, dx_1 \\
&\qquad + S \int_{C_{1,L}}^{C_{1,H}} g(x_1)\, dx_1 \\
&\qquad + A \int_{C_{1,L}}^{C_{1,H}} \int_{-\infty}^{2C_2 - x_1} \int_{U}^{\infty} l(x_2|\mu, x_1) h(\mu|x_1) g(x_1)\, d\mu\, dx_2\, dx_1 \\
&\qquad + R \int_{C_{1,L}}^{C_{1,H}} \int_{2C_2 - x_1}^{\infty} \int_{-\infty}^{U} l(x_2|\mu, x_1) h(\mu|x_1) g(x_1)\, d\mu\, dx_2\, dx_1
\end{aligned}
$$

Substituting the normal distributional parameters puts the expected total cost equation in a form suitable for minimization.

$$
\begin{aligned}
&E(TC(n_{max}=2, C_{1,L}, C_{1,H}, C_2)) \\
&\quad = S + A \int_{-\infty}^{C_{1,L}} \int_{U}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{me}} \exp\left[-\frac{1}{2}\left(\frac{x_1 - \mu}{\sigma_{me}}\right)^2\right] \frac{1}{\sqrt{2\pi}\tau_0} \exp\left[-\frac{1}{2}\left(\frac{\mu - \theta_0}{\tau_0}\right)^2\right] d\mu\, dx_1 \\
&\quad + R \int_{C_{1,H}}^{\infty} \int_{-\infty}^{U} \frac{1}{\sqrt{2\pi}\sigma_{me}} \exp\left[-\frac{1}{2}\left(\frac{x_1 - \mu}{\sigma_{me}}\right)^2\right] \frac{1}{\sqrt{2\pi}\tau_0} \exp\left[-\frac{1}{2}\left(\frac{\mu - \theta_0}{\tau_0}\right)^2\right] d\mu\, dx_1 \\
&\qquad + S \int_{C_{1,L}}^{C_{1,H}} \frac{1}{\sqrt{2\pi\left(\tau_0^2 + \sigma_{me}^2\right)}} \exp\left[-\frac{1}{2}\left(\frac{x_1 - \theta_0}{\sqrt{\left(\tau_0^2 + \sigma_{me}^2\right)}}\right)^2\right] dx_1
\end{aligned}
$$

$$+A\int_{C_{1,L}}^{C_{1,H}}\int_{-\infty}^{2C_2-x_1}\int_{U}^{\infty}\frac{1}{\sqrt{2\pi}\sigma_{me}}\exp\left[-\frac{1}{2}\left(\frac{x_2-\mu}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi}\tau_1}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\tau_1}\right)^2\right]\times$$

$$\frac{1}{\sqrt{2\pi\left(\tau_0^2+\sigma_{me}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0^2+\sigma_{me}^2\right)}}\right)^2\right]d\mu dx_2 dx_1$$

$$+R\int_{C_{1,L}}^{C_{1,H}}\int_{2C_2-x_1}^{\infty}\int_{-\infty}^{U}\frac{1}{\sqrt{2\pi}\sigma_{me}}\exp\left[-\frac{1}{2}\left(\frac{x_2-\mu}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi}\tau_1}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\tau_1}\right)^2\right]\times$$

$$\frac{1}{\sqrt{2\pi\left(\tau_0^2+\sigma_{me}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0^2+\sigma_{me}^2\right)}}\right)^2\right]d\mu dx_2 dx_1$$

An apparent problem is presented in that the value of the first observation, x_1, is imbedded within the prior distributional mean for μ, specified as θ_1. This implies that an optimal C_2 is dependent on the observed x_1 and must be found for every possible x_1 in the interval $C_{1,L}$ to $C_{1,H}$. However, search for the C_2 value which minimizes the equation for any observation, x_1, yields a single value for the C_2 cutoff. Development of a method to locate the single optimal C_2 is shown in the next section for posterior costing (Approach 2). Existence of the single optimal C_2 cutoff implies that the total cost equation is reducible to a form which does not indicate an x_1 dependence. However, this simplification is not provided in this paper and is unavailable in the literature.

Utilizing the knowledge that there exists a single optimum for C_2, it is possible to execute a multivariate search to minimize the expected total cost equation for all three required cutoff values ($C_{1,L}$, $C_{1,H}$ and C_2).

n_{max}=2: Approach 2 (Posterior Costing)

In the posterior costing approach, the components of the total cost equation present, as in approach 1, triple integrals which are required for cost minimization and parameter optimization. However, the use of the cost ratio R/(R+A) allows a shortcut for searching for the cutoff values involved in the n_{max}=2 sequential procedure. This method (which may be used for the location of any $C_{n_{max}}$) is presented in this section in lieu of a more complicated method requiring simultaneous search for the three cutoff values $C_{1,L}$, $C_{1,H}$ and C_2.

The expected total cost of the procedure after reaching the second iteration can be expressed in terms of the location of $\overline{x}_2$ relative to the C_2 cutoff. Associated costs (conditional on reaching n=2) are identified as either acceptance or rejection costs:

$$\text{Acceptance Cost} = S + AP(\mu > U|x_1,x_2) \qquad \text{if} \qquad \overline{x}_2 \le C_2, \text{ and}$$

$$\text{Rejection Cost} = S + RP(\mu \le U|x_1,x_2) \qquad \text{if} \qquad \overline{x}_2 > C_2.$$

As in the case of n_{max}=1, when $\overline{x}_2 = C_2$, there is an assumed indifference as to the disposition of the batch. This economic indifference point is located by equating the acceptance and rejection costs as follows:

$$\begin{aligned}
\text{Acceptance Cost} &= \text{Rejection Cost} \\
AP(\mu > U|x_1,x_2) &= RP(\mu \le U|x_1,x_2) \\
AP(\mu > U|x_1,x_2) &= R(1-P(\mu > U|x_1,x_2)) \\
AP(\mu > U|x_1,x_2) &= R-RP(\mu > U|x_1,x_2) \\
P(\mu > U|x_1,x_2) &= \frac{R}{A + R}
\end{aligned}$$

This economic indifference point is identical to that found for the case of n_{max}=1; indeed, the cost ratio R/(R+A) holds for any n_{max}.

This cost ratio represents the point of indifference for the accept/reject decision. The sampling procedure dictates batch acceptance when

$$P(\mu > U|x_1,x_2) \le \frac{R}{A+R}$$

and rejection if this condition is not met. The $\bar{x}_2$ (average of x_1 and x_2) value at which the inequality just holds represents the optimal value of C_2 which minimizes the expected total cost of the measurement iteration procedure given that it has reached the second observation. This value of C_2 may be found through a unidimensional search procedure.

The probability $P(\mu > U|x_1,x_2)$ is found by integration of the posterior distribution, $h(\mu|x_1,x_2)$. This posterior distribution of μ is distributed as Normal with mean θ_2 and variance τ_2^2. Explicitly,

$$P(\mu > U|x_1,x_2) = \int_U^{\infty} h(\mu|x_1,x_2)d\mu\,.$$

The optimal value of C_2 is found through solution of this integral for values of $\bar{x}_2$=C_2, searching for the cutoff value which yields a probability meeting the cost ratio condition as described. Because the posterior distribution parameter, θ_2, is given in terms of x_1 and x_2, the indifference point is actually expressed as the point at which x_2=$2C_2$-x_1. Specifically,

$$P(\mu > U|x_1,x_2) = \int_U^{\infty} \frac{1}{\sqrt{2\pi}\tau_2} \exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_2}{\tau_2}\right)^2\right] d\mu\,,$$

where

$$\theta_2 = \frac{\dfrac{x_2}{\sigma_{me}^2} + \dfrac{\theta_1}{\tau_1^2}}{\left(\dfrac{1}{\sigma_{me}^2} + \dfrac{1}{\tau_1^2}\right)}, \qquad \theta_1 = \frac{\dfrac{x_1}{\sigma_{me}^2} + \dfrac{\theta_0}{\tau_0^2}}{\left(\dfrac{1}{\sigma_{me}^2} + \dfrac{1}{\tau_0^2}\right)},$$

$$\tau_2^2 = \frac{1}{\left(\frac{1}{\sigma_{me}^2} + \frac{1}{\tau_1^2}\right)} \qquad \text{and} \qquad \tau_1^2 = \frac{1}{\left(\frac{1}{\sigma_{me}^2} + \frac{1}{\tau_0^2}\right)}.$$

At the point of interest, $x_2=2C_2-x_1$, and the posterior mean, θ_2 is given as

$$\theta_2 = \frac{\frac{2C_2 - x_1}{\sigma_{me}^2} + \frac{\theta_1}{\tau_1^2}}{\left(\frac{1}{\sigma_{me}^2} + \frac{1}{\tau_1^2}\right)}.$$

Making this substitution eliminates x_2 from the probability and leaves the integral in terms of the single observation, x_1. With the probability integral in this form, the entire expression is simplified and x_1 drops out of the expression. The reduced form is given as:

$$P(\mu > U|\bar{x}_2=C_2) =$$

$$\int_U^{\infty} \frac{1}{\sqrt{2\pi}} \frac{\sqrt{2\tau_0^2 + \sigma_{me}^2}}{\sigma_{me}\tau_0} \exp\left[\frac{\left(2\mu\tau_0^2 + \mu\sigma_{me}^2 - 2C_2\tau_0^2 - \theta_0\sigma_{me}^2\right)^2}{\tau_0^2\sigma_{me}^2} \frac{-0.5}{2\tau_0^2 + \sigma_{me}^2}\right] d\mu$$

The optimum value of C_2 is found through unidimensional search of this integral expression until the probability value falls just short of the economic indifference ratio, R/(R+A).

With this optimal C_2 value in-hand, the cutoff decision values, $C_{1,L}$ and $C_{1,H}$ are found through an appropriate multivariate search procedure utilizing the full expected total cost equation. This equation is obtained by substituting the posterior product term presented in the Bayesian relationship for the joint distribution of μ, x_1 and x_2. The required terms are found on the right side of the Bayesian decision theory relationship.

$$E(TC(n_{max}=2,C_{1,L},C_{1,H},C_2))$$

$$\begin{aligned}
&= S + A\int_{-\infty}^{C_{1,L}}\int_{U}^{\infty} h(\mu|x_1)g(x_1)d\mu dx_1 + R\int_{C_{1,H}}^{\infty}\int_{-\infty}^{U} h(\mu|x_1)g(x_1)d\mu dx_1 \\
&\quad + S\int_{C_{1,L}}^{C_{1,H}} g(x_1)dx_1 \\
&\quad + A\int_{C_{1,L}}^{C_{1,H}}\int_{-\infty}^{2C_2-x_1}\int_{U}^{\infty} h(\mu|x_1,x_2)g(x_2)g(x_1)d\mu dx_2 dx_1 \\
&\quad + R\int_{C_{1,L}}^{C_{1,H}}\int_{2C_2-x_1}^{\infty}\int_{-\infty}^{U} h(\mu|x_1,x_2)g(x_2)g(x_1)d\mu dx_2 dx_1
\end{aligned}$$

Substitution of the appropriate Normal distribution parameters leaves this cost equation in the form required for minimization and optimal parameter search,

$$E(TC(n_{max}=2,C_{1,L},C_{1,H},C_2))$$

$$\begin{aligned}
&=S \\
&+A\int_{-\infty}^{C_{1,L}}\int_{U}^{\infty}\frac{1}{\sqrt{2\pi}\tau_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi\left(\tau_0{}^2+\sigma_{me}{}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0{}^2+\sigma_{me}{}^2\right)}}\right)^2\right]d\mu dx_1 \\
&+R\int_{C_{1,H}}^{\infty}\int_{-\infty}^{U}\frac{1}{\sqrt{2\pi}\tau_0}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\sigma_{me}}\right)^2\right]\frac{1}{\sqrt{2\pi\left(\tau_0{}^2+\sigma_{me}{}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0{}^2+\sigma_{me}{}^2\right)}}\right)^2\right]d\mu dx_1 \\
&\quad + S\int_{C_{1,L}}^{C_{1,H}}\frac{1}{\sqrt{2\pi\left(\tau_0{}^2+\sigma_{me}{}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0{}^2+\sigma_{me}{}^2\right)}}\right)^2\right]dx_1
\end{aligned}$$

$$+A\int_{C_{1,L}}^{C_{1,H}}\int_{-\infty}^{2C_2-x_1}\int_{U}^{\infty}\frac{1}{\sqrt{2\pi}\tau_2}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_2}{\sigma_{me}}\right)^2\right]d\mu\frac{1}{\sqrt{2\pi\left(\tau_1^2+\sigma_{me}^2\right)}}\times$$

$$\exp\left[-\frac{1}{2}\left(\frac{x_2-\theta_1}{\sqrt{\left(\tau_1^2+\sigma_{me}^2\right)}}\right)^2\right]dx_2\frac{1}{\sqrt{2\pi\left(\tau_0^2+\sigma_{me}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0^2+\sigma_{me}^2\right)}}\right)^2\right]dx_1$$

$$+R\int_{C_{1,L}}^{C_{1,H}}\int_{2C_2-x_1}^{\infty}\int_{-\infty}^{U}\frac{1}{\sqrt{2\pi}\tau_2}\exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_2}{\sigma_{me}}\right)^2\right]d\mu\frac{1}{\sqrt{2\pi\left(\tau_1^2+\sigma_{me}^2\right)}}\times$$

$$\exp\left[-\frac{1}{2}\left(\frac{x_2-\theta_1}{\sqrt{\left(\tau_1^2+\sigma_{me}^2\right)}}\right)^2\right]dx_2\frac{1}{\sqrt{2\pi\left(\tau_0^2+\sigma_{me}^2\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\left(\tau_0^2+\sigma_{me}^2\right)}}\right)^2\right]dx_1$$

The optimal value of C_2 (known from the economic indifference point) is substituted in this equation and a multivariate search procedure is used to locate $C_{1,L}$ and $C_{1,H}$. The search procedure used in the research optimization program is that proposed by Nelder and Mead [40].

General $n_{max}>2$ Development

For any incremental increase in n_{max}, three cost components are added to the expected total cost model of n_{max}-1: one each for sampling, false acceptance and false rejection. All cost components which are found in the model of n_{max}-1 are retained, with the substitution of $C_{n_{max},L}$ for $C_{n_{max}-1}$ in the false acceptance component involving the single cutoff value and $C_{n_{max},H}$ for $C_{n_{max}-1}$ in the corresponding false rejection component.

n_{max}>2: Approach 1 (Prior Costing)

The additional sampling component gives the probability that the observation at n_{max}-1 will fall between $C_{n_{max},L}$ and $C_{n_{max},H}$, requiring an observation at n_{max}. It may be expressed as

$$S \int_{C_{1,L}}^{C_{1,H}} \int_{2C_{2,L}-x_1}^{2C_{2,H}-x_1} \cdots \cdots \int_{(n_{max}-1)C_{n_{max}-1,L}-\sum_{i=1}^{n_{max}-2} x_i}^{(n_{max}-1)C_{n_{max}-1,H}-\sum_{i=1}^{n_{max}-2} x_i} g(x_{n_{max}-1}) \ldots g(x_2)g(x_1)dx_{n_{max}-1} \ldots dx_2 dx_1$$

Recall that all relevant distributions are distributed as normal. In this component, any marginal, $g(x_n)$, is normal with mean θ_{n-1} and variance $(\tau_{n-1}^2+\sigma_{me}^2)$.

The other two terms express the probable cost of making a wrong disposition decision based upon the culmination of all observed values through n_{max}-1. The expected cost of false acceptance is given as

$$A \int_{C_{1,L}}^{C_{1,H}} \int_{2C_{2},L-x_1}^{2C_{2,H}-x_1} \cdots\cdots \int_{-\infty}^{(n_{max})C_{n_{max}}-\sum_{i=1}^{n_{max}-1} x_i} \int_{U}^{\infty} l(x_{n_{max}} | \mu, x_1, \ldots, x_{n_{max}-1}) \times$$
$$f(\mu | x_1, \ldots, x_{n_{max}-1}) g(x_{n_{max}-1}) \ldots g(x_2) g(x_1) d\mu dx_{n_{max}-1} \ldots dx_2 dx_1$$

and the expected cost of false rejection is written

$$R \int_{C_{1,L}}^{C_{1,H}} \int_{2C_{2},L-x_1}^{2C_{2,H}-x_1} \cdots\cdots \int_{(n_{max})C_{n_{max}}-\sum_{i=1}^{n_{max}-1} x_i}^{\infty} \int_{-\infty}^{U} l(x_{n_{max}} | \mu, x_1, \ldots, x_{n_{max}-1}) \times$$
$$f(\mu | x_1, \ldots, x_{n_{max}-1}) g(x_{n_{max}-1}) \ldots g(x_2) g(x_1) d\mu dx_{n_{max}-1} \ldots dx_2 dx_1$$

The sampling distribution $l(x_{n_{max}} | \mu, x_1, \ldots, x_{n_{max}-1})$ is distributed normal(μ,σ_{me}^2). Note that previous observations on μ do not affect the fact that the actual batch value is μ. The function shown as $f(\mu | x_1, \ldots, x_{n_{max}-1})$, is in the notation of a prior distribution, but is also the posterior $h(\mu | x_1, \ldots, x_{n_{max}-2})$. It is distributed as normal with mean $\theta_{n_{max}-1}$ and variance $\tau_{n_{max}-1}^2$.

Optimization of all cutoff values (($C_{1,L}$, $C_{1,H}$),..., $C_{n_{max}}$) is achieved through a multivariate search [40] for the minimum expected total cost.

n_{max}>2: Approach 2 (Posterior Costing)

As in the specific application examples shown for n_{max}=1 and 2, the posterior costing approach is easier to utilize than the prior approach due to the ability to locate $C_{n_{max}}$ (independent of other variables) through a unidimensional search. The economic indifference ratio, R/(R+A) is calculated and compared with the posterior probability $P(\mu > U | x_1, \ldots, x_{n_{max}})$ for various values of $\bar{x}_{n_{max}} = C_{n_{max}}$. This probability is given as

$$P(\mu > U | x_1, \ldots, x_{n_{max}}) = \int_U^\infty h(\mu | x_1, \ldots, x_{n\,max}) d\mu .$$

The posterior distribution $h(\mu | x_1, \ldots, x_{n_{max}})$ is distributed as normal with mean $\theta_{n_{max}}$ and variance $\tau_{n_{max}}^2$. Specifically,

$$P(\mu > U | x_1, \ldots, x_{n_{max}}) = \int_U^\infty \frac{1}{\sqrt{2\pi}\tau_{n_{max}}} \exp\left[-\frac{1}{2}\left(\frac{\mu - \theta_{n_{max}}}{\tau_{n_{max}}} \right)^2 \right] d\mu .$$

The search for the point at which this probability falls just short of the cost ratio actually involves substitution of $x_{n_{max}} = (n_{max})C_{n_{max}} - \sum_{i=1}^{n_{max}-1} x_i$ (rather than a direct substitution of $\overline{x}_{n_{max}} = C_{n_{max}}$). After substitution is made for $x_{n_{max}}$, the integrand is reducible to a term which is independent of all observations on μ.

Location of $C_{n_{max}}$ through this unidimensional approach simply reduces the scope of the subsequent multivariate search. All remaining optimal cutoff values must be found through minimization of the expected total cost equation.

As indicated in the discussion of the prior costing approach, the expected total cost model for any n_{max} simply builds on the cost components from the (n_{max}-1 model). The terms which involved the single cutoff value $C_{n_{max}-1}$ are modified through substitution of $C_{n_{max}-1,L}$ (false acceptance term) and $C_{n_{max}-1,H}$ (false rejection term) for the single value. An additional three terms are then appended to the expected total cost equation to reflect iteration, false acceptance and false rejection costs associated with the additional measurement observation.

The additional sampling cost term is identical to the term in the prior costing approach; this iteration cost reflects the probability that an additional measurement observation (at n_{max}) will be required. It is given as

$$S \int_{C_{1,L}}^{C_{1,H}} \int_{2C_{2,L}-x_1}^{2C_{2,H}-x_1} \cdots \int_{(n_{max}-1)C_{n_{max}-1,L}-\sum_{i=1}^{n_{max}-2} x_i}^{(n_{max}-1)C_{n_{max}-1,H}-\sum_{i=1}^{n_{max}-2} x_i} g(x_{n_{max}-1}) \ldots g(x_2) g(x_1)\, dx_{n_{max}-1} \ldots dx_2 dx_1$$

Again, any marginal, $g(x_n)$, is normal with mean θ_{n-1} and variance $(\tau_{n-1}^2+\sigma_{me}^2)$.

The posterior expected cost of false acceptance is given as

$$A \int_{C_{1,L}}^{C_{1,H}} \int_{2C_{2,L}-x_1}^{2C_{2,H}-x_1} \cdots\cdots \int_{-\infty}^{(n_{max})C_{n_{max}} - \sum_{i=1}^{n_{max}-1} x_i} \int_{U}^{\infty} h(\mu|x_1,\ldots,x_{n_{max}}) g(x_{n_{max}}) \ldots$$

$$g(x_2)g(x_1)d\mu dx_{n_{max}} \ldots dx_2 dx_1,$$

and the expected cost of false rejection is written

$$R \int_{C_{1,L}}^{C_{1,H}} \int_{2C_{2,L}-x_1}^{2C_{2,H}-x_1} \cdots\cdots \int_{(n_{max})C_{n_{max}} - \sum_{i=1}^{n_{max}-1} x_i}^{\infty} \int_{-\infty}^{U} h(\mu|x_1,\ldots,x_{n_{max}}) g(x_{n_{max}}) \ldots$$

$$g(x_2)g(x_1)d\mu dx_{n_{max}} \ldots dx_2 dx_1.$$

The posterior distribution $h(\mu|x_1,\ldots,x_{n_{max}})$ is distributed as normal($\theta_{n_{max}}, \tau_{n_{max}}{}^2$), and the marginal, $g(x_n)$, is also normal with mean θ_{n-1} and variance $(\tau_{n-1}{}^2+\sigma_{me}{}^2)$.

5.5 COMPUTER OPTIMIZATION OF THE ECONOMIC MODEL

Computer solution of the economic problem is, of course, possible through a variety of computer languages, logic flows and search procedures. The FORTRAN programs which are presented in the Appendices are but examples of many possible solution approaches.

The computer program from which research results are generated is presented in Appendix A. This comprehensive FORTRAN program uses the posterior costing approach in solution of the economic problem; it also contains logic which allows the user to model the problem statistically, as presented in chapters Three and Four. Appendix B provides the code for the prior costing solutions for n_{max}=1 and 2. The code of Appendix B is provided for the user's information and is not used to generate any of the results presented in subsequent sections of this paper.

The search procedure which is used in all parameter optimizations is that developed by Nelder and Mead [40] and discussed by Olsson [42]. As previously indicated in this chapter, compensation for measurement system bias is made following a zero-bias sampling plan design. Cutoff values for the optimal economic sampling plan with and without bias are presented as output from the programs (see next chapter).

The general logic for the FORTRAN solutions presented as a piece of this research is provided in this section. As indicated in previous discussions, each incremental increase in n_{max} builds on the economic model of n_{max}-1. The programming methods for each of the Bayesian costing approaches (prior and posterior) are discussed in the following sections of this paper.

The programming logic which is presented is general for the case of a single upper specification limit. However, because the lower limit expected total cost equations differ from the upper only in the limits of the integrals, the practical solution of the lower limit case can actually be performed using the upper specification logic. The FORTRAN code which is discussed and is presented in Appendix A (posterior economic costing) utilizes upper specification equations in solving both the upper and lower limit cases. This is accomplished by utilizing the logical symmetry of the solution procedure around the specification limit. That is, the optimal cutoffs for an upper specification limit, U, and a prior distribution mean, θ_0 (a distance, $-\Delta$, from U), are easily translated into the optimal cutoffs for a lower limit, L (=U), and a prior distribution mean $+\Delta$ from L. These cutoffs are simply translated symmetrically around the specification limit, modifying lower cutoff limits into higher cutoffs as logically required.

Practically, an input lower limit to the computer program is treated as an upper limit with a modified, symmetrical θ_0 around the specification, and the resulting optimal cutoffs are translated into lower limit cutoffs through symmetry around the limit prior to presentation to the program user. Given this simple solution to the lower limit problem, the following programming solution addresses only the upper specification case. As indicated in the following chapter, solutions for both upper and lower limits are provided through the FORTRAN program.

Approach 1 (Prior Costing)

The prior costing approach, as presented, requires a multivariate search in all but the basic (n_{max}=1 with a single cutoff value, C_1) case. For each incremental increase in n_{max}, two additional variables are added to the expected total cost equation. In general, the number of search variables for the prior costing methodology is given by [(n_{max}-1)*2 + 1].

At the heart of the prior costing equation to be minimized is the bivariate normal distribution given by the product of the prior and sampling distributions. This function is not cost dependent and is uniquely determined by the prior distribution, the measurement error distribution variance and the upper specification limit. On the first observation (regardless of n_{max}), this function is within a double integral and is used to find the probabilities of false acceptance and rejection on n=1:

$$\int\int \frac{1}{\sqrt{2\pi}\sigma_{me}} \exp\left[-\frac{1}{2}\left(\frac{x_1-\mu}{\sigma_{me}}\right)^2\right] \frac{1}{\sqrt{2\pi}\tau_0} \exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_0}{\tau_0}\right)^2\right] d\mu dx_1 .$$

The limits of the integrals depend on the cost term under consideration (false acceptance or false rejection). Using Mathcad 4.0[1] , μ is integrated out of the function, appropriate limits are substituted, and the expression reduces to a single integral over x_1. The reduced

[1] *Mathcad 4.0 User's Guide Windows Version* (1993), Mathsoft, Inc, Cambridge, MA.

expression is a conditional probability of false disposition given the actual location of μ relative to U. The conditional probability of false rejection is given as

$$\int_{C_{1,H}}^{B} -.19947\left[\left[\operatorname{erf}\left[.70711\frac{-U\tau_0^2 - U\sigma_{me}^2 + \theta_0^2\sigma_{me}^2 + x_1\tau_0^2}{\left[\sigma_{me}\left(\tau_0\sqrt{\tau_0^2+\sigma_{me}^2}\right)\right]}\right]-1\right]\frac{\exp\left[-.5\frac{(x_1-\theta_0)^2}{\tau_0^2+\sigma_{me}^2}\right]}{\sqrt{\tau_0^2+\sigma_{me}^2}}\right]dx_1$$

and the corresponding probability of false acceptance if the batch is non-conforming is

$$\int_{A}^{C_{1,L}} -.19947\left[\left[\operatorname{erf}\left[.70711\frac{-U\tau_0^2 - U\sigma_{me}^2 + \theta_0^2\sigma_{me}^2 + x_1\tau_0^2}{\left[\sigma_{me}\left(\tau_0\sqrt{\tau_0^2+\sigma_{me}^2}\right)\right]}\right]-1\right]\frac{\exp\left[-.5\frac{(x_1-\theta_0)^2}{\tau_0^2+\sigma_{me}^2}\right]}{\sqrt{\tau_0^2+\sigma_{me}^2}}\right]dx_1$$

Note that both of these false disposition expressions are general for n=1, regardless of n_{max} (for n_{max}=1, $C_{1,L} = C_{1,H} = C_1$). The error function erf() in the above equations is defined as:

$$\operatorname{erf}(v) = \frac{2}{\sqrt{\pi}}\int_0^v \exp\left[-t^2\right]dt .$$

The FORTRAN subroutine which solves this expression is taken from Stegun and Zucker [55].

In addition to the two costs of false disposition on the first observation, there is also the sampling component, S. The probability of incurring this iteration cost on the first observation is 1.0.

In order to optimize the n=1 cutoff variables, the conditional probability curves are considered. These are simply the bivariate normal probability function (of x_1 and μ) conditioned over ranges of μ from U to +/- infinity. The probabilities of observing a given x_1 are considered for (μ>U) and ($\mu \leq$U). If these two curves overlap, then a search for $C_{1,L}$ and $C_{1,H}$ (=C_1 for n_{max}=1) is required to find the economic optimal tradeoff point. If the two curves do not intersect, there is an interval wherein both functions are essentially zero, and any cutoff chosen in that region yields a minimum expected cost for n=1 (involving only the sampling cost, S). The points at which the functions approach zero are located through a step search and compared.

The zero points for x_1 located for each conditional probability plot may be considered as the infinity limits (+/-) for integration. In the code provided, these are identified as the A and B limits of integration. "A" is the lower limit for the probability of observing x_1 given μ>U, while "B" is the upper limit for the probability of observing x_1 if $\mu \leq$U. An illustration of the n_{max}=1 situation in which the probability plots overlap is shown in Figure 5.2.

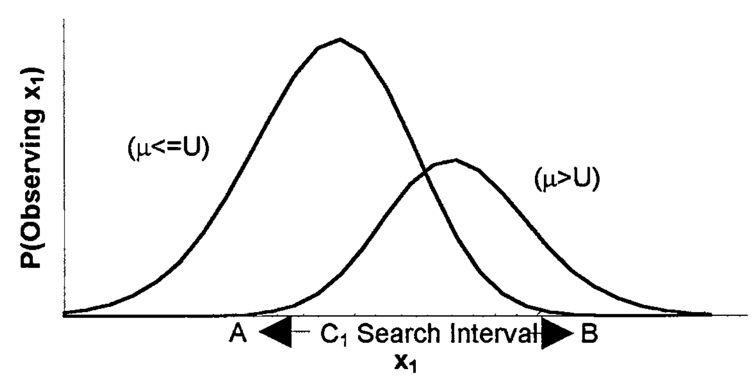

Figure 5.2 Example Curves for Conditional Probabilities of Observing x_1.

Location of the x_1 values for which the probability plots first approach zero is critical to operation of the FORTRAN routines provided, since actual solution of these probabilities is accomplished through ten-point Gaussian quadrature as described by Press, et al. [47]. Because the Nelder and Mead search routine may generate cutoff values outside of the practical limits A and B, a check is provided to kick any such values back and generate new ones within range.

The logic flow for the prior costing solution when $n_{max}=1$ is shown in Figure 5.3. This flowchart tracks the FORTRAN computer solution which is given in Appendix B. The programming logic for the Nelder and Mead subroutine is omitted from this discussion. Interested readers should see references [40] and [42].

When the tolerable n_{max} is increased from one to two, an additional cutoff value enters into the expected cost equation. Recall that the incremental step in n_{max} introduces an additional three terms into the cost equation (one each for sampling, false acceptance and false rejection). The false disposition probabilities for n=2 (regardless of n_{max}) require triple integrals since they must now also reflect the probability of requiring the second trial. Note, that in subsequent iterations, the acceptance and rejection terms closely resemble the base case of n=1. That is, the core function is bivariate normal, with the parameters of the sampling and prior sampling distributions changing to reflect the iteration number and previous observation information, respectively.

The probability that the second sampling cost is incurred is simply the probability that the first observation falls within the continuation interval ($C_{1,L}$ to $C_{1,H}$).

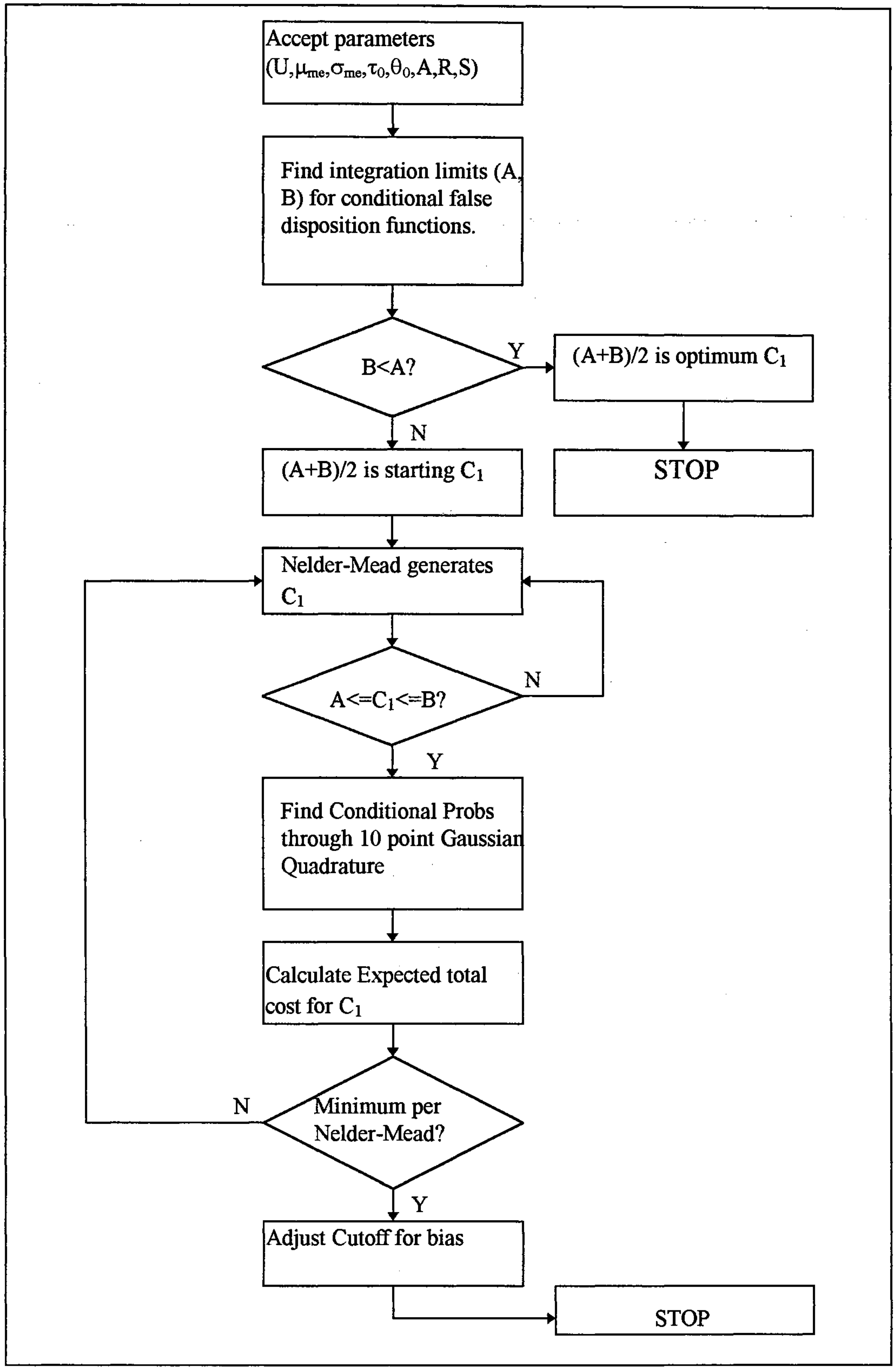

Figure 5.3 FORTRAN Flowchart for n_{max}=1, Prior Costing.

Sampling: $$\int_{C_{1,L}}^{C_{1,H}} \frac{1}{\sqrt{2\pi}\sqrt{\sigma_{me}^2+\tau_0^2}} \exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\sigma_{me}^2+\tau_0^2}}\right)^2\right] dx_1$$

The false disposition probabilities are in the form given, below.

$$\int_{C_{1,L}}^{C_{1,H}} \int\int \frac{1}{\sqrt{2\pi}\sigma_{me}} \exp\left[-\frac{1}{2}\left(\frac{x_2-\mu}{\sigma_{me}}\right)^2\right] \frac{1}{\sqrt{2\pi}\tau_1} \exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\tau_1}\right)^2\right] \times$$
$$\frac{1}{\sqrt{2\pi\left(\sigma_{me}^2+\tau_0^2\right)}} \exp\left[-\frac{1}{2}\left(\frac{x_1-\theta_0}{\sqrt{\sigma_{me}^2+\tau_0^2}}\right)^2\right] d\mu\, dx_2\, dx_1$$

Again, the limits of the second and third integral (over μ and x_2) depend on the specific disposition probability being considered. Because the marginal distribution on x_1 does not involve μ or x_2, it may treated separately. By doing so, the inner bivariate normal distribution (involving μ and x_2) may be reduced by integration over μ as in the n=1 case, and the double integral expression reduces to a single integral over x_2. The reduced expression is a conditional probability of false disposition on the second iteration given the actual location of μ relative to U.

The probability of false rejection on n=2 when the batch is conforming is given as

$$\int_{2C_{2,H}-x_1}^{B} -.19947\left[\mathrm{erf}\left[.70711\frac{-U\tau_1^2-U\sigma_{me}^2+\theta_1^2\sigma_{me}^2+x_2\tau_1^2}{\left[\sigma_{me}\left(\tau_1\sqrt{\tau_1^2+\sigma_{me}^2}\right)\right]}\right]-1\right]\times$$
$$\frac{\exp\left[-0.5\frac{(x_2-\theta_1)^2}{\tau_1^2+\sigma_{me}^2}\right]}{\sqrt{\tau_1^2+\sigma_{me}^2}} dx_2$$

and the corresponding probability of false acceptance for a non-conforming batch is

$$\int_{A}^{2C_{2,L}-x_1} -.19947\left[\operatorname{erf}\left[.70711\frac{-U\tau_1^2 - U\sigma_{me}^2 + \theta_1^2\sigma_{me}^2 + x_2\tau_1^2}{\left[\sigma_{me}\left(\tau_1\sqrt{\tau_1^2 + \sigma_{me}^2}\right)\right]}\right] - 1\right] \times$$

$$\frac{\exp\left[-0.5\dfrac{(x_2-\theta_1)^2}{\tau_1^2 + \sigma_{me}^2}\right]}{\sqrt{\tau_1^2 + \sigma_{me}^2}} dx_2 .$$

For the sake of further development, the cutoff values associated with the second iteration are left in the general notation of $C_{2,L}$ and $C_{2,H}$ (rather than C_2 for n_{max}=2). Note that these two conditional probabilities are in the exact same form as those previously shown for n=1 and may be solved through the same FORTRAN subroutine calls. However, the fact that the expressions involve a specific x_1 value (and they must be integrated over the x_1 continuation interval) appears to present a problem.

By making use of information obtained through the second solution approach (posterior costing) the problem is greatly simplified. As shown in the theoretical development section for the second approach, the probable location of x_2 is actually independent of x_1. This means that the values of the C_2 cutoffs are not dependent on the first observation, as is implied by the probability expressions; a single, optimal pair of ($C_{2,L}$, $C_{2,H}$) values (=C_2 for n_{max}=2) exists which optimize the expected total cost equation.

The additional, outer integration over x_1 presents an additional challenge. For the purpose of this research, a brute-force integration which simply steps through the ($C_{1,L}$,$C_{1,H}$) interval and uses a midpoint approximation is used to accomplish the outer leg of the double integral. The programming logic for the prior costing approach to the n_{max}=2 problem is shown in Figure 5.4.

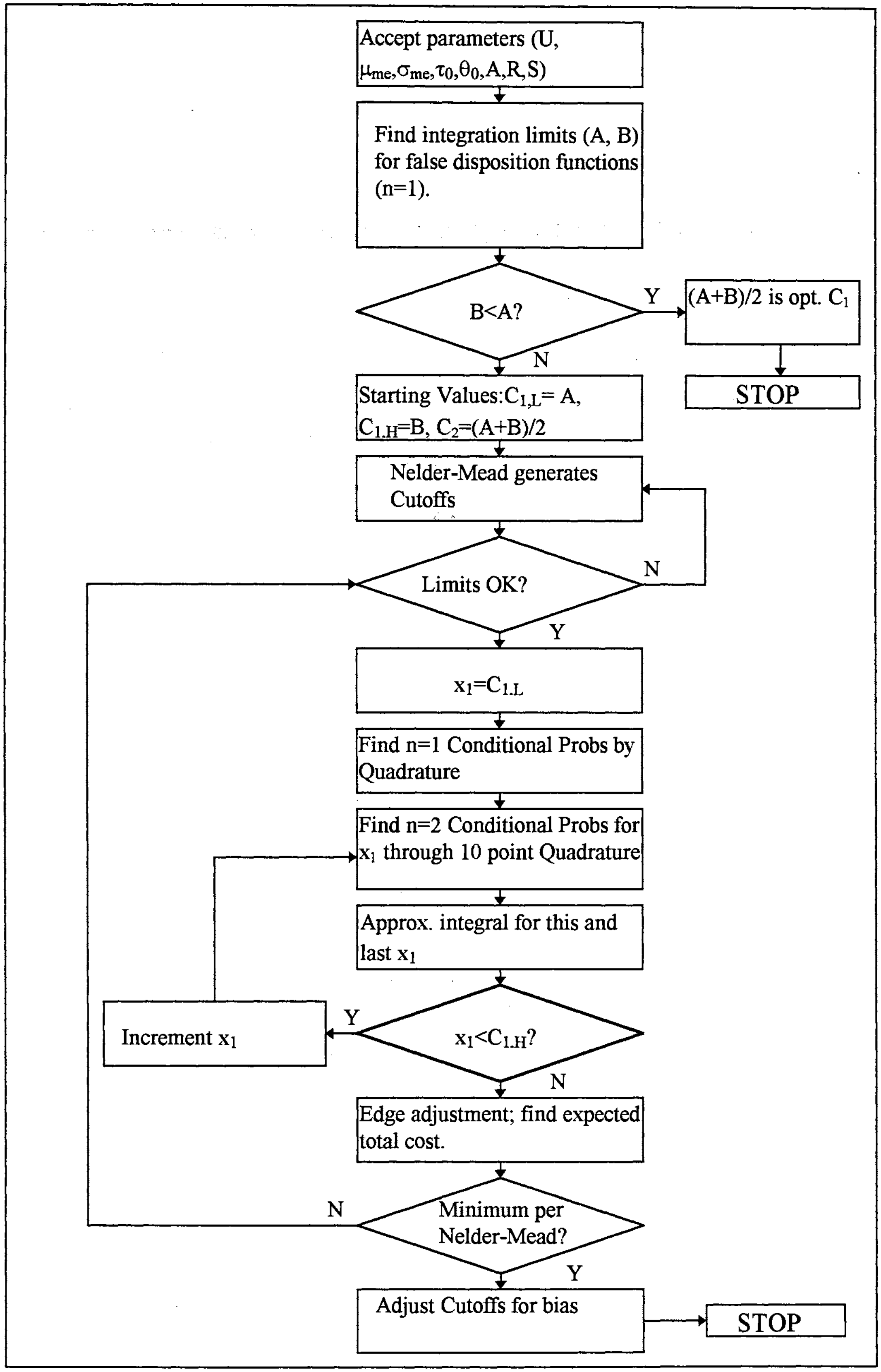

Figure 5.4 FORTRAN Flowchart for n_{max}=2, Prior Costing.

The limit checks which are performed also involve the integration limits on x_2, $(2C_2 \pm x_1)$. Because finite values (A, B) are used as substitutes for $\pm$ infinity for practical solution of the problem, it is possible that the limits involving C_2 may overstep these bounds. In order to avoid negative probabilities being generated by the quadrature subroutine, limit checks are performed prior to making the routine call.

Further increases in n_{max} compound very similarly to the n=1 and n=2 cases. That is, additional cost components (S, A and R) are introduced which reflect the probabilities of requiring the additional iteration(s). The core bivariate normal function is modified to reflect updated parameters, but continues to be solved through quadrature. However, the probability which is solved through quadrature involves early observations which are imbedded within the conditional functions. If the method used for $n_{max}=2$ is carried forward, this will require additional integration loops for early observations. Again, the knowledge gained from the posterior approach concerning independence of future and early observations allows the specification of all cutoff values prior to finding the expected total cost. That is, in the $n_{max}=3$ case, all five cutoffs ($C_{1,L}$, $C_{1,H}$, $C_{2,L}$, $C_{2,H}$, C_3) may be hypothesized prior to beginning the optimization search (rather than making a higher level cutoff dependent on early observations).

Approach 2 (Posterior Costing)

Due to the shortcut available for locating $C_{n_{max}}$ provided by the economic indifference ratio, the posterior costing approach provides an advantage over the prior approach. The ability to locate $C_{n_{max}}$ through a unidimensional search reduces the number of cutoff values which must be located through multivariate means. This reduces the runtime for the optimization computer program for a given n_{max}. For this reason, the posterior costing approach is utilized in the program of Appendix A to generate the research results.

In general, the number of search variables which must be found through the multivariate routine for the posterior costing methodology is given by [(n_{max}-1)*2].

For any $C_{n_{max}}$, the economic indifference point may be expressed as:

$$P(\mu > U | \bar{x}_{n_{max}} = C_{n_{max}}) =$$

$$\int_U^{\infty} \frac{1}{\sqrt{2\pi\tau_{n_{max}}}} \exp\left[-\frac{1}{2}\left(\frac{\mu - \dfrac{\dfrac{(n_{max})C_{n_{max}} - \sum_{i=0}^{n_{max}-1} x_i}{\sigma_{me}^2} + \dfrac{\theta_{n_{max}-1}}{\tau_{n_{max}-1}}}{\left(\dfrac{1}{\sigma_{me}^2} + \dfrac{1}{\tau_{n_{max}-1}^2}\right)}}{\tau_{n_{max}}}\right)^2\right] d\mu,$$

where x_0 may be considered as a null observation (in order to make the form general for the n_{max}=1 case). The normal distribution parameters θ_i and τ_i are as previously given in this writing.

The posterior function $h(\mu | x_1, \ldots, x_{n_{max}})$, given within the integral, may be simplified to a form which does not involve any of the previous observations, x_1 through $x_{n_{max}-1}$. This reduced form is

$$P(\mu > U | \bar{x}_{n_{max}} = C_{n_{max}}) =$$

$$\frac{\sqrt{n_{max}\tau_0^2 + \sigma_{me}^2}}{\sqrt{2\pi}(\sigma_{me} \cdot \tau_0)} \times$$

$$\exp\left[\frac{\left(n_{max}\mu\tau_0^2 + \mu\sigma_{me}^2 - n_{max}\tau_0^2 C_{n_{max}} - \theta_0\sigma_{me}^2\right)^2}{\tau_0^2 \cdot \sigma m_e^2} \frac{-1}{2(n_{max}\tau_0^2 + \sigma_{me}^2)}\right].$$

The final iteration cutoff value, $C_{n_{max}}$, is located through a unidimensional search using 10-point gaussian quadrature to solve the integral [47]. The search for optimal $C_{n_{max}}$ continues until this probability falls just short of the indifference cost ratio, R/(A+R).

An alternative solution method is taken in the FORTRAN programs in Appendix A which use the posterior costing approach. Leaving the probability equation in terms of the previous observations (although independent of them) and integrating over μ gives a closed-form solution for which limit substitution is straightforward (and quadrature is unnecessary). For any n_{max}, the probability may be written:

$$P(\mu > U | \bar{x}_{n_{max}} = C_{n_{max}}) =$$

$$\frac{1}{2}\left[1-\operatorname{erf}\left[\frac{0.70711}{\tau_{n_{max}}}U - 0.70711\frac{\left(\dfrac{(n_{max})C_{n_{max}} - \sum\limits_{i=0}^{n_{max}-1} x_i}{\sigma_{me}^2} + \dfrac{\theta_{n_{max}-1}}{\tau_{n_{max}-1}^2}\right)}{\tau_{n_{max}}\left(\dfrac{1}{\sigma_{me}^2} + \dfrac{1}{\tau_{n_{max}-1}^2}\right)}\right]\right],$$

where, again, x_0 is a null observation. The error function erf() is as previously defined in the section describing the prior costing approach [55].

Because this probability is independent of all previous observations, any x_i values may be substituted into the equation to obtain a probability for comparison with the cost ratio. In the FORTRAN routines of Appendix A, the value of the upper specification is used for all prior observations. The use of this method avoids the utilization of a quadrature routine for integration. However, the value of $C_{n_{max}}$ is still located through the use of a unidimensional search.

Note that, regardless of the solution method used, optimization of C_1 for $n_{max}=1$ does not require the minimization of an expected total cost equation through search methods. The single cutoff value may be found through the described unidimensional search and substituted into the expected total cost equation previously presented for the $n_{max}=1$ solution.

The posterior function which is integrated for the $C_{n_{max}}$ shortcut is also at the heart of the expected total cost equation for any $n_{max}>1$. The false disposition probabilities at any iteration, i, are in the form

$$\int\int \frac{1}{\sqrt{2\pi}\tau_{i-1}} \exp\left[-\frac{1}{2}\left(\frac{\mu-\theta_1}{\sigma_{me}}\right)^2\right] \frac{1}{\sqrt{2\pi\left(\tau_{i-1}{}^2+\sigma_{me}{}^2\right)}} \exp\left[-\frac{1}{2}\left(\frac{x_i-\theta_{i-1}}{\sqrt{\left(\tau_{i-1}{}^2+\sigma_{me}{}^2\right)}}\right)^2\right] d\mu dx_i$$

where the limits of the integrals depend on the disposition being considered (acceptance or rejection). These probabilities are general in that they are conditional upon reaching iteration i. In the actual expected total cost equation, the false disposition probabilities at iteration i must be multiplied by the probability of reaching that iteration. Recall that the integration limits for x_i involve the $C_{i,L}$ and $C_{i,H}$ cutoffs in addition to the previous observations x_1 through x_{i-1}.

Just as μ was integrated out of the expression for location of a given $C_{n_{max}}$, it may be eliminated from this conditional double integral expression. Making this simplification and substituting the appropriate limits for x_i gives the following general expression for the probability of false rejection at iteration i

$$\int_{i\cdot C_{i,H}-\sum_{j=0}^{i-1} x_j}^{B} \frac{1}{2}\left[\operatorname{erf}\left[\frac{0.70711}{\tau_i}U - 0.70711\frac{\left(\frac{x_i}{\sigma_{me}^{2}}+\frac{\theta_{i-1}}{\tau_{i-1}^{2}}\right)}{\tau_i\left(\frac{1}{\sigma_{me}^{2}}+\frac{1}{\tau_{i-1}^{2}}\right)}\right]+1\right]\times$$

$$\frac{1}{\sqrt{2\pi\left(\tau_{i-1}^{2}+\sigma_{me}^{2}\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_i-\theta_{i-1}}{\sqrt{\left(\tau_{i-1}^{2}+\sigma_{me}^{2}\right)}}\right)^{2}\right]dx_i$$

with the corresponding probability of false acceptance on the ith measurement observation given as

$$\int_{A}^{i\cdot C_{i,L}-\sum_{j=0}^{i-1} x_j} \frac{1}{2}\left[1-\operatorname{erf}\left[\frac{0.70711}{\tau_i}U - 0.70711\frac{\left(\frac{x_i}{\sigma_{me}^{2}}+\frac{\theta_{i-1}}{\tau_{i-1}^{2}}\right)}{\tau_i\left(\frac{1}{\sigma_{me}^{2}}+\frac{1}{\tau_{i-1}^{2}}\right)}\right]\right]\times$$

$$\frac{1}{\sqrt{2\pi\left(\tau_{i-1}^{2}+\sigma_{me}^{2}\right)}}\exp\left[-\frac{1}{2}\left(\frac{x_i-\theta_{i-1}}{\sqrt{\left(\tau_{i-1}^{2}+\sigma_{me}^{2}\right)}}\right)^{2}\right]dx_i .$$

The limits A and B are the practical limits ($\pm$ infinity, respectively) of the false disposition probabilities for the n=1 case. As in the prior costing approach, they are located through a unidimensional function search for the points at which the two conditional probability functions approach zero. Refer to Figure 5.2 and the accompanying discussion for a more detailed explanation of the A and B limits.

The solution procedure for the posterior approach is similar to the prior approach which was previously presented. That is, solution of double integrals is accomplished through a brute-force outer step integral, with the inner integral solved through 10-pt. gaussian quadrature. All cutoff values are hypothesized prior to beginning the Nelder and Mead search process, with appropriate boundary checks to ensure that the search routine has provided valid cutoffs prior to calculating the associated expected total cost.

A general computer logic flowchart is shown in Figure 5.5 which covers the computer solution provided in Appendix A which uses the posterior costing approach. FORTRAN routines are included for the specific cases of one, two and three maximum iterations (n_{max}). As previously indicated, the single observation case of $n_{max}=1$ utilizes only a step search using the economic indifference ratio as a probability comparison. For greater iteration thresholds, the required multiple integrations are performed through nested loops which step through the possible observation values defined by the various cutoff limits.

In explanation, for the case of $n_{max}=3$, the posterior approach integrates over observations x_1, x_2 and x_3. This is because the only way in which the third observation will be necessary is if the first two observations result in means which fall within the continuation regions defined by ($C_{1,L}$, $C_{1,H}$) and ($C_{2,L}$, $C_{2,H}$). However, cutoff values are compared to the mean of all observations, and cutoff values are put in terms of single observations for costing and integration purposes. Therefore, for expected costs incurred on the final iteration, an outer integration loop steps through x_1 values from $C_{1,L}$ to $C_{1,H}$, and an inner integration loop steps through x_2 values from $2C_{2,L}-x_1$ to $2C_{2,H}-x_1$. The third and innermost integral, involving the single cutoff value C_3 ($=C_{n_{max}}$), is solved through ten-point quadrature. Each time that the inner stepping loop completes a circuit, the inner loop reinitializes and the outer loop is incremented by a unit (as defined in the program).

The nested integrals which are called-out in the flowchart involve solving for the conditional false disposition probabilities at each iteration. The probabilities for all possible prior observations (point values) are accumulated into an integral for estimating the total cost of the sampling plan.

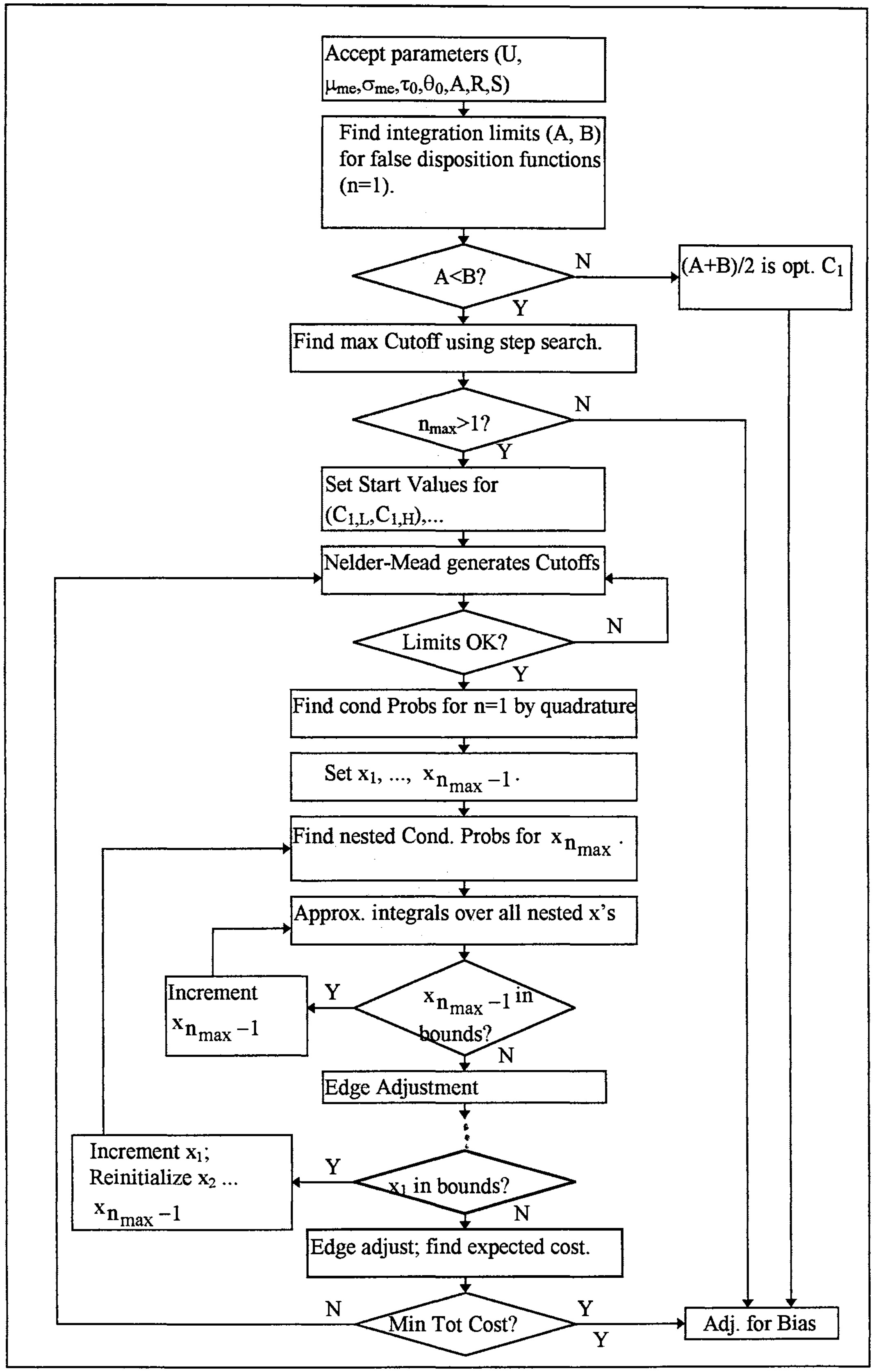

Figure 5.5 Flowchart for Approach 2, Posterior Costing.

Although the specific cases of n_{max} in the range one to three are presented in Appendix A (and in subsequent results provided in this paper), this same procedure may be extrapolated to greater values of n_{max}. The flowchart is intended to depict the general programming logic required to carry this approach to greater iteration thresholds.

For specific techniques utilized in the FORTRAN routines, please refer to the code provided in Appendix A. Embedded documentation is included in the FORTRAN code.

5.6 PROGRAM DESCRIPTION

The comprehensive FORTRAN program which implements the statistical and economic theories as presented in the current chapter and Chapter 3 is presented in Appendix A. Both models are accessible from a common main module (shown below as MAIN). The FORTRAN subroutines which implement the economic sequential theory as discussed in this chapter are summarized in the following paragraphs. Each heading represents the actual subroutine name as given in the FORTRAN code (without arguments). Further information on program operation and a description of subroutine arguments is contained within the body of the code (Appendix A) in the form of program comments.

The program module summaries which are presented in the following paragraphs pertain only to the economic branch of the computer program (see Figure 3.2). For information on the statistical branch of the program, the user is directed to Section 3.5 of this document.

As previously indicated, verification of program logic is accomplished through redundant runs using hand calculations, the FORTRAN simulators (Appendix D), Microsoft Excel 5.0[2] and Mathcad 4.0.[3]

CALCST

Calculates the expected costs associated with a user-designed economic sequential sampling plan. Accepts user-specified decision cutoff values and gives all expected cost components. Calls subroutines: VARDEF, ERRCHK1, ERRCHK2. Called from: MAIN

VARDEF

Accepts all measurement system and prior distribution parameters and returns them to the parent module. Calls subroutines: none. Called from: CALCST, A2N1MGT, A2N2MGT, A2N3MGT.

A2N1MGT

Optimizes the economic decision cutoff value C_1 for the case of $n_{max}=1$. Calls subroutines: VARDEF, SETABX, CMAXN1, COST. Called from: MAIN.

A2N2MGT

Optimizes the economic decision cutoff values $C_{1,L}$, $C_{1,H}$ and C_2 for the case of $n_{max}=2$. Calls subroutines: VARDEF, POST, SETABX, CMAXN2, NELMIN1. Called from: MAIN.

---

[2] *Microsoft Office Professional v. 4.3* (1993), Microsoft Corporation, USA.

[3] *Mathcad 4.0 User's Guide Windows Version* (1993), Mathsoft, Inc, Cambridge, MA.

FUNC

Evaluates the bivariate normal probability of μ and x, over a given range of μ relative to the specification. Calls subroutines: PCMU. Called from: QGAUS, SETABX, SETMOR.

A2N3MGT

Optimizes the economic decision cutoff values $C_{1,L}$, $C_{1,H}$, $C_{2,L}$, $C_{2,H}$ and C_3 for the case of n_{max}=3. Calls subroutines: VARDEF, SETABX, CMAXN3, NELMIN1, CLG3OUT. Called from: MAIN.

CLG2A2

Performs the integration of false disposition probabilities and calculates expected costs between $C_{1,L}$ and $C_{1,H}$ for n_{max}=2. Calls subroutines: POST, SETABX, QGAUS, NORMAL. Called from: NELMIN1.

CLGIN

Performs the integration of false disposition probabilities and calculates costs between $C_{2,L}$ and $C_{2,H}$ for n_{max}=3. Calls subroutines: POST, SETABX, QGAUS, NORMAL. Called from: CLG3OUT.

CLG3OUT

Performs the integration of false disposition probabilities and calculates costs between $C_{1,L}$ and $C_{1,H}$ for n_{max}=3. Calls subroutines: POST, SETABX, QGAUS, CLGIN, NORMAL. Called from: NELMIN1, A2N3MGT.

CMAXN1

Optimizes C_1 for the case of n_{max}=1 by stepping through C_1 values in the interval (A, B) and evaluating the cost indifference ratio R/(R+A). Calls subroutines: PCMU. Called from: A2N1MGT.

CMAXN2

Optimizes C_2 for the case of n_{max}=2 by stepping through C_2 values in the interval (A, B) and evaluating the cost indifference ratio R/(R+A). Calls subroutines: PCMU. Called from: A2N2MGT.

CMAXN3

Optimizes C_3 for the case of n_{max}=3 by stepping through C_3 values in the interval (A, B) and evaluating the cost indifference ratio R/(R+A). Calls subroutines: PCMU. Called from: A2N3MGT.

NORMAL

Computes normal areas and ordinates for an array of x values. Taken from [12]. Calls subroutines: none. Called from: ERRCHK2, ERRCHK3, CLG2A2, CLGIN, CLG3OUT.

PCMU

Evaluates the expected total cost equation using observations passed from the parent module. Type of false disposition (accept or reject) is specified through a flag passed as the second argument. Calls subroutines: ERRINT. Called from: FUNC, CMAXN1, CMAXN2, CMAXN3.

POST

Calculates the mean of a normal posterior distribution, given the prior distribution parameters. Calls subroutines: none. Called from: ERRCHK2, ERRCHK3, A2N2MGT, CLGIN, CLG3OUT, CLG2A2.

QGAUS

Performs 10-point Gaussian quadrature. Taken from [47]. Calls subroutines: FUNC. Called from: COST, ERRCHK1, ERRCHK2, ERRCHK3, CLG2A2, CLGIN, CLG3OUT.

SETABX

Finds the practical infinity limits for the conditional probability curves in order to evaluate probabilities by Gaussian quadrature. Calls subroutines: FUNC. Called from: ERRCHK1, ERRCHK2, ERRCHK3, A2N1MGT, A2N2MGT, A2N3MGT, CLG2A2, CLGIN, CLG3OUT.

SETMOR

Finds practical infinity limits for the conditional probability curves in order to evaluate probabilities by Gaussian quadrature. These are the other extremes of the curves as located by SETABX. Calls subroutines: FUNC. Called from: ERRCHK1, ERRCHK2, ERRCHK3.

ERRINT

Evaluates the error function $\mathrm{erf}(v) = \frac{2}{\sqrt{\pi}} \int_0^v \exp\left[-t^2\right] dt$. Taken from [55]. Calls subroutines: none. Called from: PCMU.

COST

Evaluates the expected cost components for the economic case of $n_{max}=1$. Calls subroutines: QGAUS. Called from: A2N1MGT.

NELMIN1

Searches for economic cutoff values through multivariate simplex search procedure given in [40]. Calls subroutines: CLG2A2, CLG3OUT. Called from: A2N2MGT, A2N3MGT.

ERRCHK1

Calculates expected costs for the economic case of $n_{max}=1$. Checks bounds for user-specified cutoff parameters. Calls subroutines: SETABX, SETMOR, QGAUS. Called from: CALCST.

ERRCHK2

Calculates expected costs for the economic case of $n_{max}=2$. Checks bounds for user-specified cutoff parameters. Calls subroutines: POST, SETABX, SETMOR, QGAUS, ERRCHK3, NORMAL. Called from: CALCST.

ERRCHK3

Calculates expected costs for the economic case of $n_{max}=3$. Checks bounds for user-specified cutoff parameters. Calls subroutines: POST, SETABX, SETMOR, QGAUS, ERRCHK3, NORMAL. Called from: ERRCHK2.

5.7 SUMMARY

The economic optimization of the batch inspection problem requires explicit definition of the cost consequences associated with the sampling plan. Costs of the plan fall into two categories: 1) sampling costs associated with the measurement process, and 2) costs incurred as a result of disposition of the batch.

Disposition costs are assessed as a consequence of false acceptance or false rejection of the batch. Inappropriate acceptance of a non-conforming batch is the greater of the evils and may carry large, hard to quantify costs such as loss of repeat customers and customer goodwill.

In designing the economically optimal sampling plan, consideration is given to prior batch history. Bayesian decision theory methods are utilized to incorporate available data into the sampling design. The distribution of prior batches (assumed to be normal) is updated following each sequential measurement observation. Because the measurement error distribution and the prior batch distribution are assumed normal, all required distributions are also normal. This is because the normal is a family of conjugate priors.

The economic model is designed with a maximum allowable iteration number, designated by the user. At each step of the sequential process, the mean of all previous observations is compared to a pair of decision cutoffs for that observation. For the upper specification problem, if the mean is less than the low cutoff, the batch is accepted and the process terminates. If the mean is greater than the high cutoff, the batch is rejected. Inequality signs are reversed for the lower specification case. A mean observation value which falls between the two disposition cutoffs dictates continuation of the inspection procedure. If the process reaches the designated maximum number of iterations, the two decision limits are replaced by a single cutoff (the two limits are equal), forcing a disposition decision at that step. For practical applications addressed in the research problem, a maximum iteration value of three is reasonable and will prove to be widely applicable.

Compensation for measurement system bias is made following a zero-bias system design. Because a non-zero measurement error mean represents a constant average offset of observed values from actual values, it may be added to zero-bias cutoffs following optimization. In this way, it is possible to avoid carrying the bias through analysis. Additionally, the zero-bias system which is provided as computer output is easily updated for any change in the bias.

Given the costs, prior distribution, measurement error distribution, specification (upper) and maximum number of iterations acceptable, the economic approach optimizes the cutoff variables at each iteration such that the expected total cost equation is minimized.

Two approaches may be taken in designing the plan and defining the expected total cost equation. The first approach looks at costs in terms of the Bayesian prior distribution; the second approach takes a posterior costing view. Both analysis alternatives lead to the same plan design.

The solution for a lower specification limit may be found through symmetry from an appropriate upper specification limit solution. By treating the lower limit as an upper limit and translating the prior distribution mean symmetrically around the specification, optimal cutoffs are found which can then be translated, by symmetry, into optimal cutoffs for the lower limit solution.

For any maximum n (n_{max}) greater than one, economic optimization using either approach requires a multivariate search for the cutoff values. The FORTRAN programs written for this research utilize the Nelder and Mead direct search routine for finding the minimum of the function. The posterior costing approach provides a slight advantage over the prior design in that the final iteration cutoff value, $C_{n_{max}}$, may be found by unidimensional search prior to beginning expected cost equation minimization. This reduces the number of cutoff values which must be located through the multiple search routine by one, saving some computer run time. Research results are generated using the posterior costing logic as given in Appendix A.

Programming examples (FORTRAN) are provided in the appendices for each approach to the economic design problem. Prior cost programs are included in Appendix B for values of n_{max} designated as one and two. The comprehensive program in Appendix A contains an additional FORTRAN routine for the case of $n_{max}=3$ using the posterior costing approach. The logic which is utilized in the programs is representative of a single solution approach to the economic design problem, and is not necessarily the most efficient. The programs utilize ten point gaussian quadrature and a step-through integration routine in solving the multiple integral terms which the expected total cost equation comprises. Compensation for measurement system bias is performed by the program following zero-bias cutoff optimization.

CHAPTER 6

OPERATION OF THE INTERACTIVE COMPUTER ROUTINES FOR THE ECONOMIC SOLUTION

6.1 INTRODUCTION

This chapter details the operation of the interactive computer program modules which implement the solution of the known measurement variance economic problem for various values of iteration maxima. The comprehensive program (Appendix A) provides two separate options for examining economic sampling plans for the homogeneous batch disposition by a single variable characteristic relative to a single specification limit. The first option provides the optimal economic cutoff values for the economic sampling procedure as described in Chapter 5. This optimization routine utilizes the posterior costing approach as previously presented and is available for maximum observation values (n_{max}) of one, two and three. The posterior approach is utilized due to increased efficiency and reduced computer runtime associated with this costing solution as presented in the previous chapter. The second economic option takes specific cutoff values from the user (again, n_{max}= 1 to 3) and returns the expected total cost of the user-input sampling plan. It also uses the posterior costing approach to the problem. Each of the modules is executable for both a single upper and a single lower specification limit. The solution logic and methodology is as presented in the preceding chapter. The actual FORTRAN program (composed and executed on an IBM-compatible personal computer using Microsoft FORTRAN version 5.1) appears in Appendix A.

The computer routines are interactive and prompt the user for the required input parameters. Output from the programs includes all cutoff decision values and expected cost components of the plan. In all cases, program output is provided on-screen with optional output to a user-specified computer file.

The same input subroutine is used for all n_{max} values and both costing options (optimization and cost estimation). Therefore, all routines present themselves in exactly the same manner to the user. The output format is also very similar, with differing n_{max} programs varying only in the number of cost components and cutoff values which are presented.

Error checks are performed for user-provided input parameters. All input values are presented for operator verification prior to beginning execution of the FORTRAN routines.

6.2 PROGRAM OPERATION

Each of the optional economic programs (optimization and cost estimation) is accessed from the main menu. The user enters the first option ("1") as shown, below.

```
Sequential Testing Program

Please select one of the following options:

  1  Economic Testing
     *  Plan Optimization
     *  Expected Cost Calculation

  2  Statistical Testing
     *  Known Measurement Error Variance
     *  Unknown Measurement Error Variance

  3  Exit Program

1
```

The secondary menu for economic testing presents the following options:

```
          Economic Testing Plans

Please select one of the following options

  1  Economic Parameter Optimization
     (Maximum Iterations Limited to Three)

  2  Expected Costs Calculation for
      User-Entered Plan
     (Maximum Iterations Limited to Three)

  3  Return to Main Menu
```

Access to the two economic options is provided through entry of a "1" or "2" at the prompt. An invalid entry brings up the following error message:

```
**** Invalid Entry.  Please Reenter. ****.
```

A user entry of "1" begins execution of the optimization routine for economic analysis of the sequential problem. This program operation is described in the following section of the chapter.

The alternative entry of "2" initiates the expected cost calculation routine which is described in the chapter section entitled "Expected Costs Calculation for User-Entered Plan (Option 2)".

Economic Parameter Optimization (Option 1)

This portion of the sequential testing program provides the optimal economic testing plan, based on the user-input distribution and cost parameters. Optimal plans are available for maximum observation values of 1 to 3, as requested by the program user. A prompt first appears requesting the maximum number of iterations (observations) which the user wishes to analyze:

```
What is the maximum number of measurement
iterations which you are willing to make?
Enter 1, 2 or 3

2
```

The user has entered a "2", indicating that he/she wishes an optimal plan for which a disposition decision is reached in a maximum of two measurement observations.

The next series of prompts request entry of the testing system distribution and cost parameters. This portion of the computer program is identical in all cases of n_{max} (= 1 to

3). The first prompt is for the specification limit which defines conformance of the variable batch characteristic being examined.

```
Enter the Specification Limit.

102.
```

The user has entered the value "102.". There are no restrictions on this input testing parameter. The user is then asked to indicate if this specification limit is an upper or lower specification. The following prompt is presented:

```
Is this an Upper (1) or Lower (2) Spec?
Enter 1 or 2.

1
```

The user has entered a "1" in the example, defining the specification entry of "102." as an upper specification limit.

The program then proceeds with the entry of parameters of the prior distribution of actual batch values. The request for the prior mean is:

```
Enter the value of the prior distribution mean.

100.5
```

This is followed by a prompt for the prior standard deviation:

```
Enter the value of the prior standard deviation.

.5
```

As inputted by the user, the prior distribution (assumed normal) has a mean of "100.5" and a standard deviation of ".5". There are no restrictions on the prior distribution mean, but the standard deviation must be positive.

The next section of code requests information concerning the measurement error system parameters. Again, this distribution is assumed to be normal. The mean of the error distribution, also termed the bias, is requested in the first prompt:

```
Enter the value of the measurement error
distribution mean (bias).
Sign Convention:  If the instrument reads higher
than the true value, this bias should be positive.

.08
```

This is followed by the prompt for the measurement error standard deviation:

```
Enter the value of the measurement error
distribution standard deviation.

.5
```

The value of the bias is unrestricted, but a negative entry for the measurement error standard deviation produces the error message which was previously shown.

The last program entries which are required for each module concern the costs which are associated with the measurement of the batch characteristic and disposition of the batch. As a matter of convention, all costs are required to be positive (or zero).

```
Enter the cost associated with a single measurement
iteration (S).

.25

Enter the cost associated with a false acceptance of
a batch of product (A).

100
```

```
Enter the cost associated with false rejection of
a batch of product (R).

20
```

The user has entered values of ".25", "100" and "20" for the iteration, false acceptance and false rejection costs, respectively. As explained in the theoretical development of the economic case, the acceptance of non-conforming product is generally much more costly than the rejection of a conforming batch. A negative entry for any one of the three sampling system costs produces an error message,

and the cost prompt is displayed, again.

Following this last parameter entry, the program values are displayed for review. The user is given the opportunity to change any of the parameters, although only one parameter may be changed at a time. In the following illustration, the user takes the opportunity to modify the bias.

```
1  Upper Specification Limit=                  102.0000
2  Prior Distribution Mean=                     100.5000
3  Prior Standard Deviation=                       1.5000
4  Error Distribution Mean (Bias)=               .0800
5  Error Distribution Std. Dev.=                    .5000
6  Iteration Cost (S)=
.25
7  False Acceptance Cost (A)=                 100.00
8  False Rejection Cost (R)=                      20.00

Is the above information correct?
Enter to accept, or # of parameter to reenter.
4
```

The program then prompts for reentry of the parameter specified by the user:

```
Enter the value of the measurement error
distribution mean (bias).
Sign Convention:  If the instrument reads higher
than the true value, this bias should be positive.
```

```
.06
```

The user has entered ".06" as a correction to the previously entered value of the measurement error mean (bias). The same error checks which were previously given are also performed for any parameter modifications. Following any corrections, the parameters are displayed in summary form and corrections may again be made.

After all parameters are entered correctly (per the user), the program begins optimization of the sampling plan which was requested (based on the n_{max} value which was entered). The runtime required for execution varies, of course, among the differing values of n_{max}. As previously indicated, the posterior costing approach provides an advantage over the prior approach in the location of the cutoff value at the iteration maximum. This reduces the computer time required for the multivariate search routine. However, the program sequence executed for the case of n_{max}=3, although programmed using the posterior economic theory, requires a relatively large interval of time for economic optimization of plan parameters. As a benchmark value, a typical run requires approximately 68 minutes on a Pentium 60. Execution messages are displayed for the integration loops for the case of an observation maximum of three message to restrain the user from slipping into a panic and contemplating a reboot.

The sampling system design is presented in summary form on-screen as soon as the optimization is complete. As an example of program output, following are the results generated by the entry of the above input examples into the posterior approach program for the case of n_{max}=2.

```
Zero-Bias C2 =       101.73191545      Bias Adj C2 =
101.67191545
Zero-Bias C1,L =      101.03840890      Bias Adj C1,L =
100.97840891
Zero-Bias C1,H =      102.48857274      Bias Adj C1,H =
102.42857275
Expected Plan Costs:
   Sampling on 1 =               .2500
   False Accept on 1 =           .0744
   False Reject on 1 =           .1609
   Sampling on 2 =               .0656
   False Accept on 2 =           .5905
   False Reject on 2 =          1.0361
Expected Total Cost =           2.1776

Send output to file? (Y/N)

Y
```

After the output display, a prompt is provided which allows the measurement system design to be written to a hard-disk file. This provides an opportunity to save and/or print the sampling plan following the program utilization. The output which is sent to file is similar to, but more complete than, the output which is displayed on the monitor. It also includes the input parameters in the form of header information. The user entry of "Y" brings up the following prompt for the file name to be used:

```
File name missing or blank - please enter file name
UNIT 2?
nmx2.op
```

The file name shown, "nmx2.op", is entirely at the user's discretion and must conform only to the naming conventions of the operating system being utilized. The following example file represents program output from the prior approach program for the case of n_{max}=2.

```
Prior Mean =                    100.50000000
Prior Std. Dev =                  1.50000000
Upper Specification =           102.00000000
Measurement Error Std. Dev. =            .50000000
Measurement Error Mean (Bias) =          .06000000
Input Costs:
   Iteration (S) =                 .2500
   False Accept (A) =           100.0000
   False Reject (R) =            20.0000
************************************
Zero-Bias C2 =        101.73191545     Bias Adj C2 =
101.67191545
Zero-Bias C1,L =      101.03840890     Bias Adj C1,L =
100.97840891
Zero-Bias C1,H =      102.48857274     Bias Adj C1,H =
102.42857275
Expected Plan Costs:
   Sampling on 1 =                 .2500
   False Accept on 1 =             .0744
   False Reject on 1 =             .1609
   Sampling on 2 =                 .0656
   False Accept on 2 =             .5905
   False Reject on 2 =            1.0361
Expected Total Cost =             2.1776
```

The summary header information is the only additional material which is provided to an output file (as opposed to the screen output). The measurement system parameters (the cutoff values C_2, $C_{1,L}$ and $C_{1,H}$) are as previously defined in Chapter 5. These are the decision limits which are compared to the average value of the measurement observations in order to determine disposition of the batch (the procedure is fully described in Chapter 5). The cost components are broken down into sampling, false acceptance and false rejection costs for each observation stage of the plan. This allows the user to examine at which point in the sampling procedure the brunt of the cost is likely to be incurred, based on the optimal cutoff values presented. This cost breakdown is useful for comparison purposes if the user also intends to experiment with various cutoff value combinations of his/her own design as described in the section of this chapter devoted the calculation of expected costs for user-entered plans (option "2" from the economic testing menu).

The program presentation for plan optimization for other values of maximum observation value (n_{max} of 1 or 3) is exactly the same as presented for n_{max}=2. The only difference appears with the change in output as cutoff values and cost components are added/dropped for changing n_{max}. As an example, the following output represents the hard-disk file which is created for the above parameter entry (distribution, costs, etc.), requesting a single observation plan (n_{max}=1) rather than the example plan of n_{max}=2 which was detailed in the previous paragraphs.

```
Prior Mean =                    100.50000000
Prior Std. Dev =                  1.50000000
Upper Specification =           102.00000000
Measurement Error Std. Dev. =        .50000000
Measurement Error Mean (Bias =       .06000000
Input Costs:
   Iteration (S) =                .2500
   False Accept (A) =          100.0000
   False Reject (R) =           20.0000
**************************************
Zero-Bias C1 =      101.65680177     Bias Adj C1 =
101.59680177
Expected Plan Costs:
   Sampling on 1 =                .2500
   False Accept on 1 =            .9896
   false Reject on 1 =           1.6688
Expected Total Cost =            2.9083
```

Note the identical format and header information. The difference appears only in the change in cutoff value(s) given and the breakdown of cost components as dictated by the change in maximum observation value between plans.

To return to the main menu from the economic testing menu, the third option ("3 Return to Main Menu") is selected. An entry of "3" at the main menu prompt causes termination of the entire sequential testing program.

Expected Costs Calculation for User-Entered Plan (Option 2)

The purpose of this program module is to allow the user to examine the expected cost consequences of any measurement plans which he/she designs. That is, rather than selecting the optimal cutoff plan, the user may desire to select and enter alternative cutoff values and examine the changes in expected costs. This option is useful in the situation that alternative cutoff values may be easier to implement than those specified in the optimal plan. Additionally, it may be a practical matter to round the optimal cutoff values prior to implementation of the plan; this program option allows the user to examine the cost consequences of the rounding of cutoff values.

Although the measurement system bias (mean of the error distribution) is requested as input in this program module, it is not utilized in the cost calculations. This is due to an assumption that all of the cutoff values which are entered by the user have the measurement bias built into them, as will the measurement observations to which they are compared. The bias is requested as standard entry by the parameter routine which is used by this and other program modules.

From the economic testing menu, the cost calculation routine is accessed by specifying option "2". An explanatory header appears to introduce the program module and signal the start of parameter entry.

```
This program module calculates the expected
total cost of a given sequential sampling
plan.  The user must supply the sequential
decision cutoff values.

Measurement System Parameter Entry:
```

The program proceeds with the user entry of the specification limit (upper or lower), prior mean, prior standard deviation, bias, measurement error standard deviation, and all costs as presented in the previous section of this chapter (not repeated, here). Following entry of all necessary parameters and verification by the user, the following prompt appears:

```
What is the maximum number of iterations'
for the plan? (1, 2 or 3)'

3
```

This request is for the observation maximum (n_{max}), and is limited to 1, 2 or 3. In the example, the user has specified that a disposition decision is desired within three measurement observations. Specification of n_{max} also determines the number of cutoff values which are required as entry for cost estimation. That is, because the user has specified an n_{max} of three, he/she will be required to enter values for $C_{1,L}$, $C_{1,H}$, $C_{2,L}$, $C_{2,H}$ and C_3. A maximum observation value other than 1, 2 or 3 is flagged as an invalid entry.

The program then begins the prompt/entry sequence for the cutoff values.

```
Enter the value of C1,L
101.2
Enter the value of C1,H
103.1
Enter the value of C2,L
101.5
Enter the value of C2,H
102.
Enter the value of C3
102.
```

Error checks are performed throughout this entry sequence to ensure that $C_{i,L} \leq C_{i,H}$ for all i. Although the example sequence additionally conforms to the inequality relationship $C_{1,L} \leq C_{2,L} \leq C_3 \leq C_{2,H} \leq C_{1,H}$, this is not necessary for calculation of plan cost components. That is, it is allowable for $C_{2,L}$ to be less than $C_{1,L}$ or C_3 to be the lowest of all cutoff values (as examples).

After all required cutoff values are entered, the information is displayed for user review and corrections, if required. The following data reflects entry of the cutoff values given in the input sequence, above.

```
Maximum iterations = 3
C1,L =     101.2000      C1,H =     103.1000
C2,L =     101.5000      C2,H =     102.0000
C3 =       102.0000

Is the above information correct?
Y to accept or N to reenter cutoff values.

Y
```

The user has entered a "Y", indicating that the entries shown are correct. An entry of "N" reinitiates entry of all cutoff parameters with the associated error checks for inequality relationships.

The program then begins the calculation of all cost components associated with the user-input sampling plan. As in the optimization case, output is displayed on-screen with the option of sending the cost information with all header data to a file located on the computer hard-disk. The following output data represents the output written to a user-specified file. For this example, the distribution parameter and cost data are as given in the economic plan optimization section of the chapter, with the user-specified cutoff values given, above.

```
Prior Mean =                    100.50000000
Prior Std. Dev =                  1.50000000
Upper Specification =           102.00000000
Meas. Error Std. Dev. =            .50000000
Meas. Error Mean (Bias) =          .06000000
Input Costs:
   Iteration (S) =                 .2500
   False Accept (A) =           100.0000
   False Reject (R) =            20.0000
```

```
************************************
Cutoff C3 =                   102.00000000
Cutoff C2,L =                 101.50000000
Cutoff C2,H =                 102.00000000
Cutoff C1,L =                 101.20000000
Cutoff C1,H =                 103.10000000
Expected Plan Costs:
  Sampling on 1 =                .2500
  False Accept on 1 =            .1643
  False Reject on 1 =            .0087
  Sampling on 2 =                .0697
  False Accept on 2 =            .1224
  False Reject on 2 =            .5098
  Sampling on 3 =                .0214
  False Accept on 3 =            .9996
  False Reject on 3 =            .0971
Expected Total Cost =           2.2430
```

The format for both other cases of maximum observations (one or two) is the same as the n_{max}=3 situation shown.

Following display of the program output and optional writing to a user-specified hard-disk file, the program asks if further plans are to be examined. The literal prompt is:

```
Would you like to input another set of cutoffs?
Enter Y or N.
```

N

The user has entered "N", indicating that no further analysis is desired. An entry of "Y" takes the user back to entry of the maximum desired number of measurement observations and cutoff values. The entry of further cutoff values is assumed to be for the same parameter system (prior, measurement error, costs, etc.) which was previously analyzed. If an entirely different disposition situation is to be examined, the user must return to the economic testing menu and reenter the expected cost calculation routine to prompt parameter input.

To return to the main menu from the economic testing menu, the third option ("3 Return to Main Menu") is selected. An entry of "3" at the main menu prompt causes termination of the entire sequential testing program.

6.3 SUMMARY

Two separate options are provided in the form of interactive FORTRAN program routines for the economic solution of the known measurement error variance sequential testing problem. The first option allows the user to solve for the optimal cutoff values which minimize the expected total cost of the measurement plan. The second option allows the user to pick his/her own set of cutoff values and obtain a breakdown of the expected total cost. Both options are available for a single upper or lower specification limit and maximum observation values (n_{max}) of 1 to 3. In each case, the user provides inspection system parameters and specifies the maximum number of measurement iterations which are to be taken. Expected costs are presented in component form by sampling, false acceptance and false rejection at each iteration (observation) stage. Upon user request, the resultant cost and sampling plan output from either FORTRAN routine is written to a hard-disk file (in addition to appearing on-screen). Error checks are performed on all input parameters to ensure consistency with theoretical solution constraints. The procedures of the computer programs are consistent with the sequential statistical solutions as presented in the previous chapter.

CHAPTER 7

COMPARISON OF RESULTS

7.1 INTRODUCTION

The economic modeling of the sequential sampling problem provides distinct cost advantages over the statistical modeling of the situation. Whereas the statistical case utilizes subjective, nebulous tolerable risks (α and β) to assess the consequences of making an incorrect disposition decision, the economic model forces the inspector/designer to explicitly assign monetary consequences to the plan in terms of sampling, false acceptance and false rejection costs. The costs which are incurred through utilization of the statistical sampling procedure are not treated explicitly in designing the plan.

Within the economic modeling problem, additional observations carry some sampling cost, S, but provide a return in the form of increased confidence in the resulting disposition decision (and reduced expected costs of incorrect disposition of the batch of product). If the testing is costly and S outweighs the cost(s) of an incorrect decision, the optimal economic plan will simply dictate that no additional observations be made.

The differences between the two sampling methods (statistical and economic) make it difficult to conduct a direct comparison of results. The two approaches to the problem carry differing assumptions and, essentially, use different sets of rules by which to assess the batch of product. The economic model explicitly treats the costs of the plan and also considers prior batch information in the determination of the optimal sampling plan. The statistical model requires that the user designate indifference limits which are used, instead of the specification limit, for batch disposition.

The statistical theory addresses the risks of false batch disposition by requiring specification of risk levels, α and β. However, recall that these risk levels are associated with indifference limits (subjectively specified by the inspector or plan designer) which cannot be equal and may or may not coincide with the specification limit against which the batch is being tested. Therefore, it is practically meaningless to compare these designated risk values to calculated risk levels of the economic plan which are relative to the specification limit. In a sense, the indifference limits specified in the statistical plan also must be considered as indicative of a level of risk which the user is willing to assume.

The most logical comparison to conduct when contrasting the two theoretical methods involves the assessment of the explicit economic costs to the statistical disposition. That is, although the statistical plan does not consider the sampling costs a priori, it is reasonable to assign these costs of false disposition to the statistical decision after the fact in assessing the performance of the statistical decision. Recall that the sampling costs which determine the optimal economic plan are explicit and represent the true risks of the inspection system.

In the following presentation and discussion, the modeling results are shown within the economic and statistical approaches. The various impacts of changing system parameters are explored for each theoretical case. In addition, an attempt is made to contrast the two methods of sequential batch inspection and disposition through computer simulation.

Verification of FORTRAN program optimization output is accomplished through parallel use of simulation (Appendix D), MathCad 4.0[1], Microsoft Excel 5.0[2] and hand calculations.

7.2 ECONOMIC MODEL

Economic Methodology

In order to assess the performance of the economic model, several different factors must be considered. The expected total cost of a given economic sequential sampling plan varies with the prior distribution, sampling and false disposition costs and the maximum number of iterations specified by the designer.

To illustrate the changes in cost components, the following example is utilized in all results presented in this section. The sampling system parameters are consistent with those

---

[1] *Mathcad 4.0 User's Guide Windows Version* (1993), Mathsoft, Inc, Cambridge, MA.

[2] *Microsoft Office Professional v. 4.3* (1993), Microsoft Corporation, USA.

utilized in previous sections of this paper to illustrate the operations of the various computer program modules presented. Units on all measurement system parameters and inspection costs are omitted.

Consider a batch production operation in which a single variable characteristic is utilized to dictate batch disposition. The characteristic is judged against a single upper specification limit. Given:

Upper Specification = 102.0
Prior Standard Deviation = 1.5
Measurement Error Standard Deviation = 0.5
Measurement Error Mean (Bias) = 0.0

Three sets of inspection system costs are examined in assessing the plan performances. Cost Set #1 may be considered the most realistic of the three in that the largest cost is incurred upon acceptance of a non-conforming batch. As discussed in a previous section of this paper, costs associated with false acceptance of a batch are often difficult to quantify and may include warranty costs, repair/replacement costs, loss of customer goodwill, lawsuits, etc. The three cost sets are shown in Table 7.1.

For purposes of comparison, the influence of the prior distribution is examined by varying the prior mean (θ_0) over a range around the upper specification. Values of the prior mean in the range 96.0 to 108.0 are used for optimization of the economic sampling parameters.

Note that in all three cases, the costs of false disposition far outweigh the costs of making an additional observation.

Table 7.1
Three Cost Sets Used for Economic Program Comparisons.

	Cost Set 1	Cost Set 2	Cost Set 3
Sampling Cost (S)	0.25	0.25	0.25
False Acceptance Cost (A)	100.00	100.00	20.00
False Rejection Cost (R)	20.00	100.00	100.00

Economic Results

The Tables of Appendix C give summaries of results for the economic optimization runs for all three cases of n_{max} (integer valued from one to three). Included in these appendix tables are the various optimal cutoff values for each set of parameters. The data shown in Appendix C is utilized in the graphs which are discussed in this section.

For a given set of sampling costs and observation maximum (n_{max}), the expected total cost of sampling decreases as the prior mean moves away from the specification limit. That is, expected costs are expectedly (relatively) high when the mean of the prior distribution falls close to the specification. In the worst case, when the prior mean coincides with the specification limit (and half of the batches may be assumed non-conforming), the probabilities of false disposition are high and drive up the expected total cost. As the prior mean moves away from the specification, the chance of false disposition lessens, reducing the expected total cost of the plan.

This influence of the prior distribution on the expected total cost is evident in Figures 7.1 through 7.8. The expected total cost is shown in each figure as well as the individual expected cost components (false disposition and sampling) which comprise the totals. Note that this pattern is consistent among cost sets (1-3), with the changes in costs reflected only in the component makeup of the expected total cost.

Predictably, for cost set #1, the expected costs of false acceptance dominate the total. When the acceptance and rejection costs are reversed in cost set #3, the results symmetrically mirror those of the first cost set, with the expected costs of false rejection dominating the total cost. The optimal cutoff values of cost sets 1 and 3 are actually symmetrical around the specification, leading to this symmetry of results. In the second cost set, with equal costs of false rejection and acceptance, the cost components are symmetrical around the specification limit, with the false rejection costs dominating for prior mean values greater than the specification and false acceptance costs having the greatest influence for lesser values.

Due to the cost set symmetry (between #1 and #3) evident in the plans for n_{max} values of 1 and 2, the third cost set for n_{max}=3 is not presented in the Appendix or in chart form.

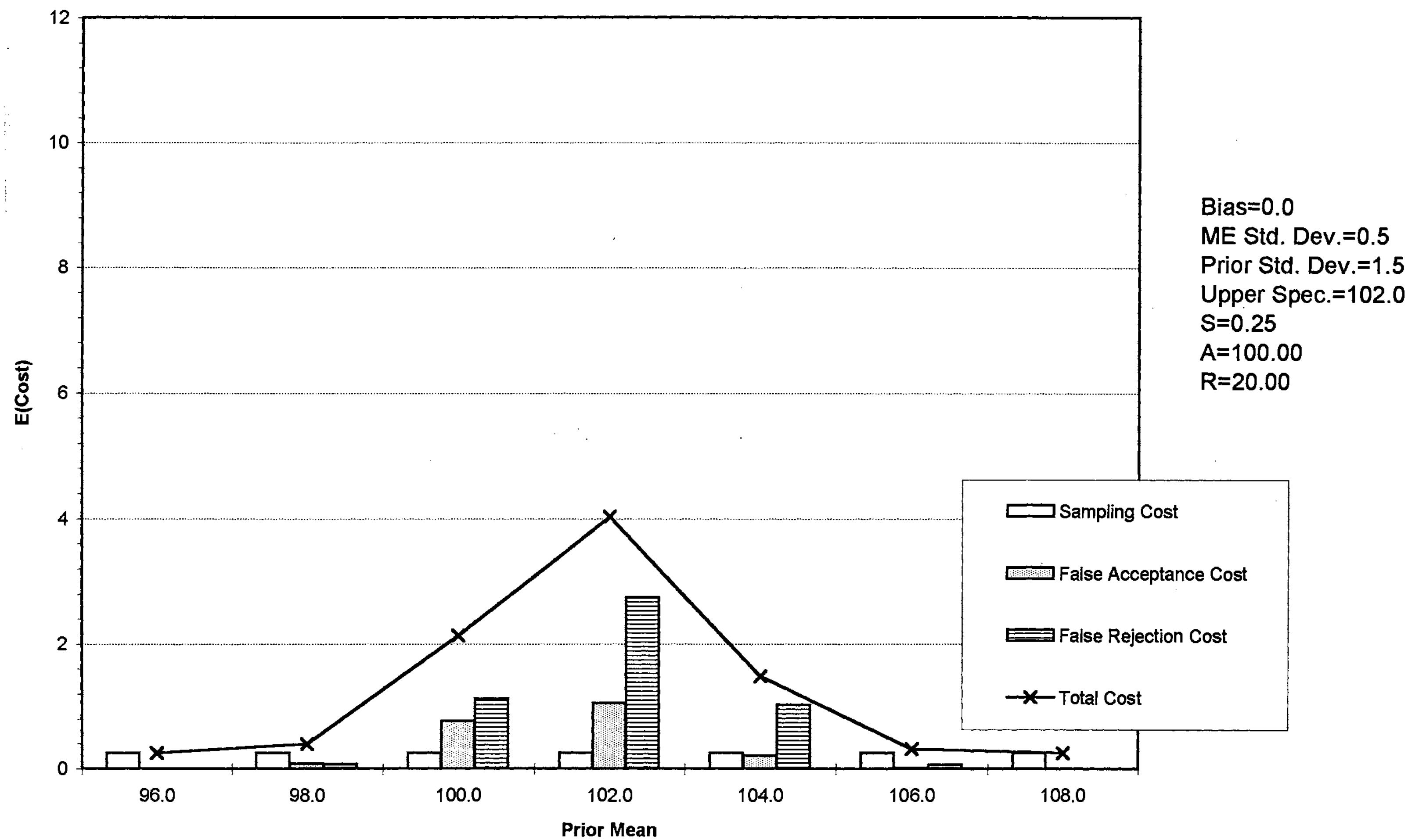

Figure 7.1 Economic Case: E(Cost) vs. Prior Mean (n_{max}=1, Cost Set #1)

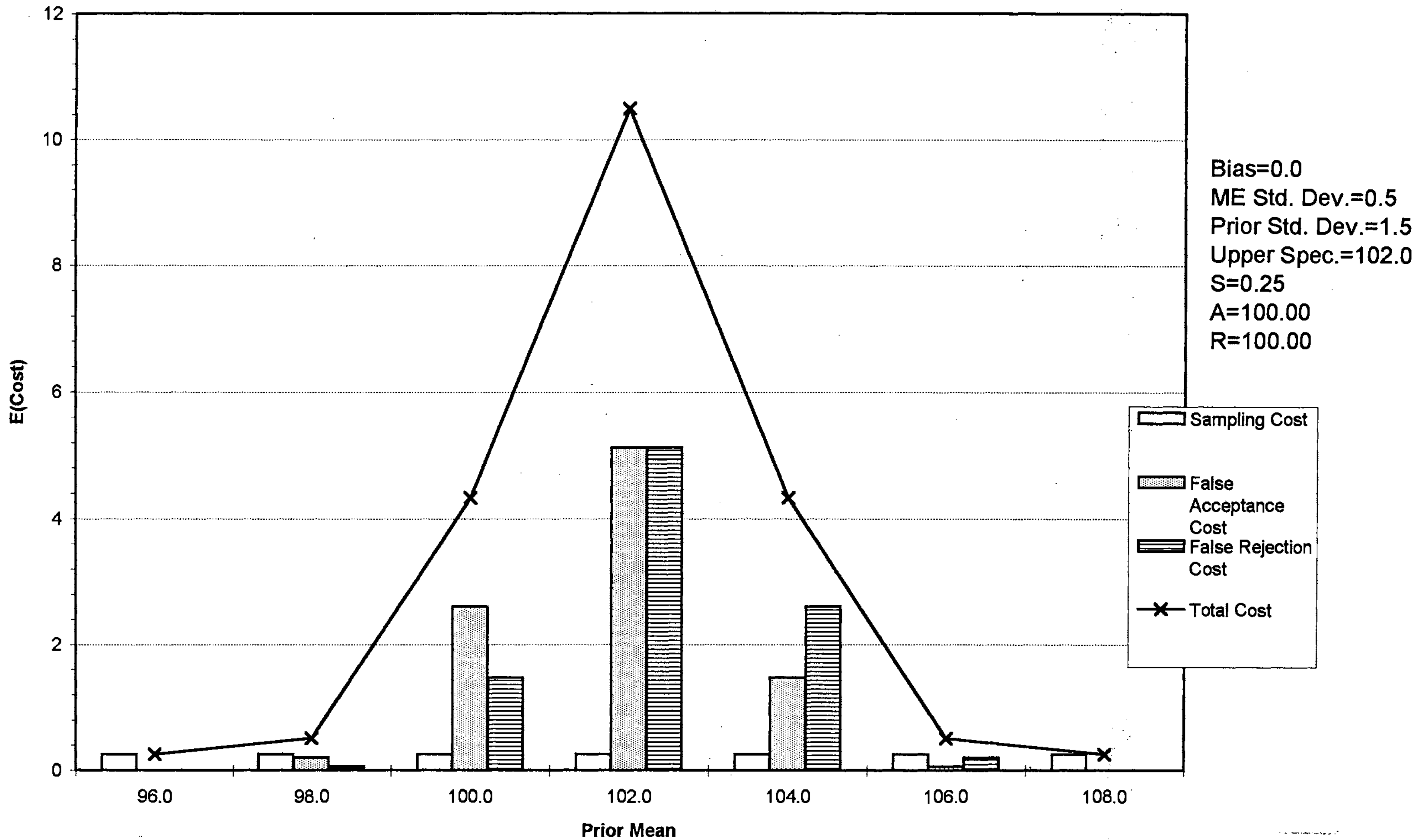

Figure 7.2 Economic Case: E(Cost) vs. Prior Mean (n_{max}=1, Cost Set #2)

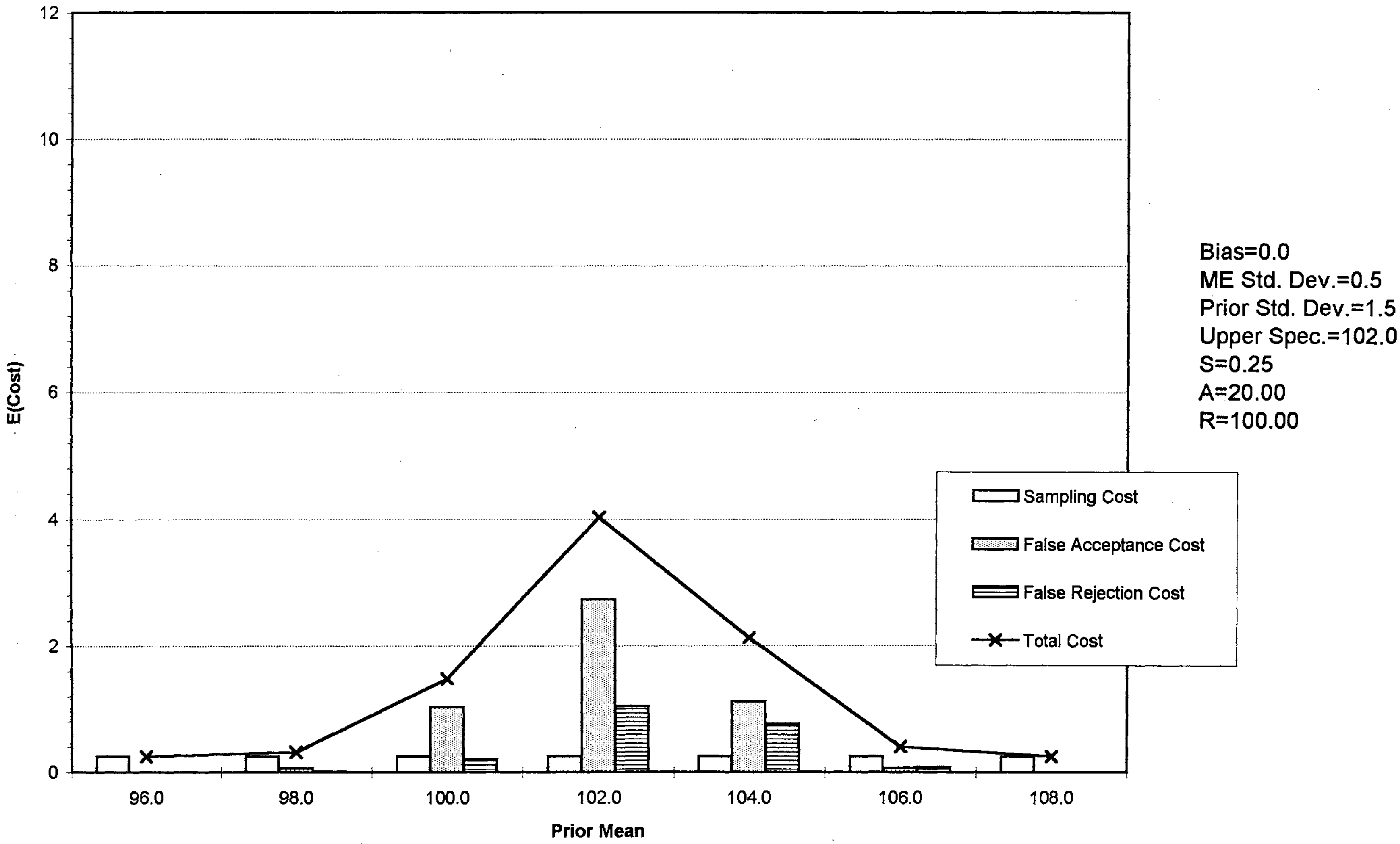

Figure 7.3 Economic Case: E(Cost) vs. Prior Mean (n_{max}=1, Cost Set #3)

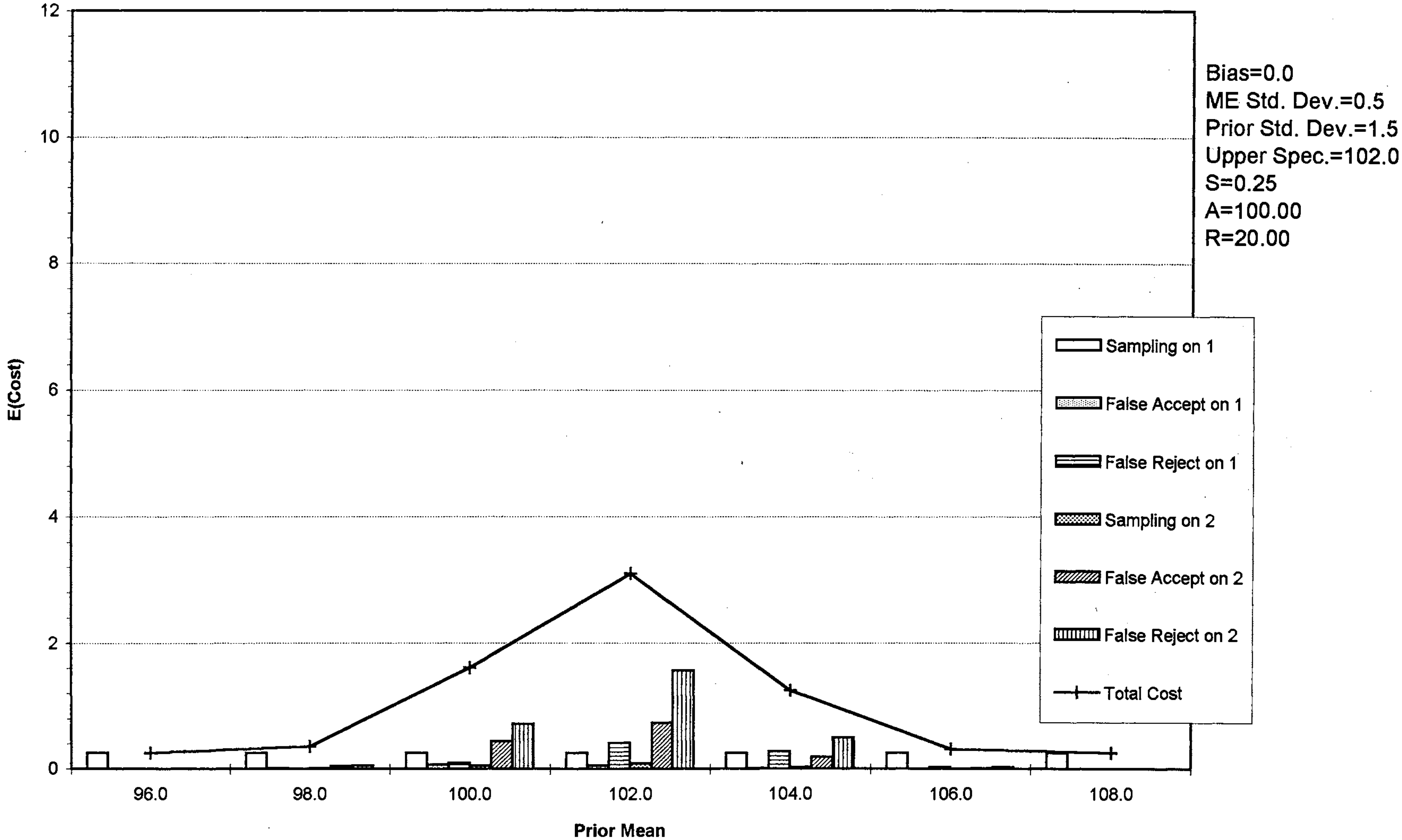

Figure 7.4 Economic Case: E(Cost) vs. Prior Mean (n_{max}=2, Cost Set #1)

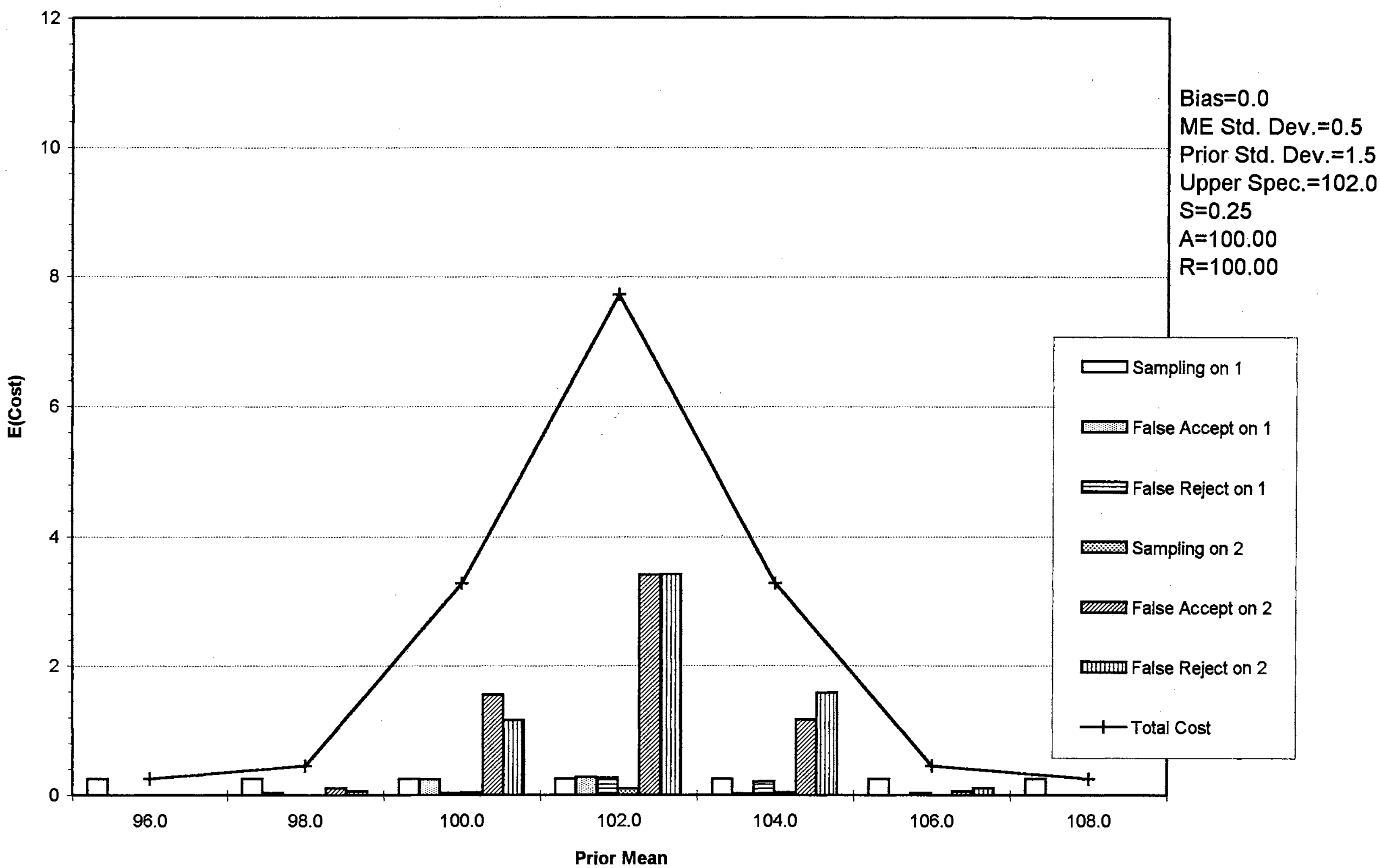

Figure 7.5 Economic Case: E(Cost) vs Prior Mean (n_{max}=2, Cost Set #2)

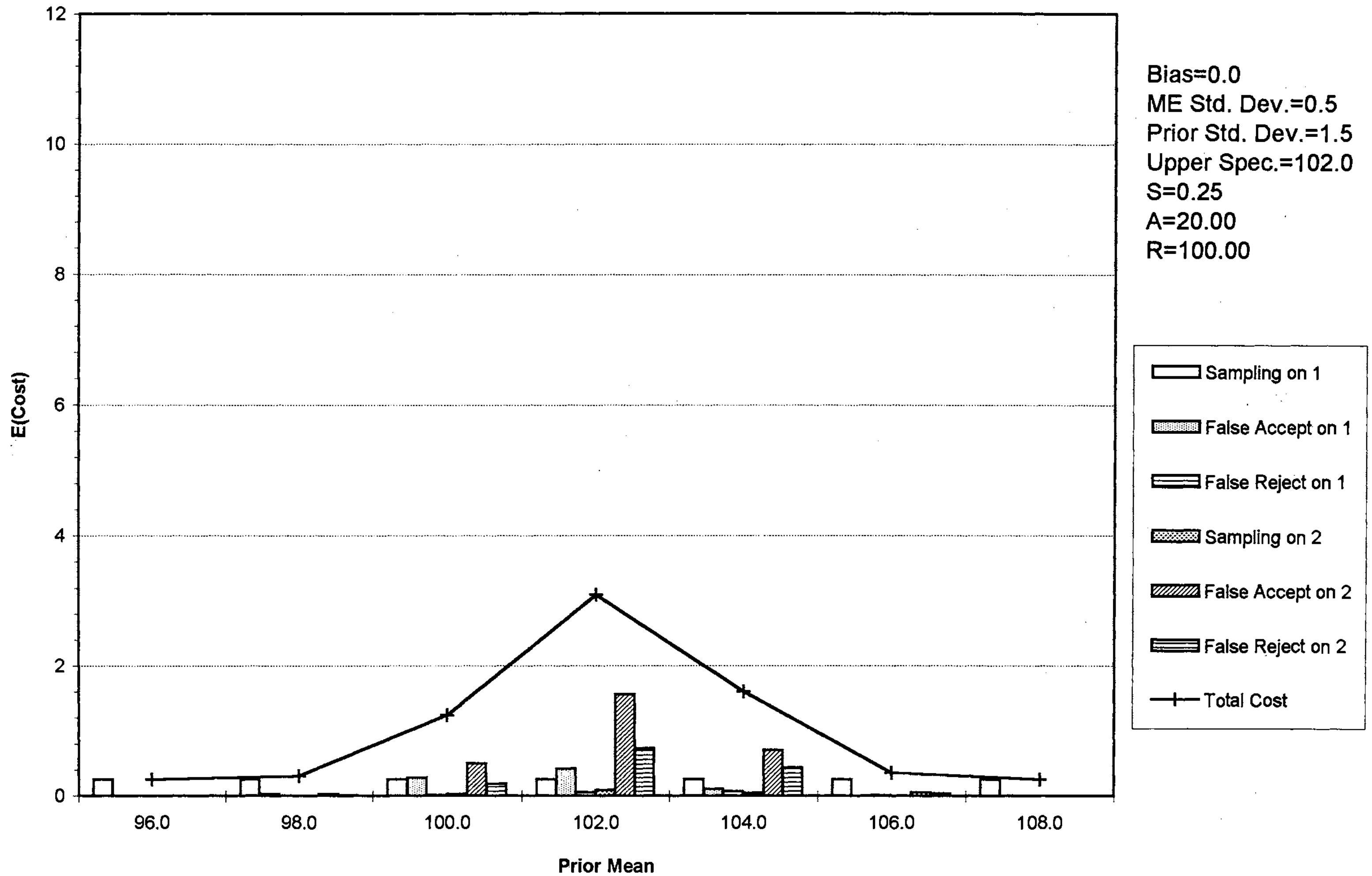

Figure 7.6 Economic Case: E(Cost) vs. Prior Mean (n_{max}=2, Cost Set #3)

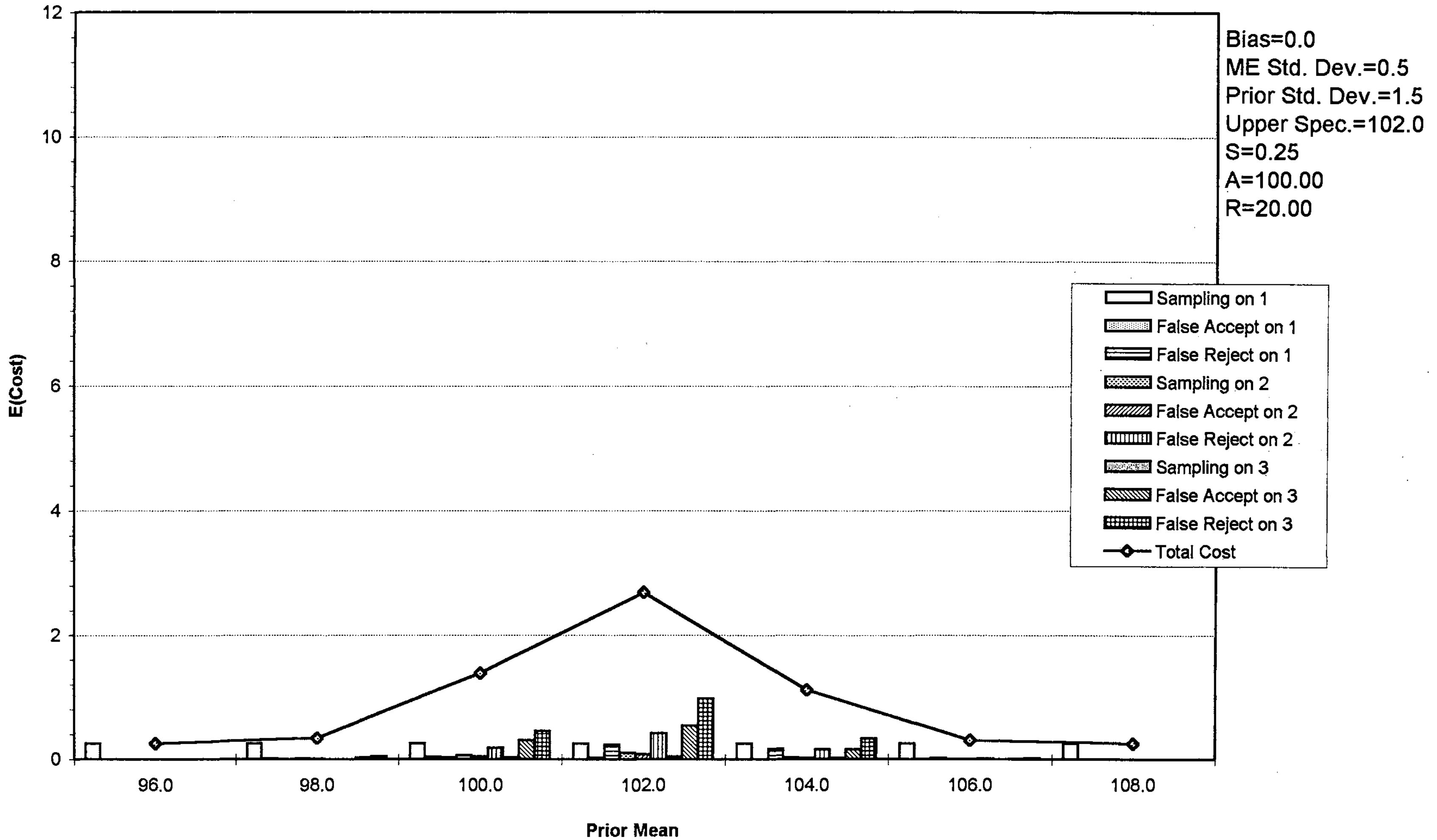

Figure 7.7 Economic Case: E(Cost) vs. Prior Mean (n_{max}=3, Cost Set #1)

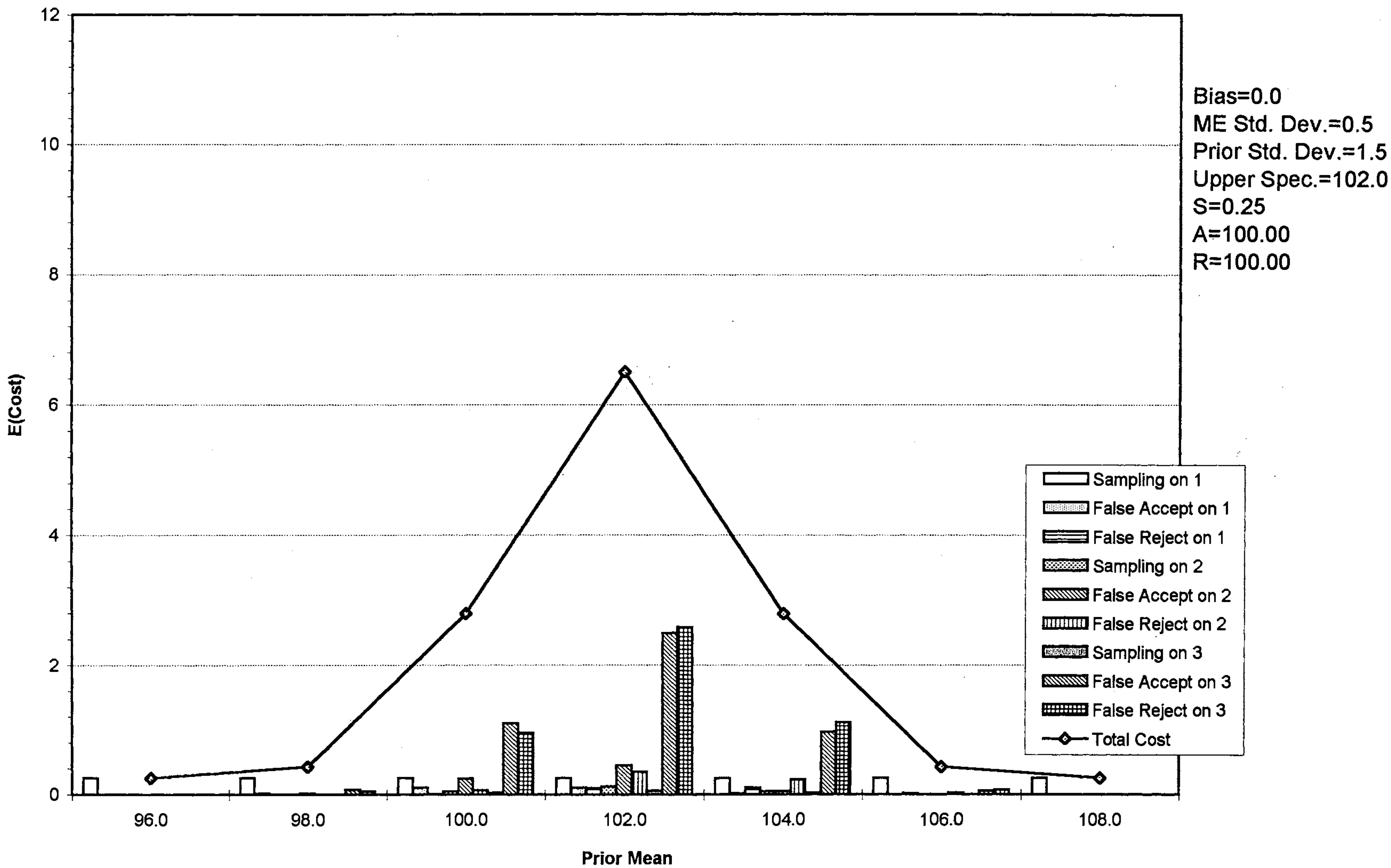

Figure 7.8 Economic Case: E(Cost) vs. Prior Mean (n_{max}=3, Cost Set #2)

Figures 7.9 through 7.11 show the influence of increasing the acceptable maximum number of observations for a given cost set and prior distribution mean. In the case of the example data which carries a relatively low iteration cost (S), an increase in n_{max} lowers the expected total cost for the sampling plan in each case examined. This trend is evident for each of the three cost sets (sampling cost constant). In a complicated inspection procedure in which the iteration cost is dominant and overshadows the false disposition costs (due to gage reset, cleaning, calibration, etc.), an optimal plan could indicate that it is not cost effective to increase the number of observations. For example, if the optimal plan for $n_{max}=2$ gives equal cutoff values for $C_{1,L}$ and $C_{1,H}$, this indicates that a decision is made on the first iteration due to the dominant sampling (measurement) costs. Logically, this cutoff value is also the C_1 value obtained for the optimal $n_{max}=1$ plan.

These data figures also illustrate the effect which the prior mean has on expected total cost of the plan. The highest expected cost (over all n_{max})occurs at prior mean values equal and close to the upper specification. As the prior mean moves away from the specification, the expected cost decreases for any n_{max}.

Note the overlap which occurs in Figure 7.10 for equal costs of erroneous acceptance and rejection (cost set #2). Prior means which are equidistant from the upper specification yield identical expected total costs. For example, the data lines for prior means of 96.0 (-6.0 from the specification) and 108.0 (+6.0) are overlaid in this chart.

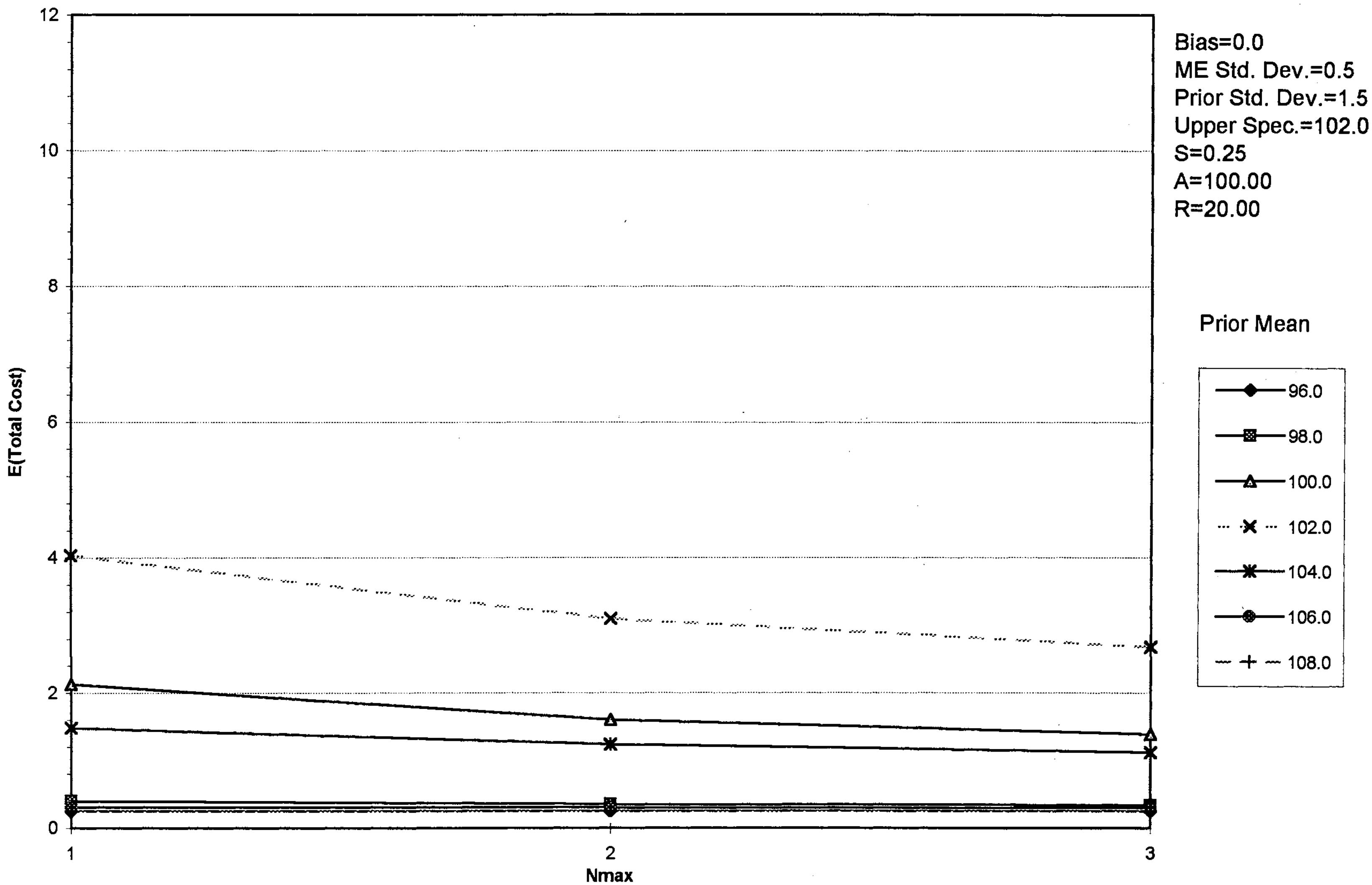

Figure 7.9 Economic Case: E(Total Cost) vs. n_{max} for Various Prior Means (Cost Set #1)

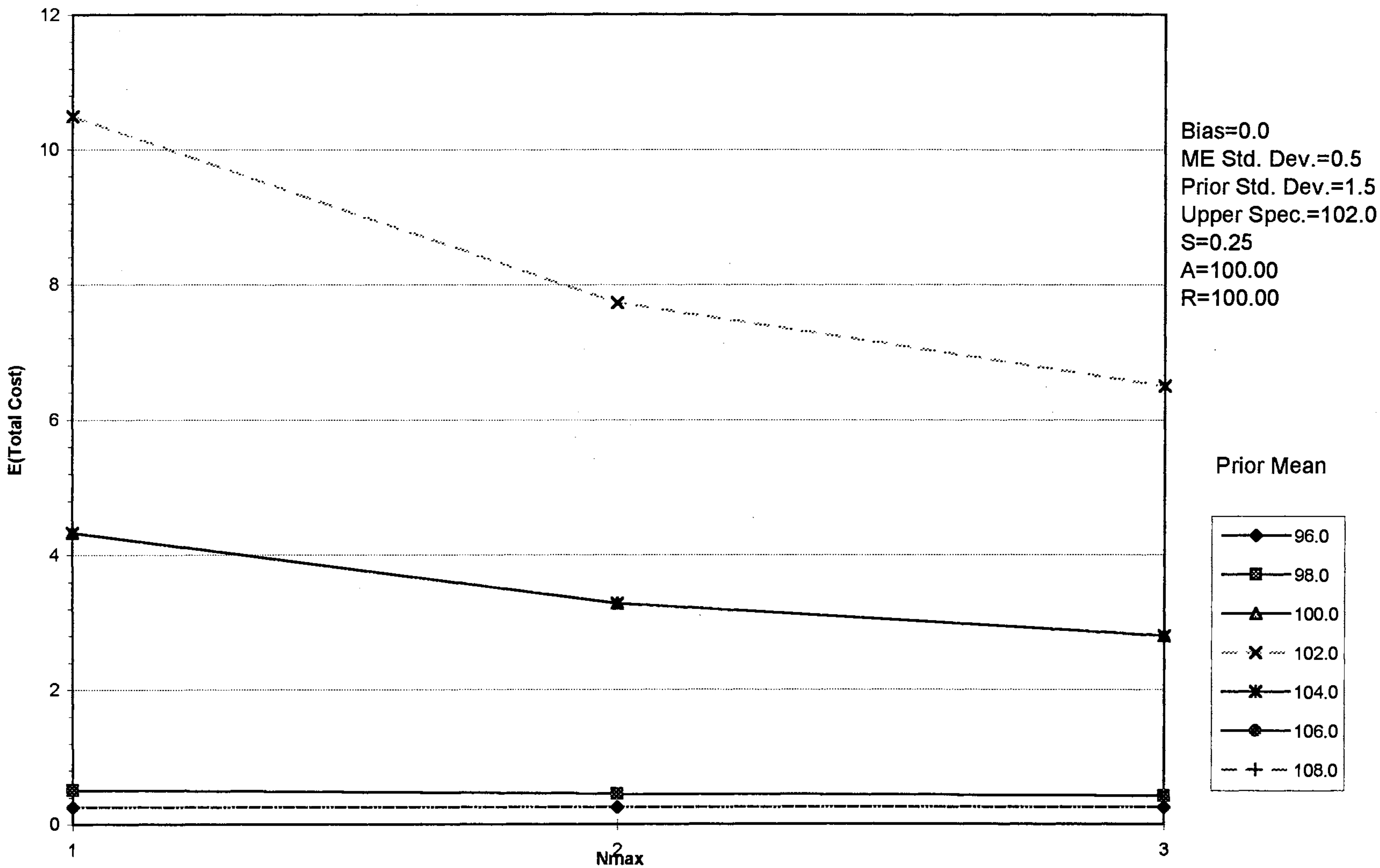

Figure 7.10 Economic Case: E(Total Cost) vs. n_{max} for Various Prior Means (Cost Set #2)

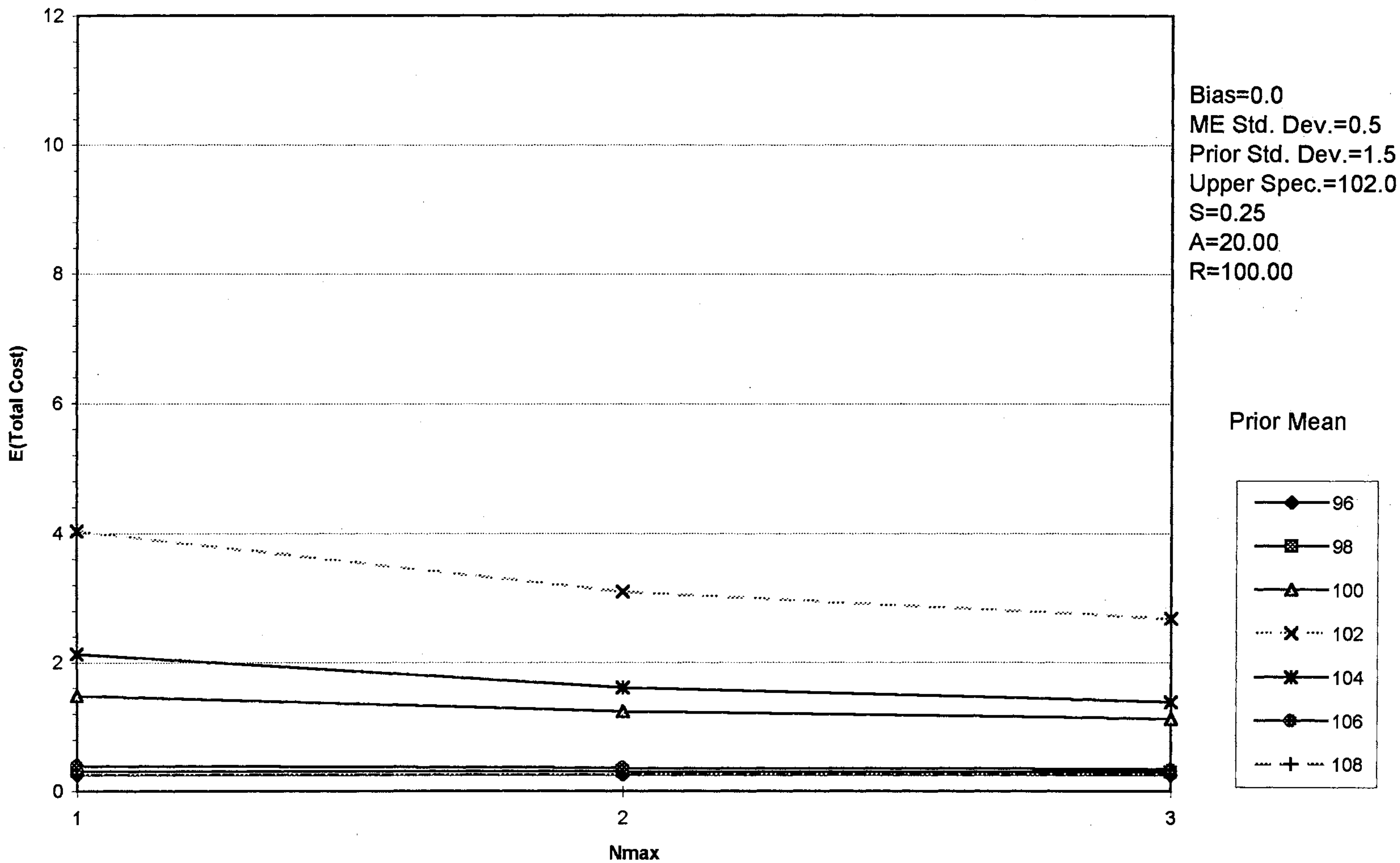

Figure 7.11 Economic Case: E(Total Cost) vs. n_{max} for Various Prior Means (Cost Set #3)

7.3 COMPARISON OF STATISTICAL AND ECONOMIC MODELS

Comparison Methodology

Unlike the economic results which are developed theoretically, results presented for the statistical case are generated through simulation. Using the SPRT theory discussed in Chapter three, FORTRAN simulators (given in Appendix D) are utilized to implement the batch disposition logic for each value of n_{max} (1-3). In order to facilitate a comparison with the economic sampling approach, a cap is placed on the maximum observation number for the statistical case. Recall that a truncated SPRT effectively limits the number of iterations and provides a vehicle for consistent disposition within the desired number of observations.

In order to create a basis for comparison among various statistical scenarios and also between the statistical and economic approaches, the simulator uses the same computer-generated random numbers (which yield identical simulated batch observations) in assessing the batch by both the statistical and economic sampling approaches. That is, the exact same observation values, in the same sequence, are treated using each of the sequential sampling plans and the optimal economic plans. This facilitates meaningful comparisons between statistical cases for various sampling parameters, and also between the statistical and economic models. The statistical approach utilizes the truncated SPRT as presented in Chapter 3, while the economic approach examines the batch using the optimal sampling plan as generated by the computer program presented in Appendix A and discussed in Chapter 5.

Although separate simulators are used for each of the three n_{max} values, the logic behind the batch assessment is the same. Figure 7.12 depicts, in flowchart form, the simulation logic utilized in the FORTRAN program for any particular n_{max}. The logic shown is for a single trial of the simulator (examination of a single batch).

For each set of sampling parameters examined, 50,000 batch trials are conducted using the simulator. In each case, an average cost, probability of false acceptance and probability of false rejection are computed over the total number of trials. Recall that the false disposition probabilities calculated for the statistical sampling method are not directly comparable to the input values of alpha and beta (which relate to the indifference limits, not the specification). The economic results which are presented from the simulation runs are comparable to the theoretical values presented in the previous section. Additional information is provided as program output on the nature of the decisions made (correct or incorrect) and the outcome of the Wald SPRT truncation rule when n_{max} is encountered.

Given this simulation method of sampling plan assessment, results are presented for several cases of input parameters. Statistical results are examined for various combinations of alpha and beta. Additionally, the statistical indifference limits (necessary for SPRT usage) are varied in order to determine their relation to average plan costs. A

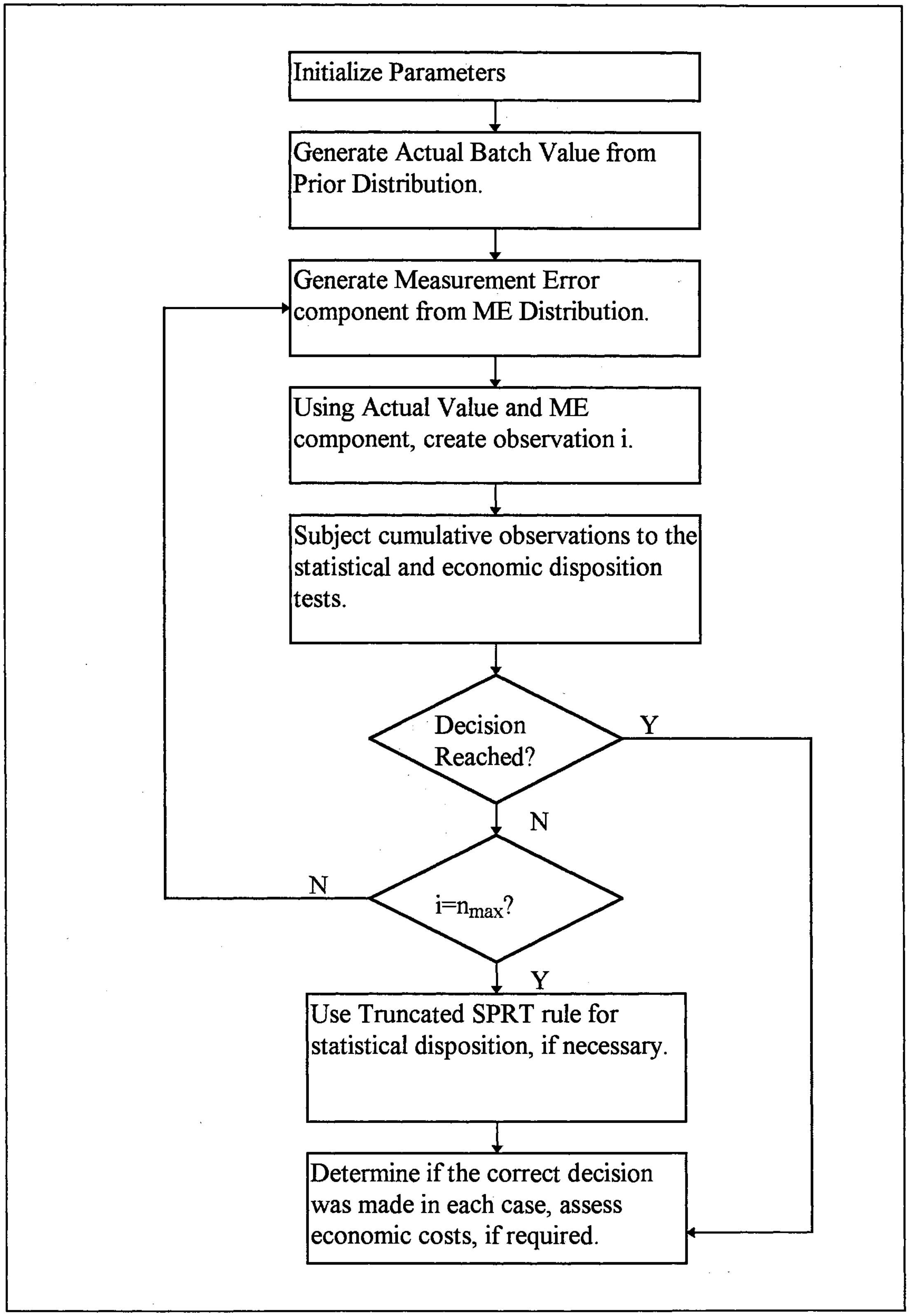

Figure 7.12 Simulation Logic for Sequential and Economic Batch Assessment.

single cost set (#1, the most realistic, as shown in Table 7.1) is examined in comparing the two decision systems. The values of α, β, and indifference limits which are selected for statistical analysis represent reasonable user estimates, assuming that the economic costs are real, yet explicitly unknown to the statistical plan user. The experimental parameter values used in the simulation runs are presented in Table 7.2, below.

Table 7.2

Statistical Parameters Used in Simulation Runs.

α, β
0.05, 0.05
0.05, 0.01
0.1, 0.01

Lower Indifference Limit, Upper Indifference Limit
101.0, 102.0
101.5, 102.0
101.0, 102.5
101.5, 102.5
101.0, 103.0
101.5, 103.0
102.0, 102.5
102.0, 103.0

All combinations of the above parameter sets are tested through simulation. Note that the Indifference limit combination of (102.0, 102.0) (the upper specification limit) is not explored (or even, allowed) using the statistical sampling procedure.

The test program simulates actual batch values from the prior distribution specified for the optimum economic plan. Although this distribution is actually unknown to the statistical model user at the time of plan design, it, as well as the economic costs of sampling are used following simulated batch disposition to assess the plan performance relative to the optimal economic plan. A single prior distribution mean is used in all experimental runs. The value chosen is equal to the upper specification limit (102.0) and represents the scenario which is the most difficult to correctly judge and is, predictably, the most costly to conduct. Other parameters of the sampling system are as given in the previous section, and below:

Upper Specification = 102.0
Prior Mean = 102.0
Prior Standard Deviation = 1.5
Measurement Error Standard Deviation = 0.5
Measurement Error Mean (Bias) = 0.0

These parameters are used consistently throughout the simulated sampling runs.

Statistical vs. Economic Model Results

Figures 7.13 through 7.21 show the results of the simulation runs for all cases of maximum allowable iterations (n_{max}=1-3). For each particular set of parameters, the statistical expected total cost and probabilities of false disposition are give as functions of the selected (α, β) and indifference pairs. For comparison, the simulation results of conducting economic sequential sampling using the optimum cutoff values are included on these graphs. In all cases, the cost performance of the economic plan surpasses statistical plan results.

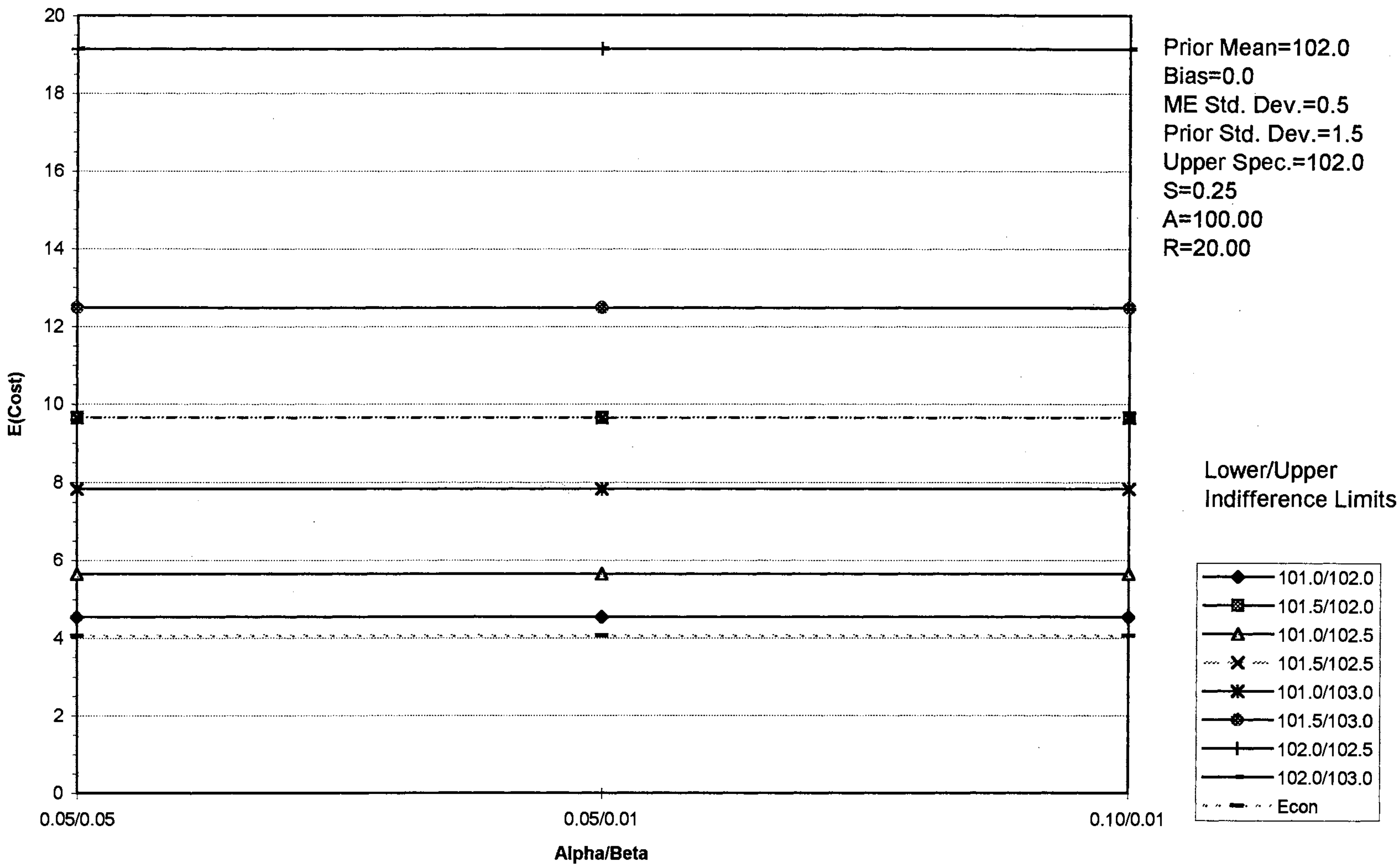

Figure 7.13 Economic & Statistical Cases: E(Cost) vs. α,β for Various Indifference Limits (n_{max}=1, Cost Set #1)

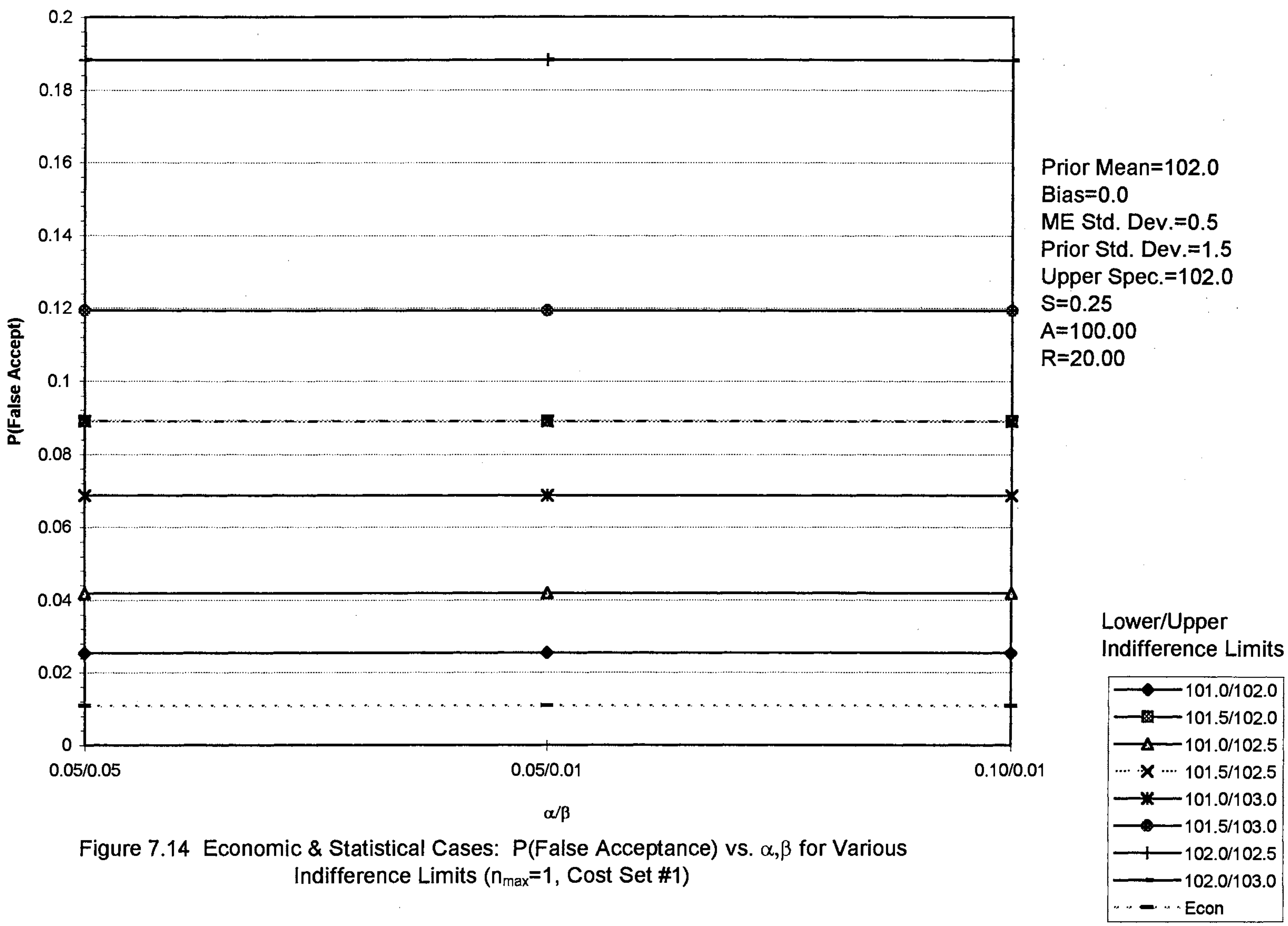

Figure 7.14 Economic & Statistical Cases: P(False Acceptance) vs. α,β for Various Indifference Limits (n_{max}=1, Cost Set #1)

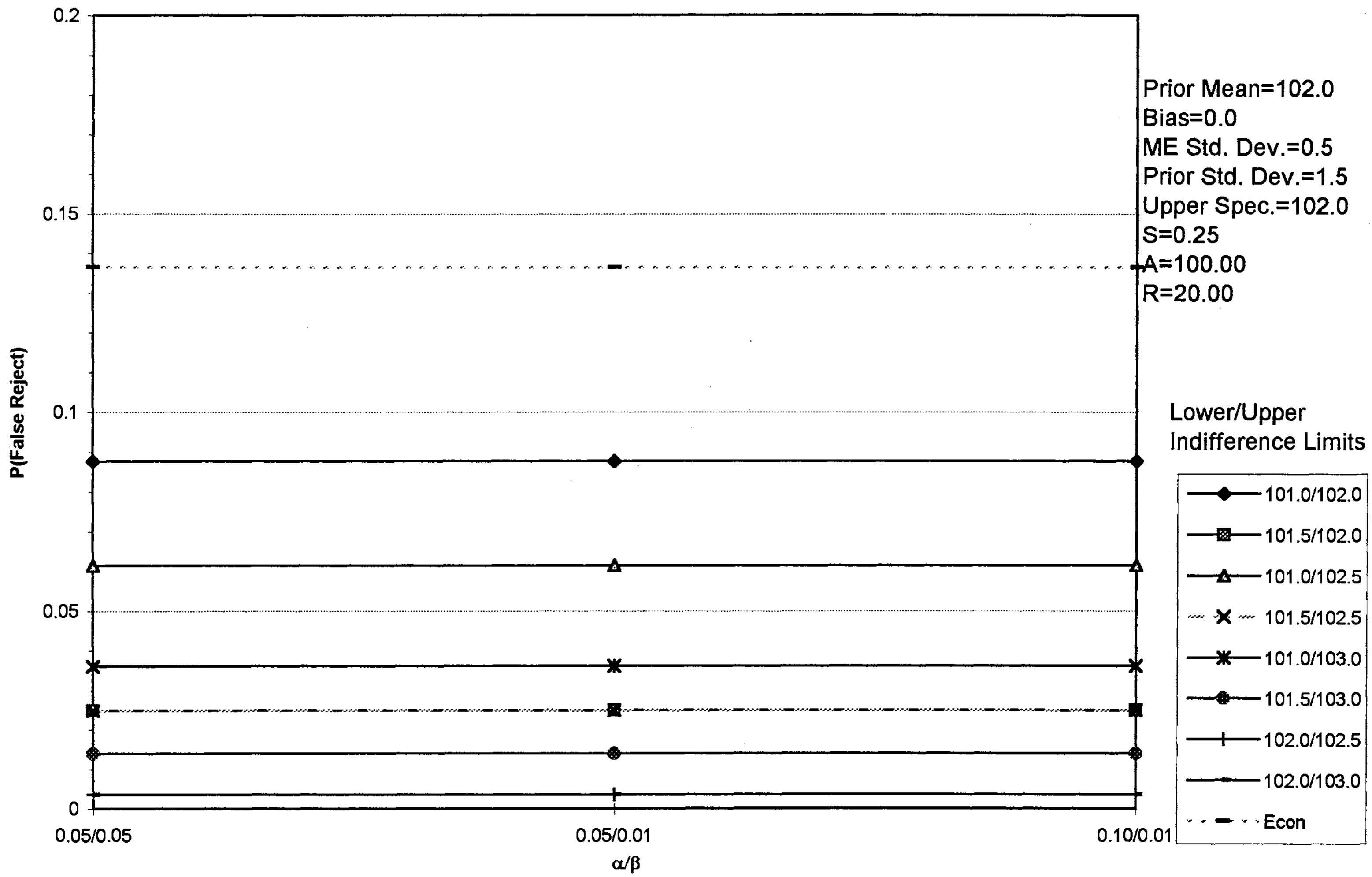

Figure 7.15 Economic & Statistical Cases: P(False Rejection) vs. α,β for Various Indifference Limits (n_{max}=1, Cost Set #1)

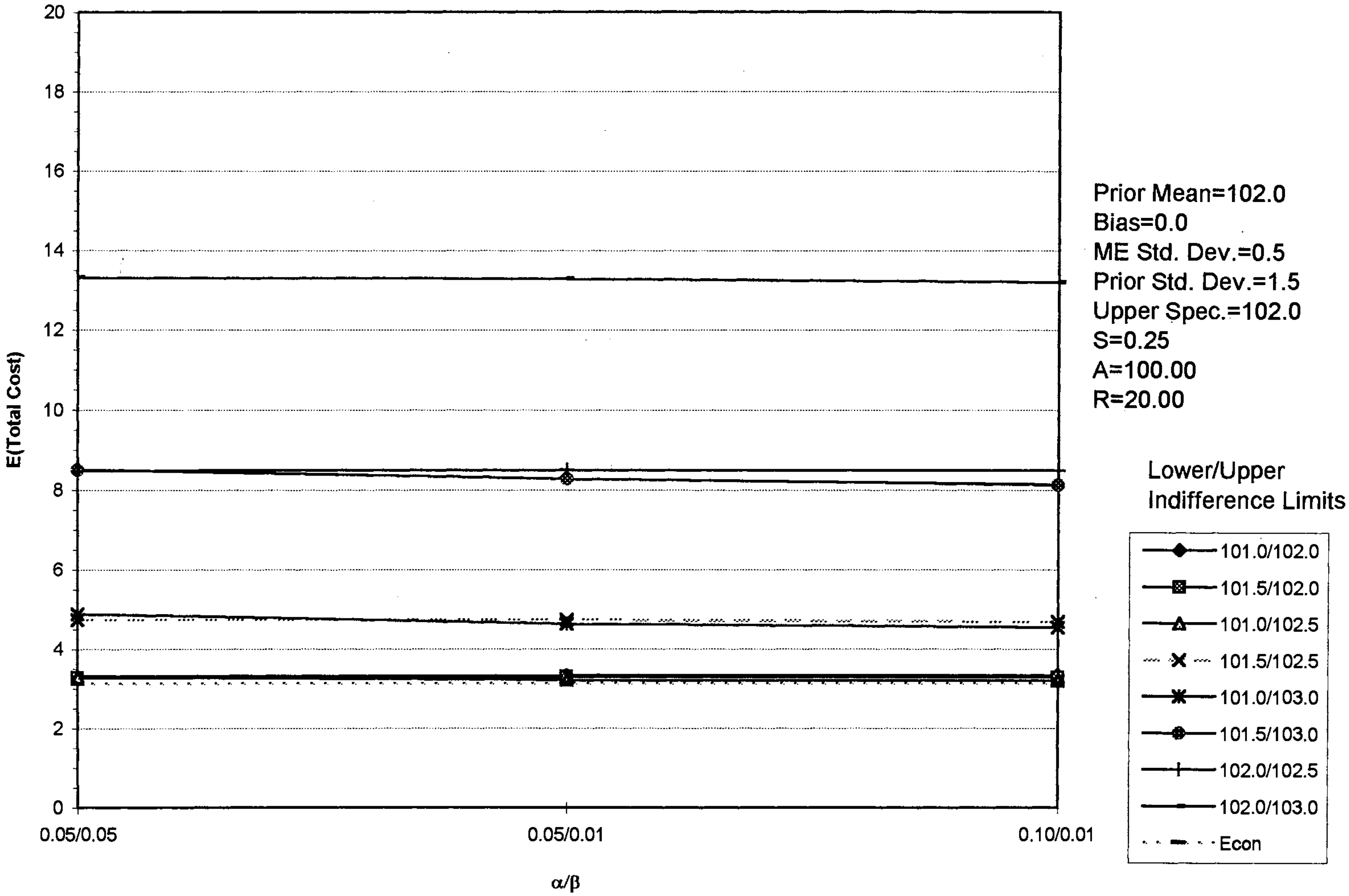

Figure 7.16 Economic & Statistical Cases: E(Total Cost) vs. α,β for Various Indifference Limits (n_{max}=2, Cost Set #1)

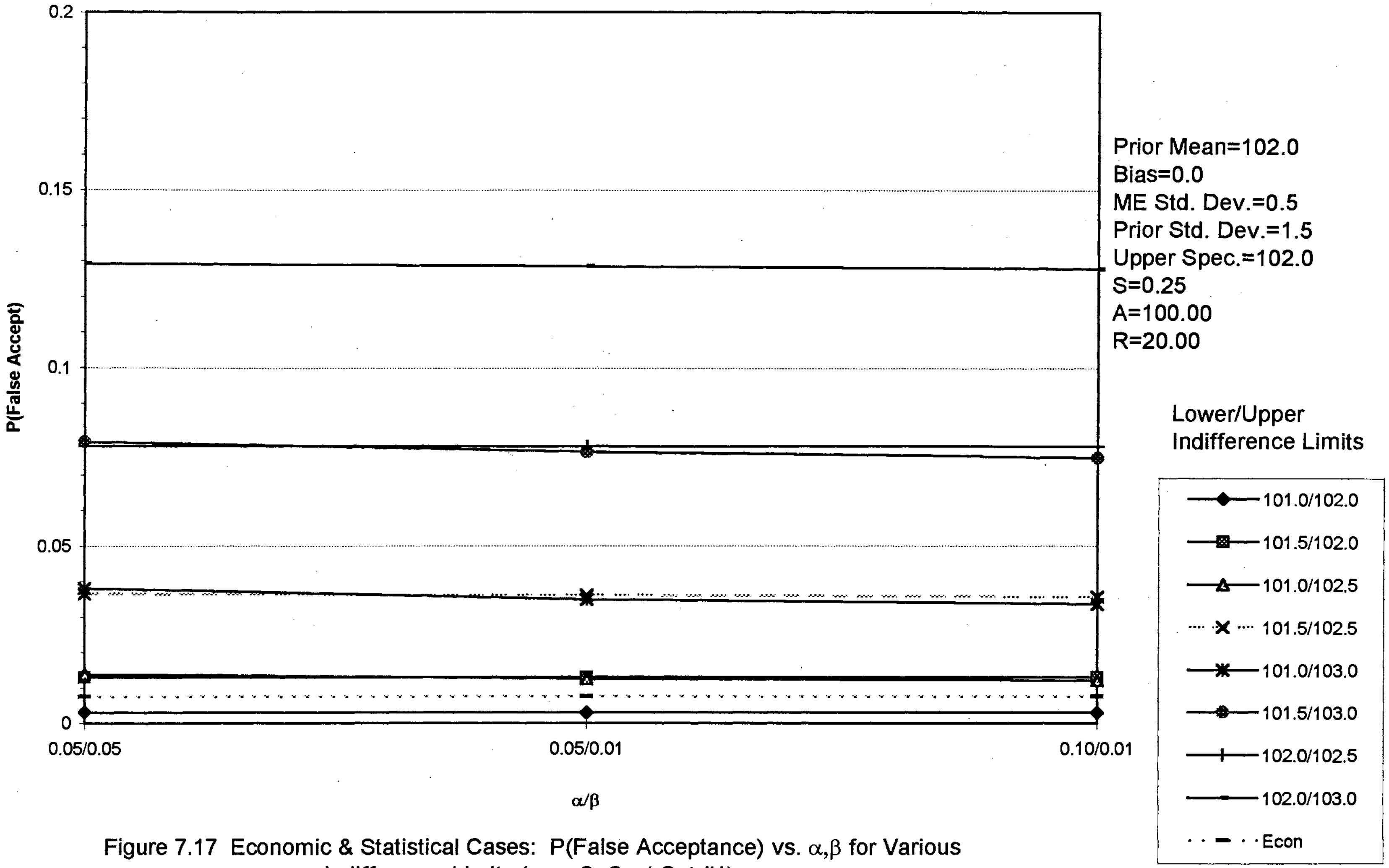

Figure 7.17 Economic & Statistical Cases: P(False Acceptance) vs. α,β for Various Indifference Limits (n_{max}=2, Cost Set #1)

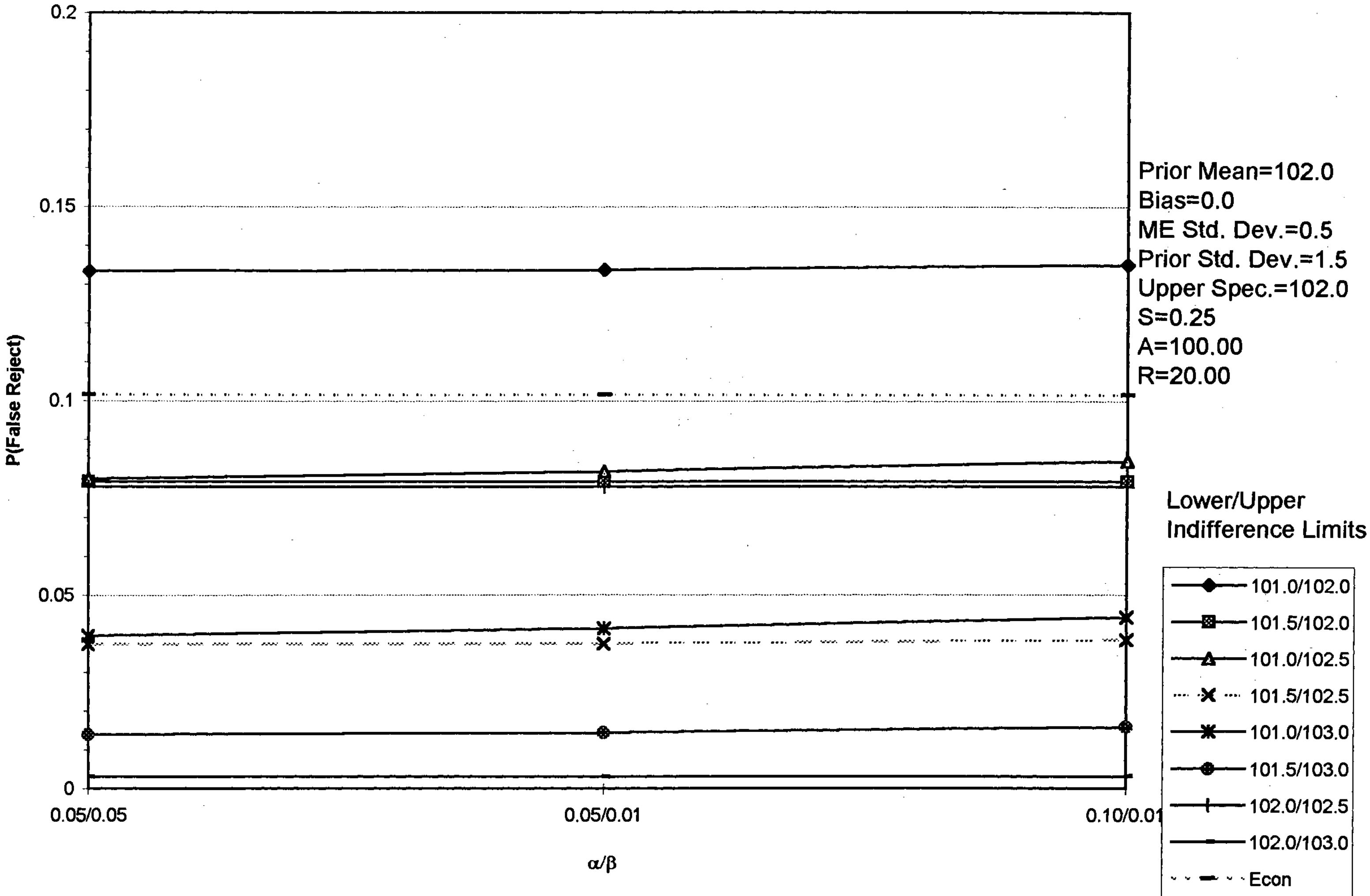

Figure 7.18 Economic & Statistical Cases: P(False Rejection) vs. α,β for Various Indifference Limits (n_{max}=2, Cost Set #1)

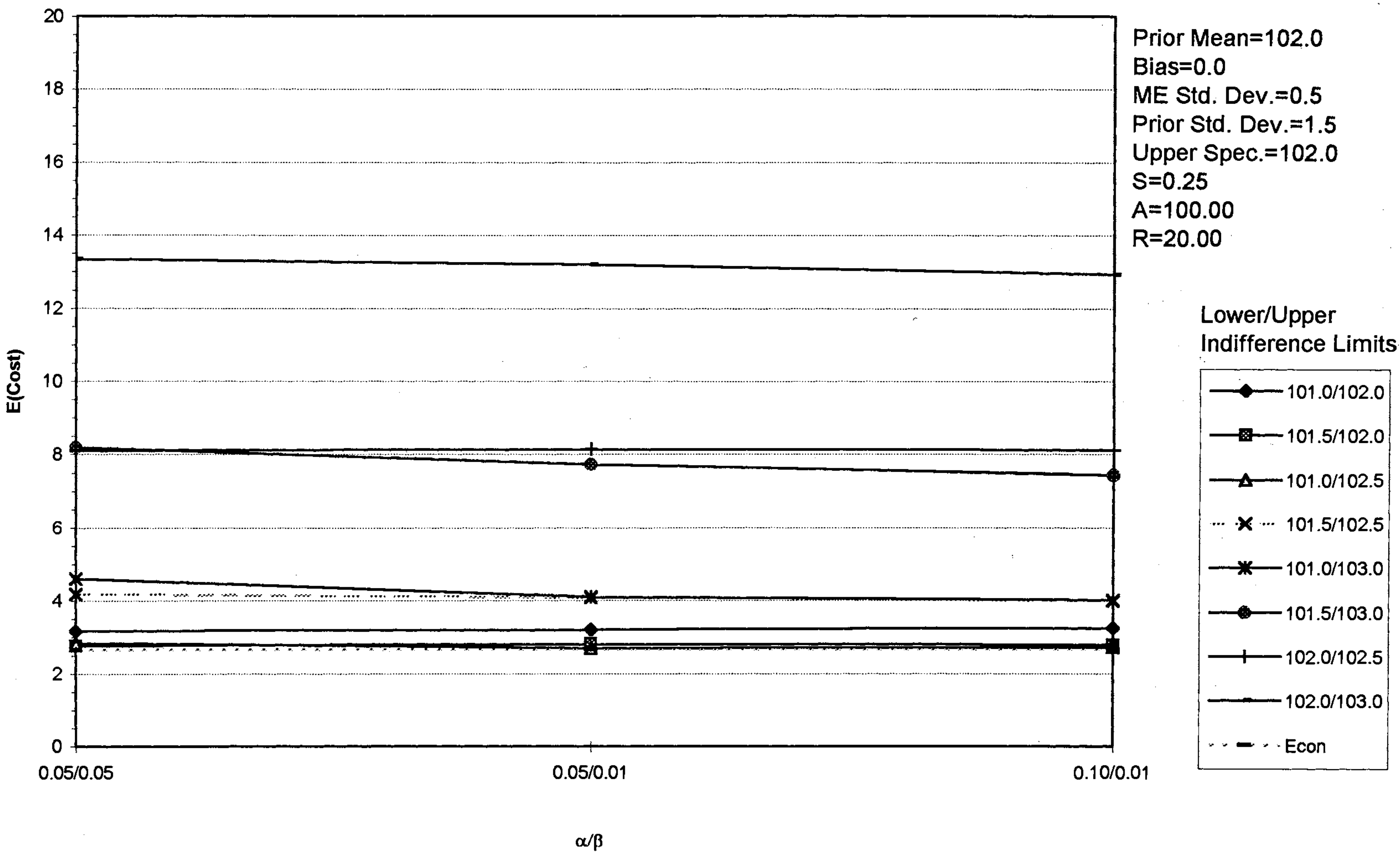

Figure 7.19 Economic & Statistical Cases: E(Cost) vs. α,β for Various Indifference Limits (n_{max}=3, Cost Set #1)

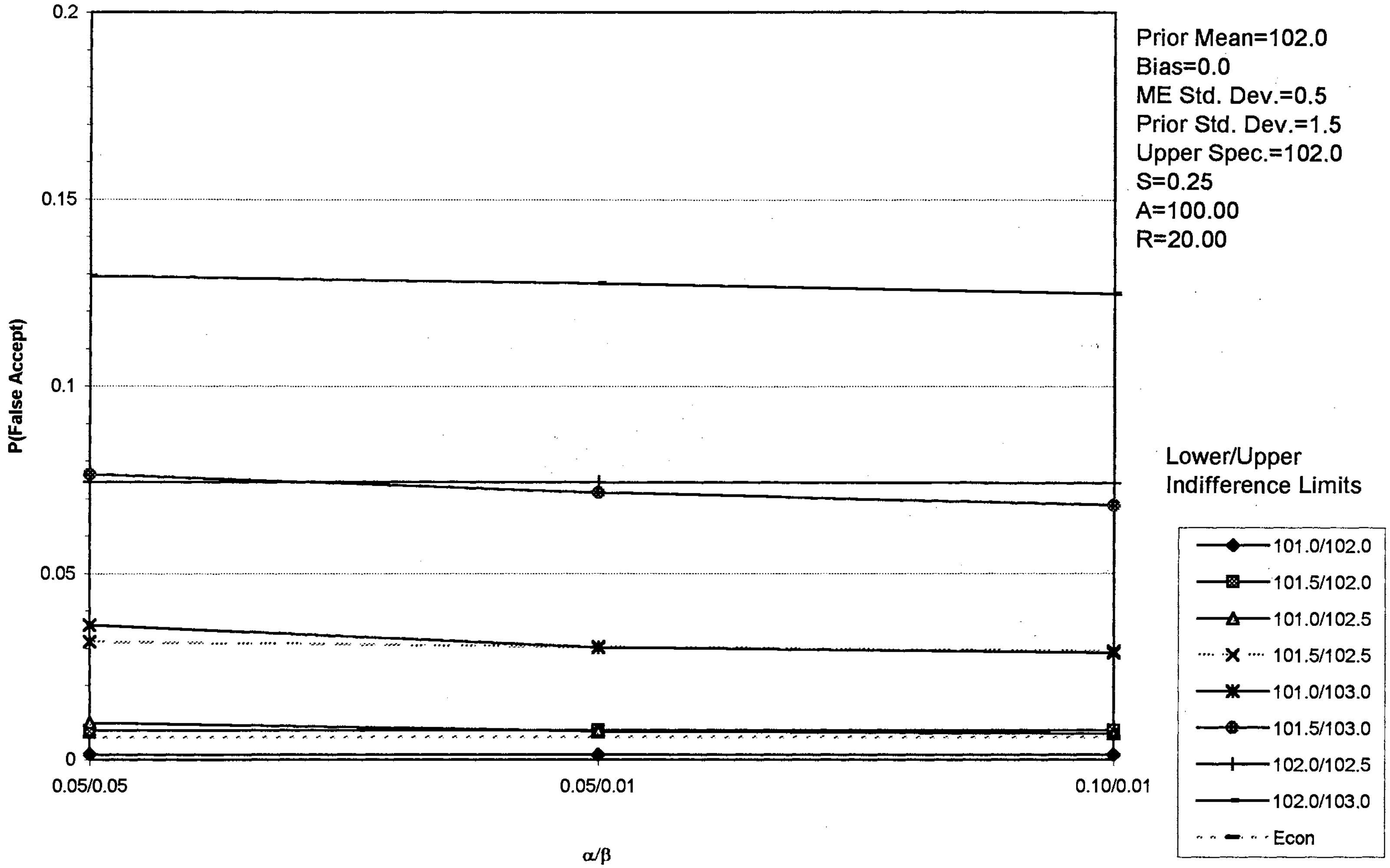

Figure 7.20 Economic & Statistical Cases: P(False Acceptance) vs. α,β for Various Indifference Limits (n_{max}=3, Cost Set #1)

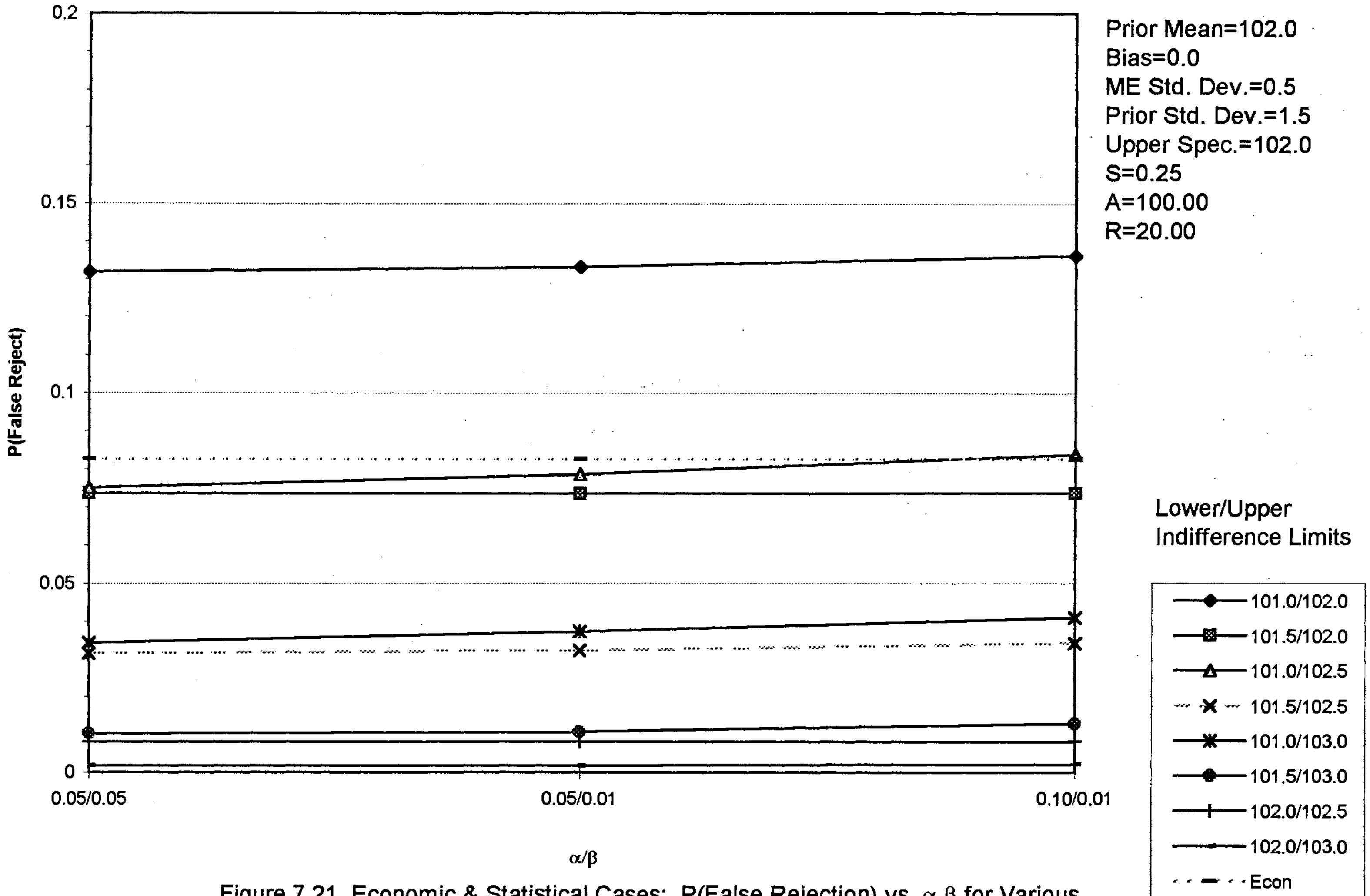

Figure 7.21 Economic & Statistical Cases: P(False Rejection) vs. α,β for Various Indifference Limits (n_{max}=3, Cost Set #1)

Note that, in the case of $n_{max}=1$, costs and probabilities are unchanging for all (α, β) pairs within any given indifference limits. This, indeed, is the result of identical statistical disposition simulation results from set to set over all 50,000 simulation trials. This is not so hard to swallow when recalling that a decision is required based upon a single iteration (or subsequent application of the truncated SPRT disposition rule).

Figures 7.22 through 7.29 provide graphical comparison of the economic and statistical expected total cost results across the three possible values of n_{max} within a chosen set of statistical indifference limits. Again, the unchanging statistical case results for $n_{max}=1$ are evident.

For n_{max} values greater than 1, the effects of changing values of α and β vary among sets of indifference limits for the statistical case. That is, the low cost (α,β) pair depends on the indifference limit pair under consideration and is not universally optimal. In all cases, the optimal economic cases outperform the statistical disposition combinations.

Some discussion must be dedicated to the choices made of risk parameters for the statistical runs. The various risk parameters are subjectively selected with consideration given to the explicit costs used in the optimal economic scenario. It may be assumed that the statistical plan user, although unaware of the exact costs of false batch disposition, has a "feel" for the relative magnitude of the consequences of mistakes and estimates α and β based on this knowledge.

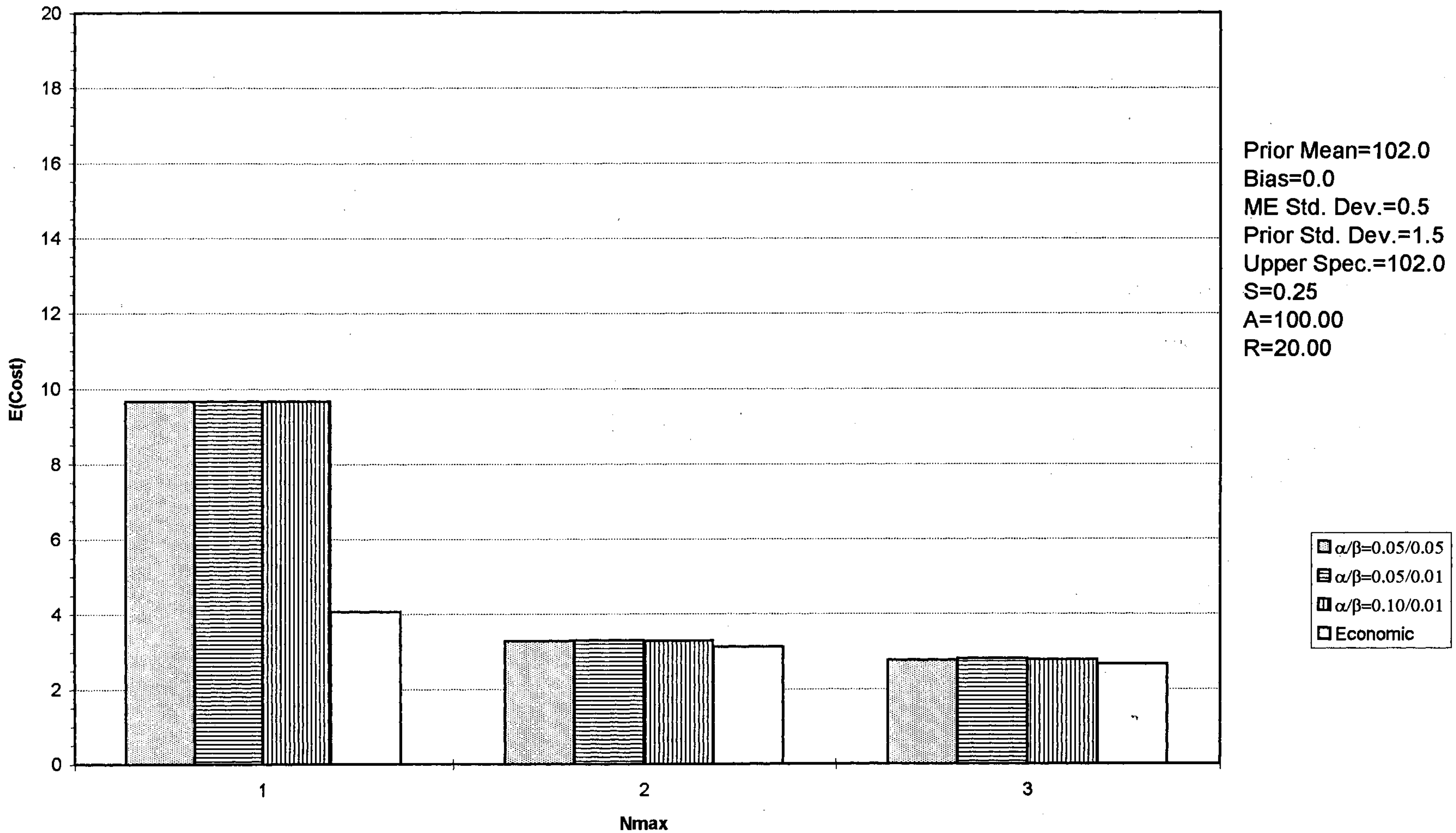

Figure 7.22 Economic & Statistical Cases: E(Cost) vs. n_{max} for Given Indifference Limits (101.5, 102.0)

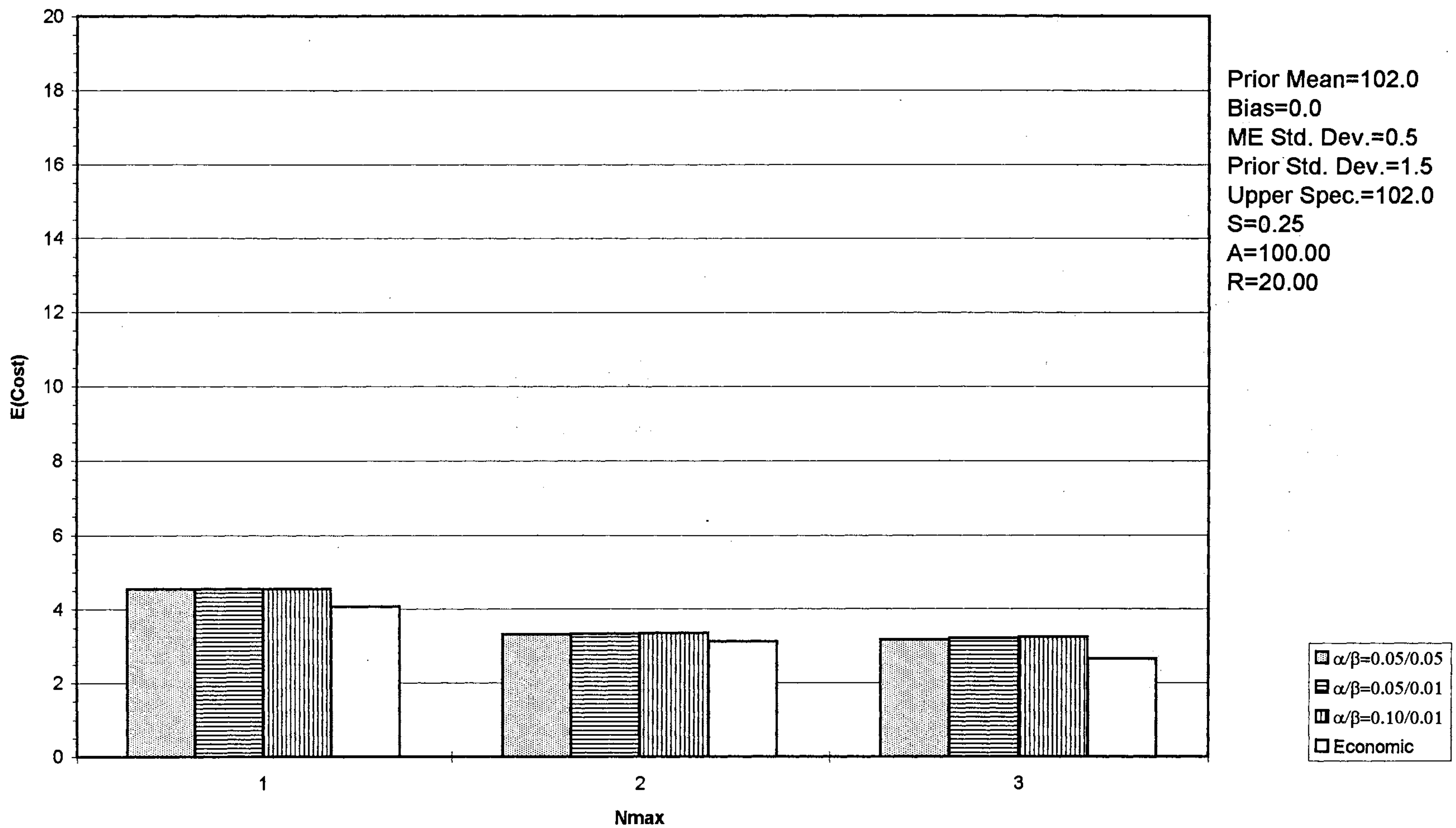

Figure 7.23 Economic & Statistical Cases: E(Cost) vs. n_{max} for Given Indifference Limits (101.0, 102.0)

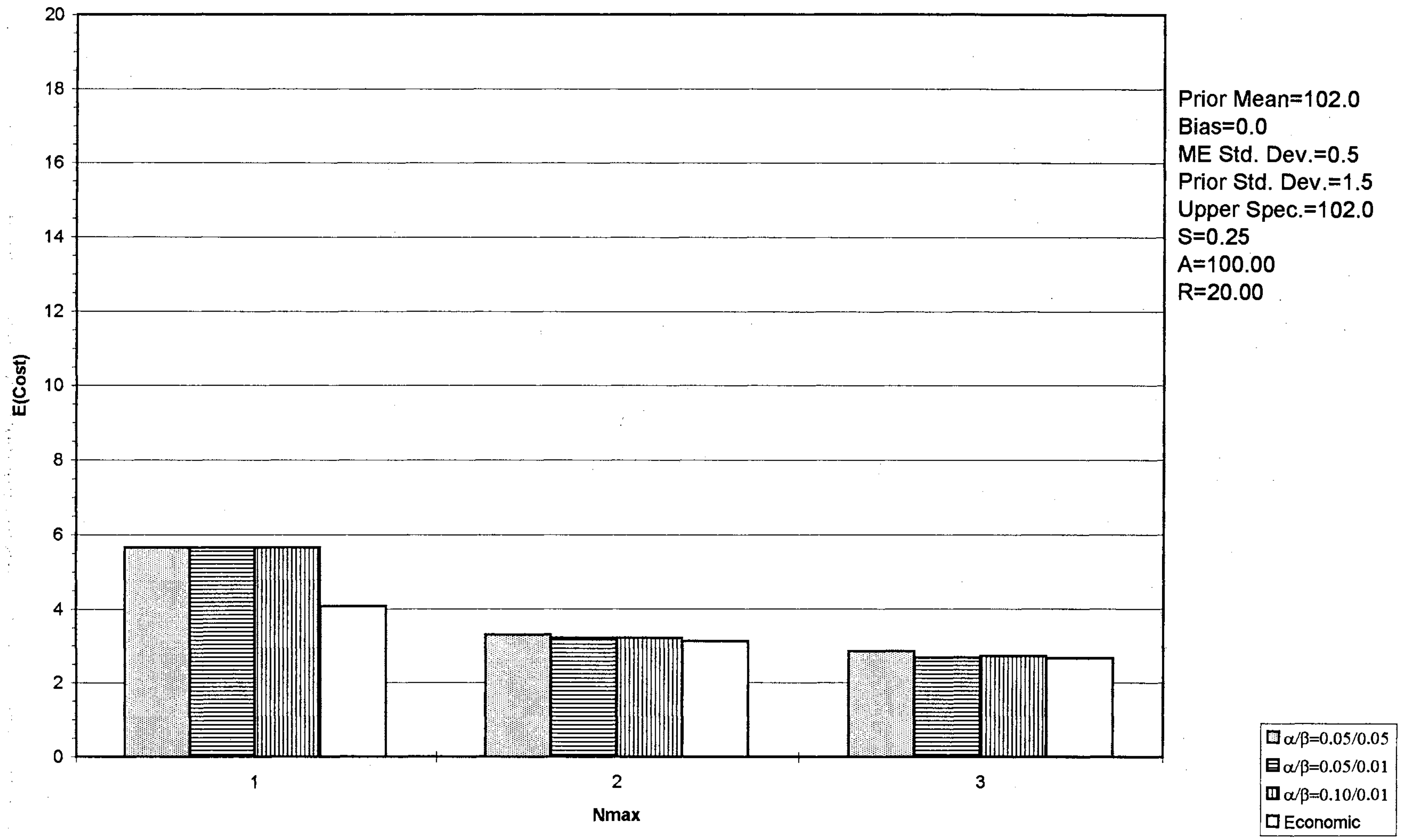

Figure 7.24 Economic & Statistical Cases: E(Cost) vs. n_{max} for Given Indifference Limits (101.0, 102.5)

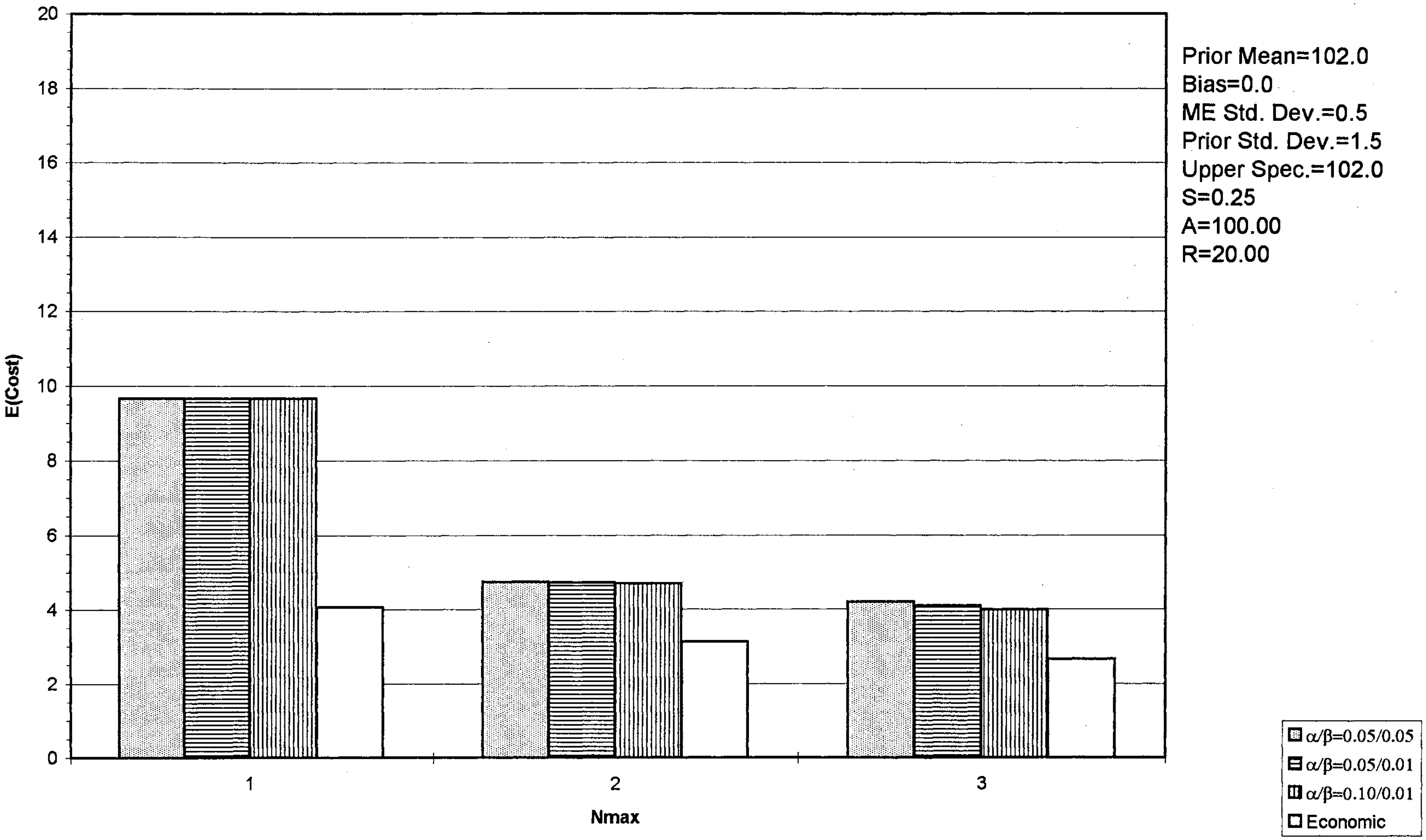

Figure 7.25 Economic & Statistical Cases: E(Cost) vs. n_{max} for Given Indifference Limits (101.5, 102.5)

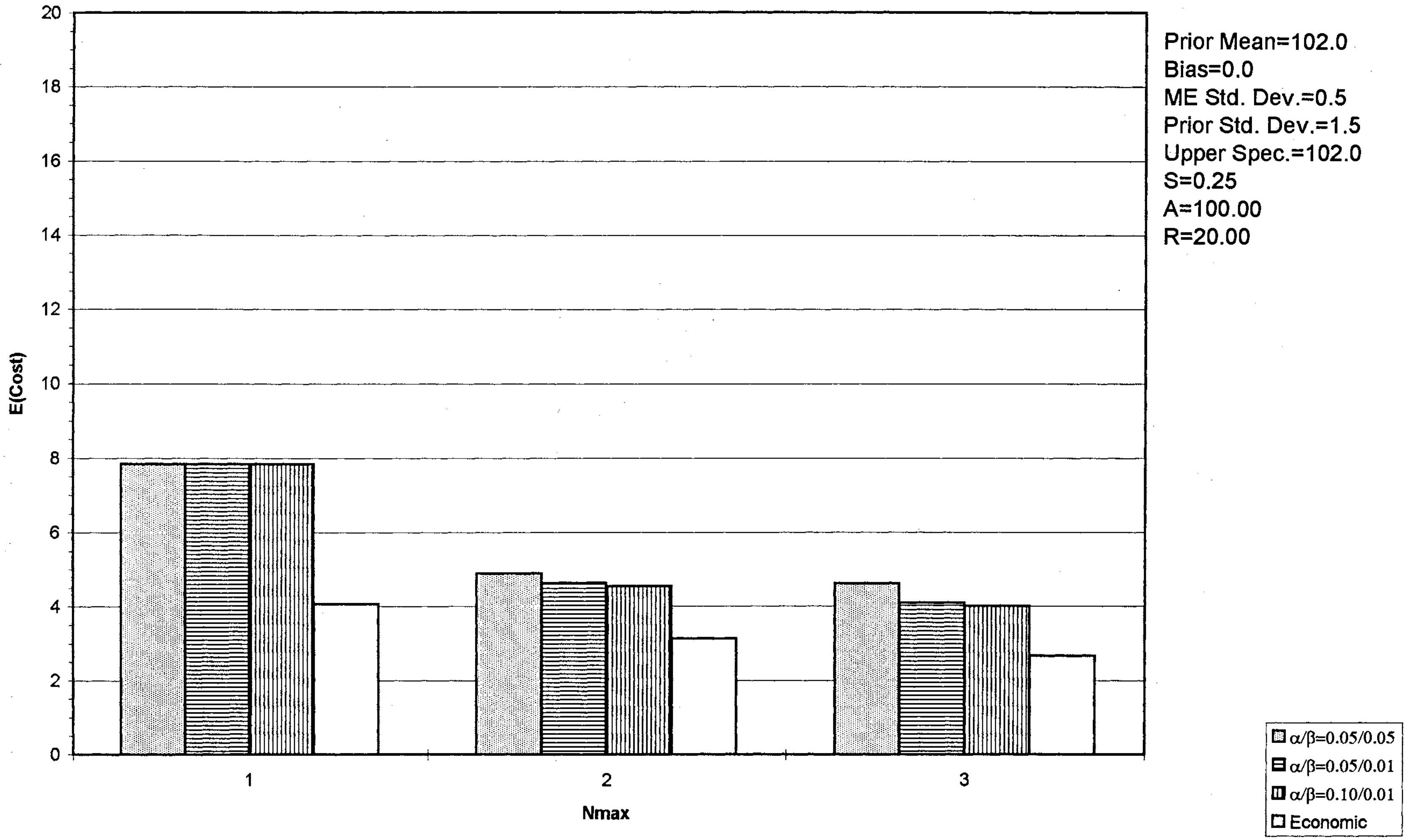

Figure 7.26 Economic & Statistical Cases: E(Cost) vs. n_{max} for Given Indifference Limits (101.0, 103.0)

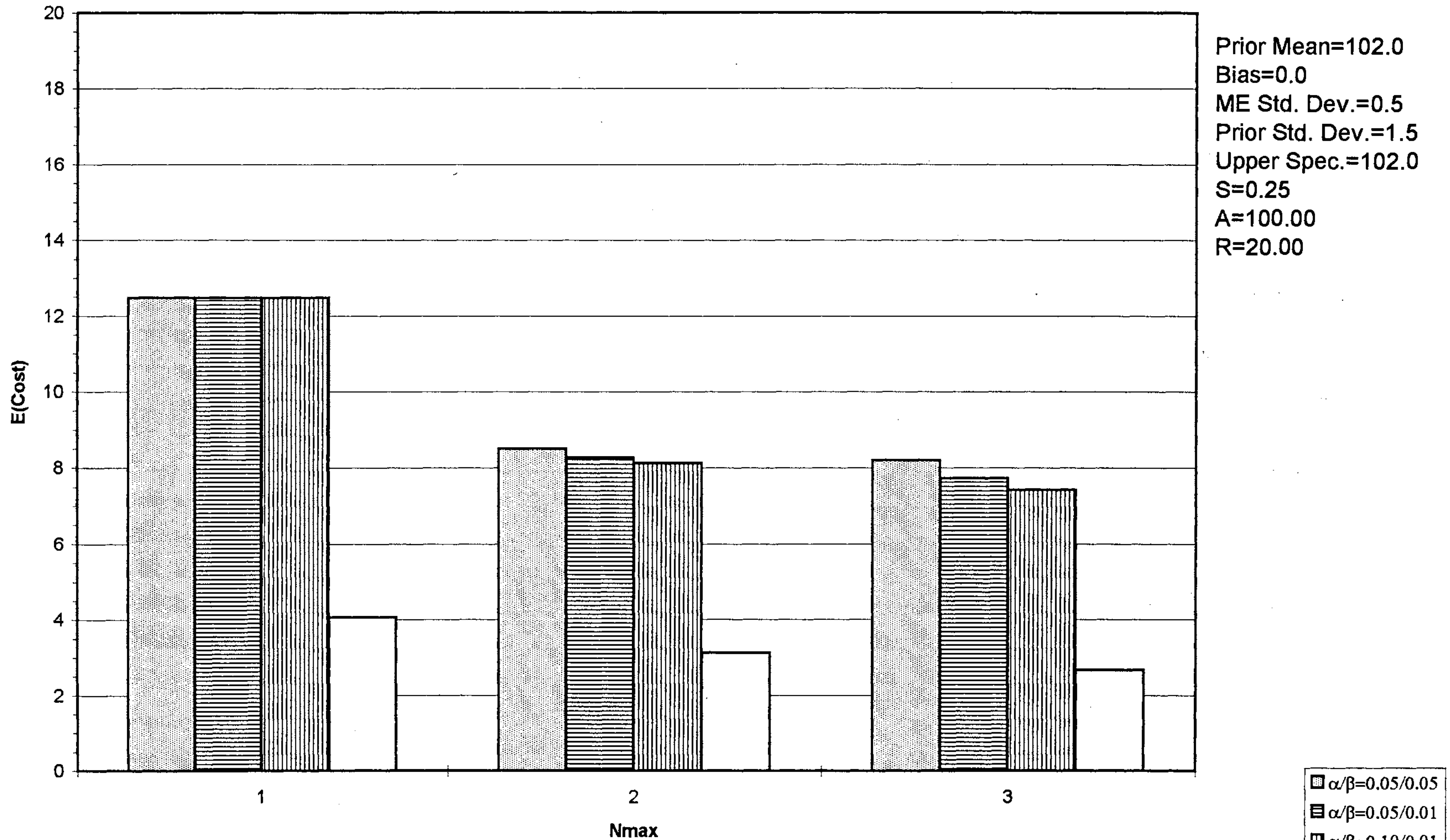

Figure 7.27 Economic & Statistical Cases: E(Cost) vs. n_{max} for Given Indifference Limits (101.5, 103.0)

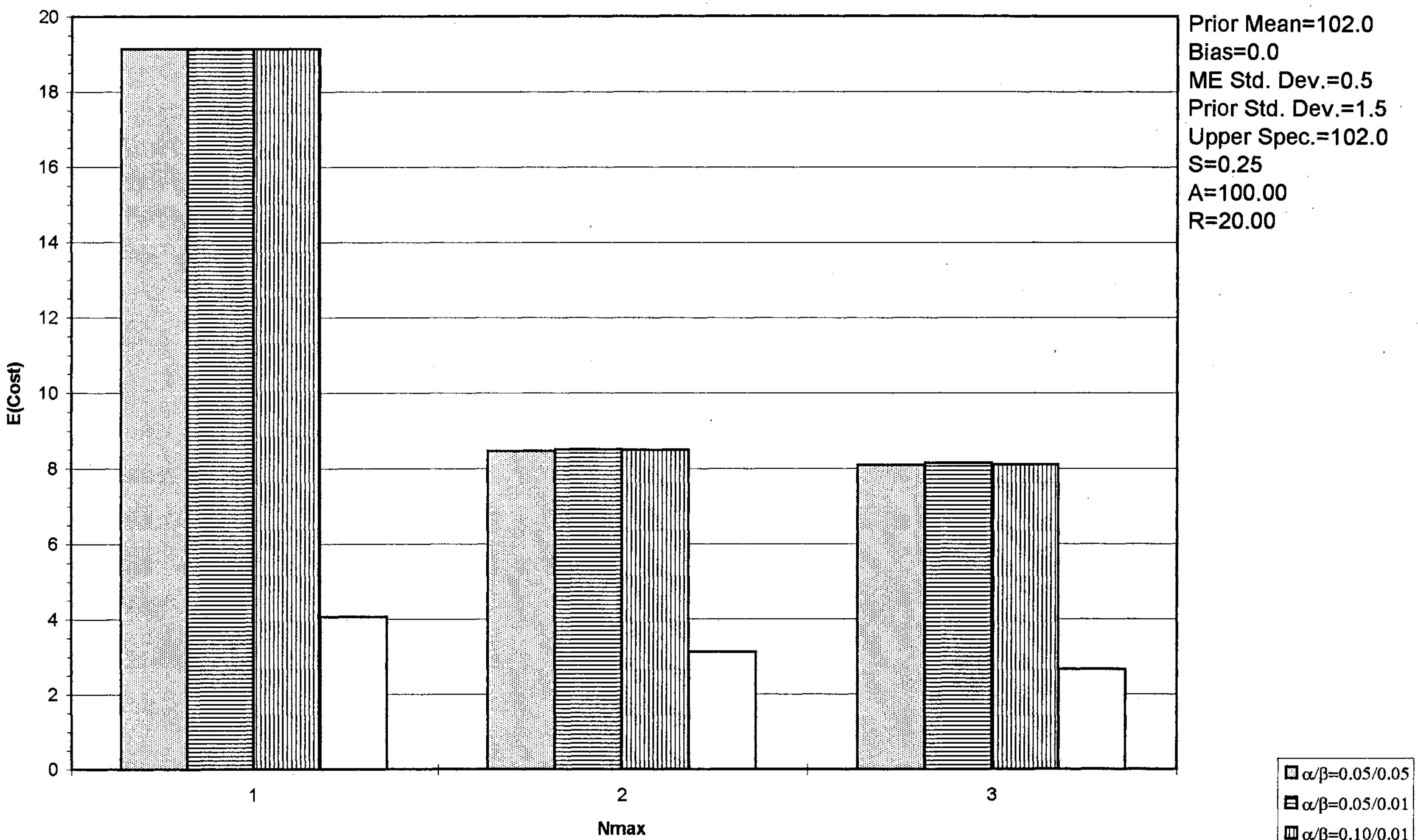

Figure 7.28 Economic & Statistical Cases: E(Cost) vs. n_{max} for Given Indifference Limits (102.0, 102.5)

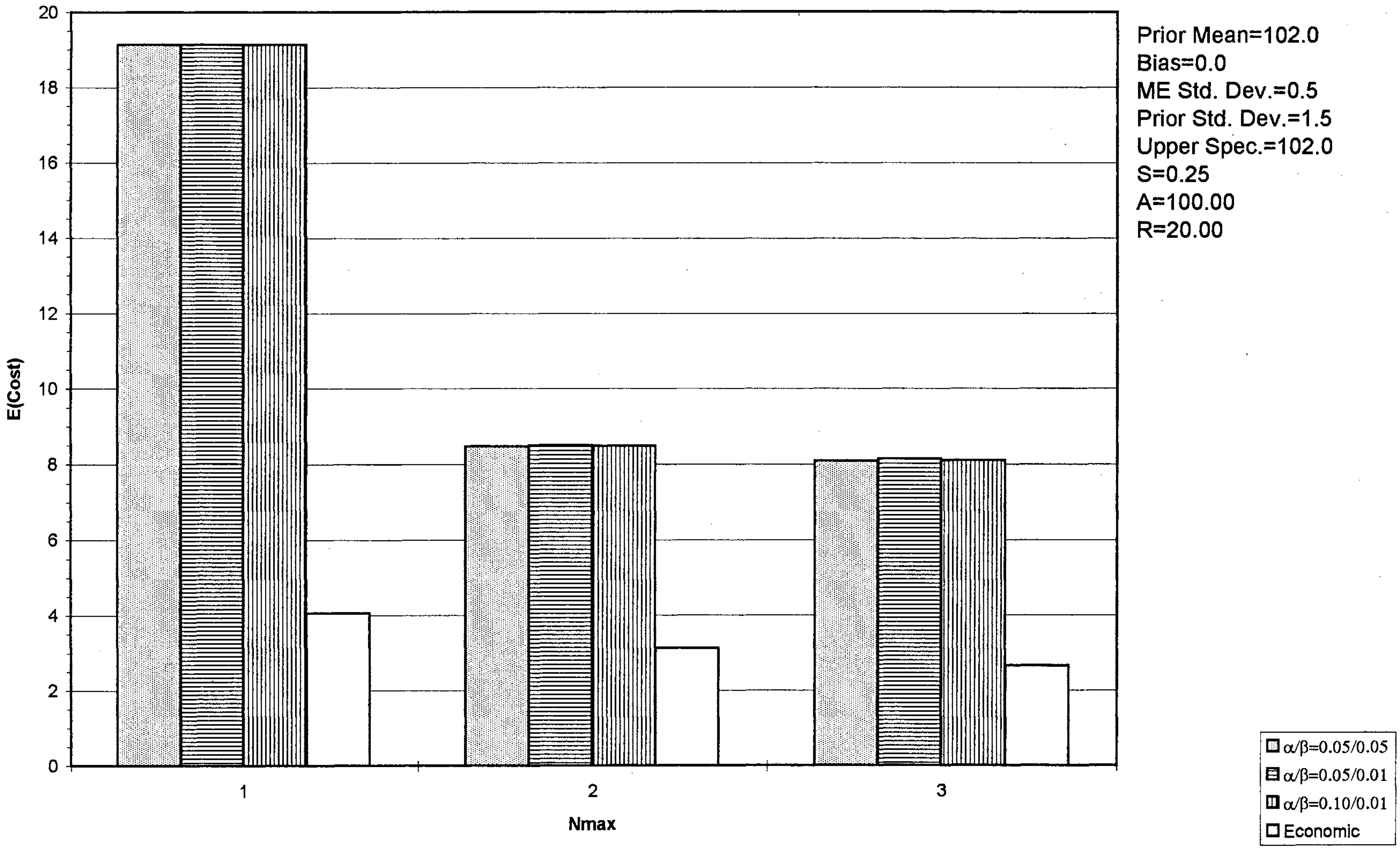

Figure 7.29 Economic & Statistical Cases: E(Cost) vs. n_{max} for Given Indifference Limits (102.0, 103.0)

The selection of the indifference limits, though somewhat indicative of decision risk, must be viewed slightly differently from the user's perspective. In Chapter 3, it is explained that the indifference limits define a zone between conforming and non-conforming batch values within which the user is indifferent as to the disposition of the batch. These limits define the null and alternative hypotheses values for the SPRT test and actually take the place of the specification limit. So, although the indifference limits reflect some of the false disposition risk, they are selected by the user to more practically differentiate the acceptable and unacceptable batch values. The use of these surrogate specifications is an intrinsic part of Wald's SPRT theory; the practically of their usage is left for the user's consideration.

For the test case in which the prior distribution centers exactly on the specification, the indifference limit values of (101.5, 102.0) provide the lowest expected costs for n_{max} values of 2 and 3. Taking n_{max}=2, a tighter search in this region gives the lowest expected total cost for the indifference limit pair of (101.3, 102.0). However, the economic optimal simulated (and theoretical) expected total cost is also less than this statistical case value.

The instances in which SPRT truncation is effected in the simulation runs are summarized in Table 7.3. This table shows the percentage of simulated batches (50,000 total for each n_{max}) for which a statistical disposition decision is not reached within the designated maximum number of observations (n_{max}). The Wald truncation rule, as discussed in Chapter 3, is used to make a quality determination in each of these cases. Note that the percentages decrease, for a given set of (α,β) and indifference limits, with increasing n_{max}. Also, wider indifference bands appear to result in less instances of no decision within the designated maximum iterations.

Table 7.3

Percentages of Simulation Trials Using the Wald Truncation Rule

α and β Pairs

	n_{max}=1			n_{max}=2			n_{max}=3		
Indifference Limits	α=0.05 β=0.05	α=0.05 β=0.01	α=0.10 β=0.01	α=0.05 β=0.05	α=0.05 β=0.01	α=0.10 β=0.01	α=0.05 β=0.05	α=0.05 β=0.01	α=0.10 β=0.01
101.0,102.0	0.3436	0.4154	0.3691	0.1673	0.2158	0.1882	0.0984	0.1346	0.1136
101.5,102.0	0.6473	0.7335	0.6612	0.3633	0.4453	0.4002	0.2459	0.3083	0.2734
101.0,102.5	0.2441	0.3026	0.2718	0.1012	0.1408	0.1209	0.0518	0.0805	0.0653
101.5,102.5	0.3628	0.4481	0.4045	0.1784	0.2348	0.2083	0.1044	0.1473	0.1259
101.0,103.0	0.1864	0.2356	0.2122	0.0664	0.0973	0.0821	0.0261	0.0464	0.0376
101.5,103.0	0.2424	0.3094	0.2801	0.1006	0.1429	0.1236	0.0478	0.0768	0.0627
102.0,102.5	0.6448	0.7700	0.7148	0.3610	0.4582	0.4154	0.2434	0.3168	0.2844
102.0,103.0	0.3448	0.4452	0.4082	0.1629	0.2221	0.1978	0.0962	0.1389	0.1192

Information on the appropriateness of the disposition decision dictated by SPRT truncation is given as output of the simulation program. Any incorrect decisions incur false disposition costs (A or R) which contribute to the costs of simulated statistical sampling.

7.4 SUMMARY

The economic model is assessed, using a practical upper specification example for various values of prior mean and economic plan cost components. The expected total cost of the economic sampling plan is greatest when the prior mean is equal to the specification, with decreasing expected costs as the mean moves from the specification in both the positive and negative directions. The makeup of the expected total cost (composed of sampling, false acceptance and false rejection components) varies depending on the relative location of the prior mean to the specification.

For a given set of sampling plan parameters, the economic model expected total cost decreases with increasing n_{max} (maximum number of observations allowed). This is because all cost sets which are examined in the research assume a relatively low iteration (observation) cost. In a practical case in which the iteration cost dominates false disposition costs, the optimum economic model will dictate, through the cutoff values, that an early decision is made (before n_{max}) and the expected total cost will be unchanging for all values of n_{max}.

Because the economic and statistical problem models are based on differing assumptions, it is difficult to conduct a direct comparison of results. The economic model assumes a prior batch distribution and makes use of explicit sampling costs associated with observation and incorrect batch disposition. Comparison of the two theoretical methods is accomplished through simulation, with known economic costs assessed to the statistical plan following the batch disposition decision. The two models are compared for a single cost set and a prior distribution mean equal to the upper specification limit.

The simulated expected total cost associated with the statistical decision model is examined for various values of tolerable risk (α and β) and for various indifference limit pairs (as required by the Wald SPRT sampling method). The expected statistical plan cost which results is compared to the optimal economic plan cost for disposition of the exact same simulated batch observation values. In all cases, the optimum economic sampling plan gives an expected total cost which is less than the statistical scenarios examined.

Within the various statistical scenarios which are simulated, expected total cost decreases with increasing n_{max} for a given set of plan parameters. In the case of single observation disposition ($n_{max}=1$), the values of α and β which are assumed do not affect the expected cost for a given set of indifference limits. Risk values which yield the lowest expected cost vary, depending on the pair of indifference limits which are utilized for the sampling scenario.

CHAPTER 8

SUMMARY AND CONCLUSION

8.1 SUMMARY

The primary objective of this research is to develop a sequential decision method for the situation of homogeneous batch disposition based on a single variable characteristic subject to measurement error relative to a single specification limit. In designing such a sampling plan, statistical and economic theoretical models are explored and compared. The original research objective specified a single upper specification; this writing and the computer program provided (Appendix A) also allow for the situation of a single lower specification limit.

Through exploration and development of the research problem, several original contributions are offered to the body of quality control literature pertinent to this problem. These are:

1) Investigation of the economic consequences of measurement error on the disposition of a homogeneous batch of product by a single variable characteristic relative to a single specification limit.

2) Application of SPRT theory to the known variance situation of homogeneous batch disposition based on a single variable characteristic subject to measurement error.

3) Exact application of the likelihood ratio test using the non-central t-distribution in the case of unknown measurement error variance. This is accomplished through the FORTRAN program provided.

4) Economic modeling of the sequential sampling of a single variable characteristic in the presence of measurement error for the purpose of disposition (either acceptance or rejection).

5) Economic optimization of sequential sampling plans and application of classical sequential statistical theory through a comprehensive FORTRAN program; expected cost analysis of user-designed sequential plans via the same program.

6) Comparison of sequential sampling plans based on economic parameters with plans derived from classical sequential statistical theory (founded on perceived levels of Types I and II error) through a computer simulation program (coded in FORTRAN).

These areas are highlighted and discussed in the following paragraphs.

Economic Consequences of Measurement Error

In practice, batch disposition as targeted in the research problem is often accomplished based on a single measurement observation. Although the economic effects of measurement error have been widely explored in regard to lot-by-lot sampling by attributes and variables, the area has been neglected in the batch sampling scenario.

SPRT Application

The statistical SPRT (Sequential Probability Ratio Test) method [59] is used in examining the known measurement variance inspection scenario. This sampling procedure has not previously been applied to the research problem.

Likelihood Ratio of Non-Central t-Distributions

In the case of unknown measurement system variance (representing composite hypotheses), exact solution of the problem using SPRT theory requires utilizing probabilities of the non-central t-distribution. Previous applications of sequential t-tests require table searches and/or approximations. The computer program provided as a product of this research effort represents an exact solution of the problem through the use of SPRT theory. This contribution is in addition to the previously unexplored application of the sequential statistical sampling theory to the research problem.

Economic Modeling of the Problem

The economic modeling of the research problem represents the single most significant contribution of this research writing. Through explicit definition of the costs associated with sampling and erroneous disposition decisions and application of Bayesian decision theory in consideration of batch historical data, optimal economic plans are developed. The economic plan design also requires a priori designation of a maximum number of observations which are to be tolerated in making the batch disposition decision. The economic model is developed using prior and posterior costing approaches of Bayesian Decision Theory.

Comprehensive Computer Program

The end-product of the research effort is a comprehensive computer program which applies the statistical and economic optimization models to the decision problem. The program also provides expected cost data for sequential economic plans which the user designs (rather than the optimal plan provided by the program).

Simulation Program

In order to assess the economic-based sampling plan against a plan based on statistical SPRT theory, a FORTRAN simulator is provided. The simulator draws observations from the prior product distribution and applies both the economic and statistical plans in reaching separate disposition decisions. The decisions are compared on economic and statistical bases.

8.2 RESULTS AND CONCLUSION

The optimum economic plan design provided as output from the computer program yields lowest expected total cost in all situations examined. Statistical scenarios are explored for various levels of error risk (α and β) and indifference limits as required by the simple hypothesis SPRT. In the economic case, expected costs are found for differing prior batch distributions and various input cost combinations. The statistical plans are compared to the optimum economic plans through the use of computer simulation programs which examine both models in terms of resulting costs.

An increase in maximum number of allowed observations produces decreasing expected costs in both the statistical and economic cases. All cost data used in the analysis assumes that the cost of an additional observation is low relative to the costs associated with incorrect decisions.

Within the statistical model, variations among risk levels have differing impacts on expected costs depending on the indifference levels which are specified by the user. That is, the specified levels of α and β must be considered in conjunction with the indifference limits to minimize the expected cost of the plan.

Economic expected costs are explored for various values of the mean of the prior distribution. Additionally, various economic input costs are examined. Expected plan costs are highest when the prior batch distribution mean is close to the specification limit. Around the pivot point of the specification, expected costs of erroneous disposition are symmetrical, as are the economic decision limits selected for economic sampling.

The comprehensive FORTRAN computer program which is presented as an end-product of this research facilitates economic plan optimization and statistical plan implementation. Economic plans may be designed for a maximum of three measurement observations. The statistical routine accepts a maximum of fifty observations in dictating a disposition decision. The statistical routine uses truncated SPRT theory in the event that no decision is reached prior to the observation maximum (as specified by the user, up to fifty iterations). An additional program module allows the user to create his/her own economic sampling plan and view expected cost information based on user inputs.

8.3 FUTURE RESEARCH OPPORTUNITIES

This research effort exposes several additional research opportunities existing in this area. Specifically,

1) Consideration of two-sided specifications in the case of homogeneous batch disposition by a single variable characteristic in the presence of measurement error.

2) Further increases in maximum allowable observations in further optimizing the economic problem for all n_{max}.

3) Theoretical optimization of risk levels and indifference limits specified in the statistical problem. Although expected statistical costs (simulated) exceeded the optimum economic situation in all cases explored, it is believed that an optimal statistical plan will produce expected costs which approach the optimal economic plan.

4) Modification of the computer program to provide application of the optimal or user-specified economic plan. The current program identifies the optimal plan

but does not accept observation input and dictate the appropriate disposition decision.

5) Consideration of the known measurement error variance case in which truncation is accomplished through the use of "Wedge Plans" rather than the Wald truncation rule [2, 59] utilized in the research.

There are other related areas in which the problem may be extended to provide additional research opportunities. It is hoped that the information presented in this research effort represents a significant contribution to the area of quality control.

REFERENCES

[1] Airey, J. R. (1931). Tables of Hh Functions. British Association Mathematical Tables, 1. Office of the British Association, Burlington House, W. 1, London.

[2] Baker, A. G. (1950). "Properties of Some Tests in Sequential Analysis". *Biometrika* 37, pp. 334-346.

[3] Barnett, V. (1973). "Bayesian, and Decision Theoretic, Methods applied to Industrial Problems". *The Statistician* 22, pp. 199-227.

[4] Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis*, from the Springer Series in Statistics. Springer-Verlag, New York.

[5] Billard, L. and Vagholkar, M. K. (1969). "A Sequential Procedure for Testing a Null Hypothesis against a Two-sided Alternative Hypothesis". *Journal of the American Statistical Association* (B)31, pp. 285-294.

[6] Boucher, T. O. and Jafari, M. A. (1992). "A Note on Economic Process Levels Under Acceptance Sampling by Variables". *International Journal of Production Economics* 28, pp. 171-174.

[7] Case, K. E. and Bennett, G. K. (1977). "The Economic Effect of Measurement Error on Variables Acceptance Sampling". *International Journal of Production Research* 15, pp. 117-128.

[8] Case, K. E. and Keats, J. B. "Attributes Sampling Based Upon Costs and Lot History". *1981 ASQC Quality Congress Transactions - San Francisco*, pp. 510-518.

[9] Chen, S. (1981). "Modeling and Design of Economically Based Double Sampling Plans". Doctoral Thesis, Oklahoma State University, Stillwater, Oklahoma.

[10] Chou, Y and Owen, D. B. (1984). "One-Sided Confidence Regions on the Upper and Lower Tail Areas of the Normal Distribution". *Journal of Quality Technology* 16, pp. 150-158.

[11] Collins, Jr., R. D, Case, K. E. and Bennett, G. K. (1978). "Inspection Error and Its Adverse Effects: A Model with Implications for Practitioners". *AIIE Transactions* 10, pp. 2-9.

[12] Cooper, B. E. (1968). "The Normal Integral". Algorithm AS-2 in *Applied Statistics* (Series C) 17, pp. 186-188.

[13] Cooper, B. E. (1968). "The Integral of Student's t-distribution". Algorithm AS-3 in *Applied Statistics* (Series C) 17, pp. 189-192.

[14] Cooper, B. E. (1968). "An Auxiliary Function for Distribution Integrals". Algorithm AS-4 in *Applied Statistics* (Series C) 17, pp. 190-192.

[15] Cooper, B. E. (1968). "The Integral of a Non-Central t-distribution". Algorithm AS-5 in *Applied Statistics* (Series C) 17, pp. 193-194.

[16] Cox, D. R. (1952). "Sequential Tests for Composite Hypotheses". *Proceedings of the Cambridge Philosophical Society* 48, pp. 290-299.

[17] David, H. T., Fay, E. A. and Walsh, J. E. (1959). "Acceptance Inspection by Variables When the Measurements are Subject to Errors". *Annals of the Institute of Statistical Mathematics* 10, pp. 107-129.

[18] David, H. T. and Kruskal, W. H. (1956). "The WAGR Sequential t-Test Reaches a Decision with Probability One". *Annals of Mathematical Statistics* 27, pp. 797-805.

[19] DeGroot, M. H. and Nadler, J. (1958). "Some Aspects of the Use of the Sequential Probability Ratio Test". *Journal of the American Statistical Association* 53, pp. 187-199.

[20] Diviney, T. E. and David, N. A. (1963). "A Note on the Relationship Between Measurement Error and Product Acceptance". *Journal of Industrial Engineering* 14, pp. 218-219.

[21] Eagle, A. R. (1954). "A Method for Handling Errors in Testing and Measuring". *Industrial Quality Control*, March, pp. 10-15.

[22] Fertig, K. W. and Mann, N. R. (1983). "Sample Sizes for Variables Sampling Plans Based on a Decision Theoretic Approach". *Journal of Quality Technology* 15, pp. 63-67.

[23] Grubbs, F. E. (1948). "On Estimating Precision of Measuring Instruments and Product Variability". *Journal of the American Statistical Association* 43, pp. 243-264.

[24] Grubbs, F. E. and Coon, H. J. (1954). "On Setting Test Limits Relative to Specification Limits". *Industrial Quality Control*, March, pp. 15-20.

[25] Hahn, G. J. (1982). "Removing Measurement Error in Assessing Conformance to Specifications". *Journal of Quality Technology* 14, pp. 117-121.

[26] Hamaker, H. (1979). "Acceptance Sampling for Percent Defective by Variables and by Attributes". *Journal of Quality Technology* 11, pp. 139-148.

[27] Jaech, J. L. (1984). "Removing the Effects of Measurement Errors in Constructing Statistical Tolerance Intervals". *Journal of Quality Technology* 16, pp. 69-73.

[28] Johnson, N. L. (1961). "Sequential Analysis: A Survey". *Journal of the American Statistical Association* (A) 124, pp. 372-411.

[29] Johnson, N. L. and Welch, B. L. (1940). "Applications of the Non-Central t-Distribution". *Biometrika* 31, pp. 362-389.

[30] Kirkpatrick, R. L. (1970). "Confidence Limits on a Percent Defective Characterized by Two Specification Limits". *Journal of Quality Technology* 2, pp. 150-155.

[31] Ladany, S. P. (1976). "A Practical Meaning of the Bayesian Decision Making Approach to Acceptance Sampling". *Journal of Quality Technology* 8, pp. 127-132.

[32] Lieberman, G. J. and Resnikoff, G. J. (1950). "Sampling Plans for Inspection by Variables". *Journal of the American Statistical Association* 50, pp. 457-516.

[33] Lotti, M. A. (1982). "Measurement Variability and Product Acceptance". *Quality Progress*, September, pp. 38-41.

[34] Mathematical and Statistical Principles Underlying Military Standard 414 (1958). Technical Report, Office of the Assistant Secretary of Defense (Supply and Logistics), Washington, D.C.

[35] Mee, R. W. (1984). "Tolerance Limits and Bounds for Proportions Based on Data Subject to Measurement Error". *Journal of Quality Technology* 16, pp. 74-80.

[36] Mee, R. W., Owen, D. B. and Shyu, J. (1986). "Confidence Bounds for Misclassification Probabilities Based on Data Subject to Measurement Error". *Journal of Quality Technology* 18, pp. 29-40.

[37] Mei, W., Case, K. E. and Schmidt, J. W. (1975). "Bias and Imprecision in Variables Acceptance Sampling: Effects and Compensation". *International Journal of Production Research* 13, pp. 327-340.

[38] MIL-STD-414 (1957) "Sampling Procedures and Tables for Inspection by Variables for Percent Defective". Department of Defense, Washington, D.C.

[39] Moskowitz, H. and Tang, K. (1992). "Bayesian Variables Acceptance-Sampling Plans: Quadratic Loss Function and Step-Loss Function". *Technometrics* 34, pp. 340-347.

[40] Nelder, J. A. and Mead, R. (1965). "A Simplex Method for Function Minimization". *The Computer Journal* 7, pp. 308-313.

[41] Odeh, R. E., Chou, Y. and Owen, D. B. (1989). "Sample-Size Determination for Two-Sided B-Expectation Tolerance Intervals for a Normal Distribution". *Technometrics* 31, pp. 461-468.

[42] Olsson, D. M. (1974). "A Sequential Simplex Program for Solving Minimization Problems". *Journal of Quality Technology* 6, pp. 53-57.

[43] Owen, D. B. (1965). "A Special Case of a Bivariate Non-Central t-Distribution". *Biometrika* 52, pp. 437-446.

[44] Owen, D. B. (1967). "Variables Sampling Plans Based on the Normal Distribution". *Technometrics* 9, pp. 417-423.

[45] Owen, D. B. and Chou, Y. (1983). "Effect of Measurement Error and Instrument Bias on Operating Characteristics for Variables Sampling Plans". *Journal of Quality Technology* 15, pp. 107-117.

[46] Owen, D. B. and Weisen, J. M. (1959). "A Method of Computing Bivariate Normal Probabilities with an Application to Handling Errors in Testing and Measurement". *Bell System Technical Journal* 38, pp. 553-572.

[47] Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. (1986). *Numerical Recipes*. Cambridge University Press, New York, NY.

[48] Resnikoff, G. J. and Lieberman, G. J. (1957). *Tables of the Non-Central t-Distribution*. Stanford University Press, Stanford, CA.

[49] Rushton, S. (1950). "On a Sequential t-Test". *Biometrika* 37, pp. 326-333.

[50] Schilling, E. G. (1982). *Acceptance Sampling in Quality Control*, Marcel Dekker, New York.

[51] Schmidt, Jr., J. W., Bennett, G. K. and Case, K. E. (1980). "A Three Action Cost Model for Acceptance Sampling by Variables". *Journal of Quality Technology* 12, pp. 10-18.

[52] Schmidt, J. W., Case, K. E. and Bennett, G. K. (1974). "The Choice of Variables Sampling Plans Using Cost Effective Criteria". *AIIE Transactions* 6, pp. 178-184.

[53] Schneiderman, M. A. and Armitage, P. (1962). "Closed Sequential t-Tests". *Biometrika* 49, pp. 359-366.

[54] Sobel, M. and Wald, A. (1949). "A Sequential Decision Procedure for Choosing One of Three Hypotheses Concerning the Unknown Mean of a Normal Distribution". *Annals of Mathematical Statistics* 10, pp. 502-522.

[55] Stegun, I. A. and Zucker, R. (1970). "Automatic Computing Meghods for Special Functions". *Journal of Research of the National Bureau of Standards - B. Mathematical Sciences 74B*, pp. 211-224.

[56] *Tables of the Confluent Hypergeometric Function F(n/2,1/2,x) and Related Functions* (1949). Applied Mathematics Series 3, National Bureau of Standards, Washington, D.C.

[57] *Tables to Facilitate Sequential t-Tests* (1951). Applied Mathematics Series 7, National Bureau of Standards, Washington, D.C.

[58] Tang, K. and Schneider, H. (1987). "The Effects of Inspection Error on a Complete Inspection Plan". *IIE Transactions* 19, pp. 421-428.

[59] Wald, A. (1947). *Sequential Analysis*. John Wiley & Sons, Inc., New York, NY.

[60] Wallis, W. A. (1947). "Use of Variables in Acceptance Inspection for Percent Defective" in *Techniques of Statistical Analysis*, (C. Eisenhart, M. Hastay and W. A. Wallis, eds.). Statistical Group, Columbia University, McGraw-Hill, New York, NY, pp. 3-93.

[61] Weingarten, H. (1982). "Confidence Intervals for the Percent Nonconforming Based on Variables Data". *Journal of Quality Technology* 14, pp. 207-210.

[62] Wetherill, G. B. (1975). *Sequential Methods in Statistics*. Chapman and Hall, Ltd., London.

[63] Wetherill, G. B. and Campling, G. E. G. (1966). "The Decision Theory Approach to Sampling Inspection". *Journal of the Royal Statistical Association, Series B* 28, pp. 381-416.

APPENDICES

APPENDIX A

COMPREHENSIVE COMPUTER PROGRAM FOR ECONOMIC PLAN OPTIMIZATION AND STATISTICAL BATCH DISPOSITION (FORTRAN Code Listing)

```
c**************************************************
c    main for overseeing economic and statistical program
c    modules
c**************************************************
     igo=0
5    continue
     print*
     print*,'         Sequential Testing Program'
     print*
     print*,'Please select one of the following options:'
     print*
     print*,' 1  Economic Testing'
     print*,'    * Plan Optimization'
     print*,'    * Expected Cost Calculation'
     print*
     print*,' 2  Statistical Testing'
     print*,'    * Known Measurement Error Variance'
     print*,'    * Unknown Measurement Error Variance'
     print*
     print*,' 3  Exit Program'
     print*
     read(*,210,err=5)igo
     if(igo.lt.1.or.igo.gt.3)then
        print*,'**** Invalid Entry.  Please Reenter. ****'
        print*
        goto 5
     endif
     print*
     if(igo.eq.3)goto 200
     if(igo.eq.1)then
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
50      continue
        print*
        print*,'         Economic Testing Plans'
        print*
        print*,'Please select one of the following options:'
        print*
        print*,' 1  Economic Parameter Optimization'
        print*,'    (Maximum Iterations Limited to Three)'
        print*
        print*,' 2  Expected Costs Calculation for'
        print*,'    User-Entered Plan'
        print*,'    (Maximum Iterations Limited to Three)'
        print*
        print*,' 3  Return to Main Program'
        print*
        read(*,210,err=5)igo
        if(igo.lt.1.or.igo.gt.3)then
           print*,'Invalid Entry.  Please Reenter.'
           print*
           goto 50
        endif
        if(igo.eq.3)goto 5
        if(igo.eq.2)then
```

```
          call calcst(igo)
          goto 50
        endif
60      print*
        print*,'What is the maximum number of measurement'
        print*,'iterations which you are willing to make?'
        print*,'Enter 1, 2 or 3'
        print*
        read(*,*,err=60)nmx
        if(nmx.lt.1.or.nmx.gt.3)then
          print*,'**** Invalid Entry.  Please Reenter. ****'
          print*
          goto 60
        endif
        if(nmx.eq.1)call a2n1mgt(nmx)
        if(nmx.eq.2)call a2n2mgt(nmx)
        if(nmx.eq.3)call a2n3mgt(nmx)
        goto 50
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      endif
      if(igo.eq.2)then
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
100     continue
        print*
        print*,'          Statistical Testing Plans'
        print*
        print*,'Please select one of the following options:'
        print*
        print*,' 1  Known Measurement Error Variance'
        print*,'    Sequential Data Entry and Batch Disposition'
        print*
        print*,' 2  Unknown Measurement Error Variance'
        print*,'    Sequential Data Entry and Batch Disposition'
        print*
        print*,' 3  Return to Main Program'
        print*
        read(*,210,err=100)igo
        if(igo.lt.1.or.igo.gt.3)then
          print*,'**** Invalid Entry.  Please Reenter. ****'
          print*
          goto 100
        endif
        if(igo.eq.3)goto 5
        if(igo.eq.1)then
          call known(igo)
          goto 100
        endif
        if(igo.eq.2)then
          call unkn(igo)
          goto 100
        endif
      endif
200   continue
      print*
```

```
      print*,'Exiting Program...'
      print*
210   format(i1)
      return
      end
c     ****************************************************
c     ****************************************************
c     ****************************************************
c     sub for the economic case of nmax=1
c     called from main
c
c     ****************************************************
      subroutine a2n1mgt(n)
      dimension thta(4),tau(4),pcs(3)
      real*8 a,b,tcnewlo,xmin(2)
      character iopt
      character*5 spec
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a,b
      common /ncur/ j
      spec='Upper'
c     call routine to input parameters
      call vardef(u,thta(1),tau(1),bias,sme,s1,a2,r2,nspc)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
c     this to account for lower spec by symmetry, only
c     Use given spec, but find for symmetrical prior mean
      if(nspc.eq.2)then
        thta(1)=2*u-thta(1)
        spec='Lower'
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      n=1
c
      tau(2)=sqrt(1./(1./sme**2.+1./tau(1)**2.))
      j=1
c     j is the iteration number
c     set limits for the conditional error curves
      call setabx(a,b,j)
      if(b.le.a)then
        tcnewlo=s1
        pcs(1)=s1
        pcs(2)=0
        pcs(3)=0
        xmin(1)=b+(.5*(a-b))
        goto 200
      endif
      call cmaxn1(a,b,xmin(1))
      call cost(xmin(1),tcost,pcs)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
c     this to account for lower spec by symmetry, only
c     change prior mean back before print
```

```
c     find c1 by symmetry around the specification
      if(nspc.eq.2)then
        xmin(1)=2*u-xmin(1)
        thta(1)=2*u-thta(1)
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      write(*,610)xmin(1),xmin(1)-bias
      write(*,612)
      write(*,614)pcs(1)
      write(*,616)pcs(2)
      write(*,618)pcs(3)
      write(*,620)tcost
90    print*,'Send output to file? (Y/N)'
      read(*,500)iopt
      if(iopt.ne.'Y'.and.iopt.ne.'N'.and.iopt.ne.'y'.and.
     &iopt.ne.'n')goto 90
      if(iopt.eq.'n'.or.iopt.eq.'N')goto 200
      write(2,510)thta(1)
      write(2,520)tau(1)
      write(2,530)spec,u
      write(2,540)sme
      write(2,550)bias
      write(2,560)
      write(2,570)s1
      write(2,580)a2
      write(2,590)r2
      write(2,600)
      write(2,610)xmin(1),xmin(1)-bias
      write(2,612)
      write(2,614)pcs(1)
      write(2,616)pcs(2)
      write(2,618)pcs(3)
      write(2,620)tcost
200   continue
500   format(a1)
510   format(' Prior Mean =          ',f16.8)
520   format(' Prior Std. Dev =      ',f16.8)
530   format(1x,a5,' Specification = ',f16.8)
540   format(' Measurement Error Std. Dev. = ',f13.8)
550   format(' Measurement Error Mean (Bias = ',f13.8)
560   format(' Input Costs:')
570   format('    Iteration (S) =     ',f12.4)
580   format('    False Accept (A) =  ',f12.4)
590   format('    False Reject (R) =  ',f12.4)
600   format(' *************************************')
610   format(' Zero-Bias C1 = ',f16.8,5x,'Bias Adj C1 = ',f16.8)
612   format(' Expected Plan Costs:')
614   format('    Sampling on 1 =     ',f12.4)
616   format('    False Accept on 1 = ',f12.4)
618   format('    false Reject on 1 = ',f12.4)
620   format(' Expected Total Cost =  ',f12.4)
      return
      end
```

```
c    ****************************************************
c    ****************************************************
c    ****************************************************
c    sub which evaluates the conditional probability
c    at a point, x
c    flg=flag indicating type i or ii error
c    j=iteration number
c    ffunc=function value which is returned
c    functions obtained from mcad
c
c    calls sub errint from stegun and zucker
c    must bring in a x to this sub, which is obtained
c    from gaussian quadrature
c*****************************************************
c
c
      subroutine func(j,flg,x,ffunc)
c
      common /parms/ tau,sme,thta
c
      real*8 x,ffunc,from,pcdx,pi
      dimension tau(4),thta(4)
c
c     eqn common to both errors (type I and II)
c
c
      pi=3.141592654d0
c     call to pcmu simply gets another part of the function
      call pcmu(x,flg,from)
      pcdx=1.d0/(sqrt(2.d0*pi)*sqrt(tau(j)**2.+sme**2.))*
     &exp(-.5d0*((x-thta(j))/sqrt(tau(j)**2.+sme**2.))**2.)
      ffunc=from*pcdx
 25   continue
      return
      end
c*****************************************************
c    ************************************************
c    ************************************************
c     sub for solution of the nmax=2 problem using
c     approach 2 (posterior costs).
c*****************************************************
c     nmx= maximum iteration requested
c
      subroutine a2n2mgt(nmx)
c
      dimension thta(4),tau(4)
      real*8 start(10),step(10),xmin(20),xsec(20),tcnewlo,
     &tcsec,reqmin,a,b,x1,c2,pcs(9),xtmp,tcst
      character iopt
      character*5 spec
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a,b
      common /ncur/ j
      common /cult/ c2
```

```
      spec='Upper'
c     call sub to initialize input parameters
c     nspc returned as the type of spec (upper or lower)
      call vardef(u,thta(1),tau(1),bias,sme,s1,a2,r2,nspc)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
c     this to account for lower spec by symmetry, only
c     Use given spec, but find for symmetrical prior mean
      if(nspc.eq.2)then
         thta(1)=2*u-thta(1)
         spec='Lower'
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
      n=1
      nmx=2
      x1=u
c
c      calculate posterior parameters based on x1=u
c      purpose is to find c2 (which does not vary with x1,
c      so the value of x1 used is irrelevant
c
      do 30 i=2,nmx+1
      tau(i)=sqrt(1./(1./sme**2.+1./tau(i-1)**2.))
30    continue
c     find posterior distribution parameters at u
c     j=iteration number
      call post(nmx,x1)
      j=1
      call setabx(a,b,j)
c     if b<a, this implies that the two conditional error curves
c     do not intersect (appreciably) and the cost will simply be
c     that of an iteration
      if(b.le.a)then
        tcnewlo=s1
        pcs(1)=s1
        pcs(2)=0.d0
        pcs(3)=0.d0
        pcs(4)=0.d0
        pcs(5)=0.d0
        pcs(6)=0.d0
        xmin(1)=b+(.5*(a-b))
        goto 200
      endif
c     store the value of c2 in the variable xmin(3)
c     should be between a and b
c     set x1=u to locate c2 (shouldn't matter)
      j=2
      call cmaxn2(a,b,x1,xmin(3))
      c2=xmin(3)
c     n=2 unknowns for nelder-mead search, c1l(1) and c1h(2)
      n=2
      start(1)=(a+c2)/2.d0
      start(2)=(b+c2)/2.d0
```

```
      step(1)=(b-a)/2.d0
      step(2)=step(1)
      reqmin=.000001d0
      icount=200
      call nelmin1(n,start,xmin,xsec,tcnewlo,
     &tcsec,reqmin,step,icount,pcs)
      call clg2a2(xmin,tcst,pcs)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c     this to account for lower spec by symmetry, only
c     change prior mean back before print
c     find c1 by symmetry around the specification
      if(nspc.eq.2)then
        xtmp=xmin(1)
        xmin(1)=2*u-xmin(2)
        xmin(2)=2*u-xtmp
        xmin(3)=2*u-xmin(3)
        thta(1)=2*u-thta(1)
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      write(*,610)xmin(3),xmin(3)-bias
      write(*,614)xmin(1),xmin(1)-bias
      write(*,616)xmin(2),xmin(2)-bias
      write(*,618)
      write(*,620)pcs(1)
      write(*,622)pcs(2)
      write(*,624)pcs(3)
      write(*,626)pcs(4)
      write(*,628)pcs(5)
      write(*,630)pcs(6)
      write(*,650)tcnewlo
90    print*,'Send output to file? (Y/N)'
      read(*,500)iopt
      if(iopt.ne.'Y'.and.iopt.ne.'N'.and.iopt.ne.'y'.and.
     &iopt.ne.'n')goto 90
      if(iopt.eq.'n'.or.iopt.eq.'N')goto 200
      write(2,510)thta(1)
      write(2,520)tau(1)
      write(2,530)spec,u
      write(2,540)sme
      write(2,550)bias
      write(2,560)
      write(2,570)s1
      write(2,580)a2
      write(2,590)r2
      write(2,600)
      write(2,610)xmin(3),xmin(3)-bias
      write(2,614)xmin(1),xmin(1)-bias
      write(2,616)xmin(2),xmin(2)-bias
      write(2,618)
      write(2,620)pcs(1)
      write(2,622)pcs(2)
      write(2,624)pcs(3)
```

```
      write(2,626)pcs(4)
      write(2,628)pcs(5)
      write(2,630)pcs(6)
      write(2,650)tcnewlo
200   continue
500   format(a1)
510   format(' Prior Mean =          ',f16.8)
520   format(' Prior Std. Dev =      ',f16.8)
530   format(1x,a5,' Specification = ',f16.8)
540   format(' Measurement Error Std. Dev. =  ',f13.8)
550   format(' Measurement Error Mean (Bias) = ',f13.8)
560   format(' Input Costs:')
570   format('    Iteration (S) =     ',f12.4)
580   format('    False Accept (A) =  ',f12.4)
590   format('    False Reject (R) =  ',f12.4)
600   format(' *************************************')
610   format(' Zero-Bias C2 =   ',f16.8,5x,'Bias Adj C2 =   ',f16.8)
614   format(' Zero-Bias C1,L = ',f16.8,5x,'Bias Adj C1,L = ',f16.8)
616   format(' Zero-Bias C1,H = ',f16.8,5x,'Bias Adj C1,H = ',f16.8)
618   format(' Expected Plan Costs:')
620   format('    Sampling on 1 =     ',f12.4)
622   format('    False Accept on 1 = ',f12.4)
624   format('    False Reject on 1 = ',f12.4)
626   format('    Sampling on 2 =     ',f12.4)
628   format('    False Accept on 2 = ',f12.4)
630   format('    False Reject on 2 = ',f12.4)
650   format(' Expected Total Cost =  ',f12.4)
      return
      end
c*****************************************************
c    *************************************************
c    *************************************************
c    sub for solution of the nmax=3 problem using
c    approach 2 (posterior costs).
c*****************************************************
c
      subroutine a2n3mgt(nmx)
c
      dimension thta(4),tau(4)
      real*8 start(10),step(10),xmin(20),xsec(20),tcnewlo,
     &tcsec,reqmin,a,b,x1,x2,c3,pcs(9),xtmp,tcst
      character iopt
      character*5 spec
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a,b
      common /ncur/ j
      common /cult/ c3
      spec='Upper'
c    call sub to initialize input parameters
c    nspc returned as the type of spec (upper or lower)
      call vardef(u,thta(1),tau(1),bias,sme,s1,a2,r2,nspc)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c    this to account for lower spec by symmetry, only
```

```
c    Use given spec, but find for symmetrical prior mean
     if(nspc.eq.2)then
        thta(1)=2*u-thta(1)
        spec='Lower'
     endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
     nmx=3
     x1=dble(u)
     x2=dble(u)
c
c     calculate posterior parameters based on x1=u, x2=u
c     purpose is to find c3 (which does not vary with x1 or x2
c     so the value of x1 used is irrelevant
c
     do 30 i=2,nmx+1
     tau(i)=sqrt(1./(1./sme**2.+1./tau(i-1)**2.))
30    continue
c    find posterior distribution parameters at u
     call post(nmx-1,x1)
     call post(nmx,x2)
     j=1
     call setabx(a,b,j)
     if(b.le.a)then
       tcnewlo=s1
       pcs(1)=s1
       pcs(2)=0.d0
       pcs(3)=0.d0
       pcs(4)=0.d0
       pcs(5)=0.d0
       pcs(6)=0.d0
       pcs(7)=0.d0
       pcs(8)=0.d0
       pcs(9)=0.d0
       xmin(1)=b+(.5*(a-b))
       print*,'Single iteration required; c1= ',xmin(1)
       goto 200
     endif
c    store the value of c2 in the variable xmin(3)
c    should be between a and b
c    set x1=u to locate c2 (shouldn't matter)
     j=3
     call cmaxn3(a,b,x1,x2,xmin(5))
     c3=xmin(5)
c    n=4 unknowns for nelder-mead search, c1l(1) and c1h(2)
c    c2l(3) and c2h(4).
     n=4
     start(1)=(a+c3)/2.d0
     start(2)=(b+c3)/2.d0
     start(3)=(a+c3)/2.d0
     start(4)=(b+c3)/2.d0
     step(1)=(b-a)/2.d0
     step(2)=step(1)
     step(3)=step(1)
```

```
      step(4)=step(1)
      reqmin=.000001d0
      icount=200
      call nelmin1(n,start,xmin,xsec,tcnewlo,
     &tcsec,reqmin,step,icount,pcs)
      call clg3out(xmin,tcst,pcs)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c     this to account for lower spec by symmetry, only
c     change prior mean back before print
c     find c1 by symmetry around the specification
      if(nspc.eq.2)then
        xtmp=xmin(1)
        xmin(1)=2*u-xmin(2)
        xmin(2)=2*u-xtmp
        xtmp=xmin(3)
        xmin(3)=2*u-xmin(4)
        xmin(4)=2*u-xtmp
        xmin(5)=2*u-xmin(5)
        thta(1)=2*u-thta(1)
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      write(*,602)xmin(5),xmin(5)-bias
      write(*,604)xmin(3),xmin(3)-bias
      write(*,610)xmin(4),xmin(4)-bias
      write(*,614)xmin(1),xmin(1)-bias
      write(*,616)xmin(2),xmin(2)-bias
      write(*,618)
      write(*,620)pcs(1)
      write(*,622)pcs(2)
      write(*,624)pcs(3)
      write(*,626)pcs(4)
      write(*,628)pcs(5)
      write(*,630)pcs(6)
      write(*,632)pcs(7)
      write(*,634)pcs(8)
      write(*,636)pcs(9)
      write(*,650)tcnewlo
90    print*,'Send output to file? (Y/N)'
      read(*,500)iopt
      if(iopt.ne.'Y'.and.iopt.ne.'N'.and.iopt.ne.'y'.and.
     &iopt.ne.'n')goto 90
      if(iopt.eq.'n'.or.iopt.eq.'N')goto 200
      write(2,510)thta(1)
      write(2,520)tau(1)
      write(2,530)spec,u
      write(2,540)sme
      write(2,550)bias
      write(2,560)
      write(2,570)s1
      write(2,580)a2
      write(2,590)r2
      write(2,600)
```

```
      write(2,602)xmin(5),xmin(5)-bias
      write(2,604)xmin(3),xmin(3)-bias
      write(2,610)xmin(4),xmin(4)-bias
      write(2,614)xmin(1),xmin(1)-bias
      write(2,616)xmin(2),xmin(2)-bias
      write(2,618)
      write(2,620)pcs(1)
      write(2,622)pcs(2)
      write(2,624)pcs(3)
      write(2,626)pcs(4)
      write(2,628)pcs(5)
      write(2,630)pcs(6)
      write(2,632)pcs(7)
      write(2,634)pcs(8)
      write(2,636)pcs(9)
      write(2,650)tcnewlo
200   continue
500   format(a1)
510   format(' Prior Mean =           ',f16.8)
520   format(' Prior Std. Dev =       ',f16.8)
530   format(1x,a5,' Specification = ',f16.8)
540   format(' Measurement Error Std. Dev. =  ',f13.8)
550   format(' Measurement Error Mean (Bias) = ',f13.8)
560   format(' Input Costs:')
570   format('    Iteration (S) =     ',f12.4)
580   format('    False Accept (A) =  ',f12.4)
590   format('    False Reject (R) =  ',f12.4)
600   format(' ***************************************')
602   format(' Zero-Bias C3 =   ',f16.8,5x,'Bias Adj C3 =   ',f16.8)
604   format(' Zero-Bias C2,L = ',f16.8,5x,'Bias Adj C2,L = ',f16.8)
610   format(' Zero-Bias C2,H = ',f16.8,5x,'Bias Adj C2,H = ',f16.8)
614   format(' Zero-Bias C1,L = ',f16.8,5x,'Bias Adj C1,L = ',f16.8)
616   format(' Zero-Bias C1,H = ',f16.8,5x,'Bias Adj C1,H = ',f16.8)
618   format(' Expected Plan Costs:')
620   format('    Sampling on 1 =     ',f12.4)
622   format('    False Accept on 1 = ',f12.4)
624   format('    False Reject on 1 = ',f12.4)
626   format('    Sampling on 2 =     ',f12.4)
628   format('    False Accept on 2 = ',f12.4)
630   format('    False Reject on 2 = ',f12.4)
632   format('    Sampling on 3 =     ',f12.4)
634   format('    False Accept on 3 = ',f12.4)
636   format('    False Reject on 3 = ',f12.4)
650   format(' Expected Total Cost =  ',f12.4)
      return
      end
c*******************************************************
c     **************************************************
c     **************************************************
c     sub that performs the cluged integration of the cost
c     components between c1l and c1h (in array c1)
c
c     pcs array holds the various cost components
c        1=sampling on 1
```

```
c       2=false accept on 1
c       3=false reject on 1
c       4=sampling on 2
c       5=false accept on 2
c       6=false reject on 2   ...7-9 for iteration 3
c*****************************************************
      subroutine clg2a2(c1,tctot,pcs)
      dimension thta(4),tau(4),x(2),p(2)
      real*8 c1(2),tc4int,tc5int,tc6int,tctot,pcs(9)
      real*8 tc4pre,tc5pre,tc6pre,delx,bignum
      real*8 acc,rej,acc2,rej2,a,b,c1l,c1h,c2,x1,x1pre
      real*8 a1,b1
      real*8 alo,blo,ahi,bhi
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a1,b1
      common /ncur/ j
      common /cult/ c2
c
      print*,'Working...'
      c1l=c1(1)
      c1h=c1(2)
c
c     this sequence finds the +/- infinity limits for the
c     conditional false dispositions at n=2, based on the
c     possible extreme values of x1 ==>C1l and C1h
c
      nmx=2
      j=2
      call post(nmx,c1l)
      call setabx(alo,blo,j)
      call post(nmx,c1h)
      call setabx(ahi,bhi,j)
      a=dmin1(alo,ahi)
      b=dmax1(blo,bhi)
c
      bignum=9.99d+55
      tctot=0.d0
      tc4int=0.d0
      tc5int=0.d0
      tc6int=0.d0
      istp=0
      delx=(b1-a1)/100.d0
ccccccccccccccccccc
c
      x1=c1l
c
      j=1
      call setabx(a1,b1,j)
c
c     compare c1h, c1l to a,b from n=1 to see if they were
c     at all probable
c
      if(c1l.gt.c1h)then
         rej=100.d0
```

```
        acc=100.d0
        goto 50
      endif
      if(c1h.gt.b1)then
        rej=100.d0
       else
        flg=1.
        call qgaus(j,flg,c1h,b1,rej)
      endif
      if(c1l.lt.a1)then
        acc=100.d0
       else
        flg=2.
        call qgaus(j,flg,a1,c1l,acc)
      endif
50    continue
      pcs(1)=dble(s1)
      pcs(2)=dble(a2)*acc
      pcs(3)=dble(r2)*rej
      if(c1l.gt.c1h)goto 80
      j=2
70    continue
c     compare 2c2-x1 limits to extremes to determine if prob is
c     nonzero
      call post(nmx,x1)
      if(2.d0*c2-x1.lt.a)then
        acc2=0.d0
       else
        flg=2.
        call qgaus(j,flg,a,2.d0*c2-x1,acc2)
      endif
      if(2.d0*c2-x1.gt.b)then
        rej2=0.d0
       else
        flg=1.
        call qgaus(j,flg,2.d0*c2-x1,b,rej2)
      endif
      cost4=s1
      costa5=acc2*a2
      costr6=rej2*r2
c*****************************************************
      if (istp.eq.0)then
        tc4pre=cost4
        tc5pre=costa5
        tc6pre=costr6
        x1pre=x1
        istp=1
          else
c     marginal on x1 is N(theta1,sme^2+tau1^2)
        x(1)=(x1pre-thta(1))/sqrt(tau(1)**2.+sme**2.)
        x(2)=(x1-thta(1))/sqrt(tau(1)**2.+sme**2.)
        call normal(x,p)
        prob=p(2)-p(1)
        tc4int=tc4int+prob*((cost4+tc4pre)/2.d0)
```

```
      tc5int=tc5int+prob*((costa5+tc5pre)/2.d0)
      tc6int=tc6int+prob*((costr6+tc6pre)/2.d0)
      tc4pre=cost4
      tc5pre=costa5
      tc6pre=costr6
      x1pre=x1
    endif
c*****************************************************
    x1=x1+delx
c   this next chunk of code attempts to account for the
c   slack between the last x1 and c1h
    if (x1.gt.c1h)then
     tc4int=tc4int+((c1h-x1pre)/delx)*prob*((cost4+tc4pre)/2.d0)
     tc5int=tc5int+((c1h-x1pre)/delx)*prob*((costa5+tc5pre)/2.d0)
     tc6int=tc6int+((c1h-x1pre)/delx)*prob*((costr6+tc6pre)/2.d0)
      goto 80
    endif
    goto 70
80  continue
    tctot=pcs(1)+pcs(2)+pcs(3)+tc4int+tc5int+tc6int
    pcs(4)=tc4int
    pcs(5)=tc5int
    pcs(6)=tc6int
200 continue
    return
    end
c******************************************************
c   ************************************************
c   ************************************************
c   sub that performs the cluged integration of the cost
c   components between c2l and c2h
c
c   x1 is the 1st observation value at which the
c   integration is performed
c   tc7-9int are the cost components for the third
c   iteration passed back to the outer intgral loop
c   (integrating between c1l and c1h).
c******************************************************
    subroutine clgin(c2l,c2h,x1,tc7int,tc8int,tc9int)
    dimension thta(4),tau(4),x(2),p(2)
    real*8 x1,tctot,delx2,c2l,c2h
    real*8 tc7int,tc8int,tc9int,tc7pre,tc8pre,tc9pre
    real*8 acc3,rej3,alst,blst,c3,x2,x2pre,a,b
    common /costs/ s1,a2,r2
    common /parms/ tau,sme,thta,u
    common /inf2/ alst,blst
    real*8 alo,blo,ahi,bhi
    common /ncur/ j
    common /cult/ c3
c
    print*,'    Inner Integral Loop'
    tctot=0.d0
    tc7int=0.d0
    tc8int=0.d0
```

```
      tc9int=0.d0
      istp=0
      delx2=(blst-alst)/100.d0
c
c     this sequence finds the +/- infinity limits for the
c     conditional false dispositions at n=2, based on the
c     possible extreme values of x1 ==>C1l and C1h
c
      nmx=3
      call post(nmx,2.d0*c2l-x1)
      call setabx(alo,blo,nmx)
      call post(nmx,2.d0*c2h-x1)
      call setabx(ahi,bhi,nmx)
      a=dmin1(alo,ahi)
      b=dmax1(blo,bhi)
cccccccccccccccccc
      x2=2.d0*c2l-x1
50    continue
      j=3
70    continue
      call post(nmx,x2)
ccccccccc      call setabx(a,b,j)
      x3lim=3.d0*c3-x1-x2
      if(x3lim.lt.a)then
         acc3= 0.d0
        else
         flg=2.
         call qgaus(j,flg,a,3.d0*c3-x1-x2,acc3)
      endif
      if(x3lim.gt.b)then
         rej3= 0.d0
        else
         flg=1.
         call qgaus(j,flg,3.d0*c3-x1-x2,b,rej3)
      endif
      cost7=s1
      costa8=acc3*a2
      costr9=rej3*r2
c*****************************************************
      if (istp.eq.0)then
         tc7pre=cost7
         tc8pre=costa8
         tc9pre=costr9
         x2pre=x2
         istp=1
           else
c     marginal on x2|x1 is N(theta2,sme^2+tau2^2)
         x(1)=(x2pre-thta(2))/sqrt(tau(2)**2.+sme**2.)
         x(2)=(x2-thta(2))/sqrt(tau(2)**2.+sme**2.)
         call normal(x,p)
         prob=p(2)-p(1)
         tc7int=tc7int+prob*((cost7+tc7pre)/2.d0)
         tc8int=tc8int+prob*((costa8+tc8pre)/2.d0)
         tc9int=tc9int+prob*((costr9+tc9pre)/2.d0)
```

```
        tc7pre=cost7
        tc8pre=costa8
        tc9pre=costr9
        x2pre=x2
      endif
c*****************************************************
      x2=x2+delx2
c     this next chunk of code accounts for the
c     slack between the last x1 and c1h
      if (x2.gt.2.*c2h-x1)then
       tc7int=tc7int+((2.*c2h-x1-x2pre)/delx2)*prob*
     &      ((cost7+tc7pre)/2.d0)
       tc8int=tc8int+((2.*c2h-x1-x2pre)/delx2)*prob*
     &      ((costa8+tc8pre)/2.d0)
       tc9int=tc9int+((2.*c2h-x1-x2pre)/delx2)*prob*
     &      ((costr9+tc9pre)/2.d0)
        goto 80
      endif
      goto 70
80    continue
      return
      end
c*****************************************************
c     ************************************************
c     ************************************************
c     sub that performs the cluged integration of the cost
c     components between c1l and c1h (in array c)
c     array c also contains current values of c2l and c2h
c
c     cost components carried in array pcs
c        1=sampling on 1
c        2=false accept on 1
c        3=false reject on 1
c        4=sampling on 2
c        5=false accept on 2
c        6=false reject on 2   ...7-9 for iteration 3
c*****************************************************
      subroutine clg3out(c,tctot,pcs)
      dimension thta(4),tau(4),x(2),p(2)
      real*8 c(4),x1,tctot,delx1,c1l,c1h,x1pre,c2l,c2h
      real*8 tc4pre,tc5pre,tc6pre,tc4int,tc5int,tc6int
      real*8 tc7int,tc8int,tc9int,tc7pre,tc8pre,tc9pre
      real*8 acc,rej,a,b,c3,bignum,acc2,rej2,tc7,tc8,tc9,pcs(9)
      real*8 alo,blo,ahi,bhi
      real*8 a1,b1
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a1,b1
      common /inf2/ a,b
      common /ncur/ j
      common /cult/ c3
c
      print*,'Outer Integral Loop'
      bignum=9.99d+55
      tctot=0.d0
```

```
      costl=0.
      costa2=0.
      costr3=0.
      tc4int=0.d0
      tc5int=0.d0
      tc6int=0.d0
      tc7int=0.d0
      tc8int=0.d0
      tc9int=0.d0
c
      c1l=c(1)
      c1h=c(2)
      c2l=c(3)
      c2h=c(4)
c
c     sequence that checks if tested cutoff values are beyond
c     the "infinity" limits that have been found at n=1
      if(c1h.gt.b1)then
        tctot=bignum
        goto 100
      endif
      if(c1l.lt.a1)then
         tctot=bignum
         goto 100
      endif
      if(c2l.lt.c1l)then
         tctot=bignum
         goto 100
      endif
      if(c2h.gt.c1h)then
         tctot=bignum
         goto 100
      endif
      if(c2l.gt.c3)then
         tctot=bignum
         goto 100
      endif
      if(c2h.lt.c3)then
         tctot=bignum
         goto 100
      endif
c
      istp=0
c     delx1 is the step size of integration
      delx1=(c1h-c1l)/100.d0
c
c     this sequence finds the +/- infinity limits for the
c     conditional false dispositions at n=2, based on the
c     possible extreme values of x1 ==>C1l and C1h
c
      j=2
      call post(j,c1l)
      call setabx(alo,blo,j)
      call post(j,c1h)
```

```
      call setabx(ahi,bhi,j)
      a=dmin1(alo,ahi)
      b=dmax1(blo,bhi)
ccccccccccccccccccc
      x1=c1l
      j=1
        flg=1.
        call qgaus(j,flg,c1h,b1,rej)
        flg=2.
        call qgaus(j,flg,a1,c1l,acc)
50    continue
      cost1=s1
      costa2=a2*acc
      costr3=r2*rej
70    continue
      j=2
      call post(j,x1)
c
      if(2.d0*c2l-x1.lt.a)then
        acc2=0.d0
       else
        flg=2.
        call qgaus(j,flg,a,2.d0*c2l-x1,acc2)
      endif
      if(2.d0*c2h-x1.gt.b)then
        rej2=0.d0
       else
        flg=1.
        call qgaus(j,flg,2.d0*c2h-x1,b,rej2)
      endif
      cost4=s1
      costa5=a2*acc2
      costr6=r2*rej2
      call clgin(c2l,c2h,x1,tc7,tc8,tc9)
c***************************************************
      if (istp.eq.0)then
        tc4pre=dble(cost4)
        tc5pre=dble(costa5)
        tc6pre=dble(costr6)
        tc7pre=tc7
        tc8pre=tc8
        tc9pre=tc9
        x1pre=x1
        istp=1
          else
c     marginal on x1 is N(theta1,sme^2+tau1^2)
        x(1)=(x1pre-thta(1))/sqrt(tau(1)**2.+sme**2.)
        x(2)=(x1-thta(1))/sqrt(tau(1)**2.+sme**2.)
        call normal(x,p)
        prob=p(2)-p(1)
        tc4int=tc4int+prob*((dble(cost4)+tc4pre)/2.d0)
        tc5int=tc5int+prob*((dble(costa5)+tc5pre)/2.d0)
        tc6int=tc6int+prob*((dble(costr6)+tc6pre)/2.d0)
        tc7int=tc7int+prob*((tc7+tc7pre)/2.d0)
```

```
      tc8int=tc8int+prob*((tc8+tc8pre)/2.d0)
      tc9int=tc9int+prob*((tc9+tc9pre)/2.d0)
      tc4pre=dble(cost4)
      tc5pre=dble(costa5)
      tc6pre=dble(costr6)
      tc7pre=tc7
      tc8pre=tc8
      tc9pre=tc9
      x1pre=x1
    endif
c**************************************************
    x1=x1+delx1
c   this next chunk of code accounts for the
c   slack between the last x1 and c1h
    if (x1.gt.c1h)then
     tc4int=tc4int+((c1h-x1pre)/delx1)*prob*
   &      ((dble(cost4)+tc4pre)/2.d0)
     tc5int=tc5int+((c1h-x1pre)/delx1)*prob*
   &      ((dble(costa5)+tc5pre)/2.d0)
     tc6int=tc6int+((c1h-x1pre)/delx1)*prob*
   &      ((dble(costr6)+tc6pre)/2.d0)
     tc7int=tc7int+((c1h-x1pre)/delx1)*prob*
   &      ((tc7+tc7pre)/2.d0)
     tc8int=tc8int+((c1h-x1pre)/delx1)*prob*
   &      ((tc8+tc8pre)/2.d0)
     tc9int=tc9int+((c1h-x1pre)/delx1)*prob*
   &      ((tc9+tc9pre)/2.d0)
      goto 80
    endif
    goto 70
80  continue
    tctot=dble(cost1+costa2+costr3)+tc4int+tc5int+tc6int+
   &tc7int+tc8int+tc9int
    pcs(1)=dble(cost1)
    pcs(2)=dble(costa2)
    pcs(3)=dble(costr3)
    pcs(4)=tc4int
    pcs(5)=tc5int
    pcs(6)=tc6int
    pcs(7)=tc7int
    pcs(8)=tc8int
    pcs(9)=tc9int
100  continue
    return
    end
c**************************************************
c   ************************************************
c   ************************************************
c   sub to calculate c1
c   by step search (a and b are +/- infinity)
c   conv is convergence check size
c**************************************************
    subroutine cmaxn1(a,b,c)
c
```

```
      real*8 a,b,c,ratchk
      common /costs/ s1,a2,r2
      c=a
      cpre=c
      conv=.00001
c     del is step size from a to b
c     rat is the cost indifference ratio
      del=(b-a)/100.d0
      rat=r2/(r2+a2)
      flg=2.
      n=1
c     pcmu evaluates the function at c
c     returns ratio ratchk to compare to rat
c     sgn used to see if rat has been overstepped
c     then reduce del and come at it again
c
      call pcmu(c,flg,ratchk)
      delchk=rat-ratchk
      print*
      sgn=delchk/abs(delchk)
      if(abs(delchk).le.conv)goto 200
      c=c+del
20    continue
c
      call pcmu(c,flg,ratchk)
      delchk=(rat-ratchk)
      if (abs(delchk).gt.conv)then
         if(delchk/abs(delchk).ne.sgn)then
            del=del/10.
            c=cpre+del
            goto 20
         endif
         cpre=c
         c=c+del
         goto 20
      endif
c
200   continue
      return
      end
c**************************************************
c     ************************************************
c     ************************************************
c     sub to calculate c2
c     by step search (a and b are +/- infinity)
c     function evaluated at observation x1
c     conv is convergence check size
c**************************************************
      subroutine cmaxn2(a,b,x1,c)
c
      real*8 a,b,c,ratchk,x1
      common /costs/ s1,a2,r2
      common /ncur/ j
      c=a
```

```
      cpre=c
      conv=.000010
c     del is step size from a to b
c     rat is the cost indifference ratio
      del=(b-a)/100.d0
      rat=r2/(r2+a2)
      flg=2.
      n=1
c     pcmu evaluates the function at c
c     returns ratio ratchk to compare to rat
c     sgn used to see if rat has been overstepped
c     then reduce del and come at it again
c
      call pcmu(2*c-x1,flg,ratchk)
      delchk=rat-ratchk
      sgn=delchk/abs(delchk)
      if(abs(delchk).le.conv)goto 200
      c=c+del
20    continue
      call pcmu(2*c-x1,flg,ratchk)
      delchk=(rat-ratchk)
      if (abs(delchk).gt.conv)then
        if(delchk/abs(delchk).ne.sgn)then
          del=del/10
          c=cpre+del
          goto 20
        endif
        cpre=c
        c=c+del
        goto 20
      endif
200   continue
      return
      end
c*****************************************************
c     ************************************************
c     ************************************************
c     subroutine to calculate c3
c     by step search (a and b are +/- infinity)
c     function is evaluated at x1 and x2 (observations)
c     conv is convergence check size
c*****************************************************
c
      subroutine cmaxn3(a,b,x1,x2,c)
c
      real*8 a,b,c,ratchk,x1,x2
      common /costs/ s1,a2,r2
      common /ncur/ j
      c=a
      cpre=c
      conv=.000010
c     del is step size from a to b
c     rat is the cost indifference ratio
      del=(b-a)/100.d0
```

```
      rat=r2/(r2+a2)
      flg=2.
      n=1
c     pcmu evaluates the function at c
c     returns ratio ratchk to compare to rat
c     sgn used to see if rat has been overstepped
c     then reduce del and come at it again
c
      call pcmu(3*c-x1-x2,flg,ratchk)
      delchk=rat-ratchk
      sgn=delchk/abs(delchk)
      if(abs(delchk).le.conv)goto 200
      c=c+del
20    continue
      call pcmu(3*c-x1-x2,flg,ratchk)
      delchk=(rat-ratchk)
      if (abs(delchk).gt.conv)then
         if(delchk/abs(delchk).ne.sgn)then
            del=del/10
            c=cpre+del
            goto 20
         endif
         cpre=c
         c=c+del
         goto 20
      endif
200   continue
      return
      end
c****************************************************
c     ************************************************
c     ************************************************
      subroutine normal(x,p)
c
c     algorithm  as 2 j.r.statist.soc. c,(1968) v.17,no.2
c     by B. E. Cooper
c
c     computes normal areas and ordinates for an array of x values
c****************************************************
c
      dimension x(2),p(2),q(2),z(2)
      dimension a(5)
c
      dimension connor(17)
      data connor
     1/ 8.0327350124e-17,  1.4483264644e-15,  2.4558270103e-14,
     2  3.9554295164e-13,  5.9477940136e-12,  8.3507027951e-11,
     3  1.0892221037e-9,   1.3122532964e-8,   1.4503852223e-7,
     4  1.4589169001e-6,   1.3227513228e-5,   1.0683760684e-4,
     5  7.5757575758e-4,   4.6296296296e-3,   2.380952381e-2,  0.1,
     6  3.3333333333e-1/
c
      data rrt2pi /0.3989422804/
c
```

```
      n=2
c
      ifault=0
      if(n) 1,1,2
    1 ifault=1
      go to 100
    2 do 31 i=1,n
      s=x(i)
      y=s*s
      if (s) 10,11,12
   11 z(i)=rrt2pi
      p(i)=0.5
      q(I)=0.5
      goto 31
c
c
c
c     print*,'series approximation'
   10 s=-s
   12 z(i)=rrt2pi*exp(-.5*y)
      if (s-2.5)13,14,14
   13 y=-.5*y
      p(i)=connor(1)
      do 15 l=2,17
   15 p(i)=p(i)*y+connor(l)
      p(i)=(p(i)*y+1.0)*x(i)*rrt2pi+0.5
      q(i)=1.0-p(i)
      goto 31
c
c
c
   14 continue
      a(2)=1.0
      a(5)=1.0
      a(3)=1.0
      y=1.0/y
      a(4)=1.0+y
      r=2.0
   19 do 17 l=1,3,2
      do 18 j=1,2
      k=l+j
      ka=7-k
      a(k)=a(ka)+a(k)*r*y
18    continue
      r=r+1.0
   17 continue
      atst=(a(2)/a(3))-(a(5)/a(4))
      if(abs(atst).gt.(.000001))goto 19
   20 p(i)=(a(5)/a(4))*z(i)/x(i)
      if(x(i))21,11,22
   21 p(i)=-p(i)
      q(i)=1.0-p(i)
      goto 31
   22 q(i)=p(i)
```

```
      p(i)=1.0-p(i)
   31 continue
  100 continue
      return
      end
c*****************************************************
c    ************************************************
c    ************************************************
c    called from nelmin
c    takes care of the accounting in solving
c    each leg of the tc equation (using qgaus) with
c    estimates provided by nelmin.
c    should receive the array of unknowns, must pass a
c    function value back to nelmin
c
c    calls sub errint which evaluates the erf function
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine pcmu(pcarg,flg,from)
      real*8 pcarg,from,a,b
      real*8 erfarg,erf,erfc,erfrat
      dimension tau(4),thta(4)
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a,b
      common /ncur/ j
      erfrat=((pcarg/sme**2.)+(thta(j)/tau(j)**2.))/
     &(((1./sme**2.)+(1./tau(j)**2.d0))*tau(j+1))
      erfarg=.7071067811865475244d0/tau(j+1)*u-
     &.7071067811865475244d0*erfrat
      call errint(erfarg,erf,erfc)
      if (flg.eq.2.)then
         from=.5d0*(1.d0-erf)
        else
         from=.5d0*(erf+1.d0)
      endif
      return
      end
c*****************************************************
c    ************************************************
c    ************************************************
c    sub to calculate the posterior mean
c*****************************************************
      subroutine post(ix,x1)
      real*8 x1
      dimension thta(4),tau(4)
      common /parms/ tau,sme,thta,u,a,b
      thta(ix)=(x1/sme**2.d0+thta(ix-1)/tau(ix-1)**2.d0)/
     &(1./sme**2.d0+1./tau(ix-1)**2.d0)
40    continue
      return
      end
c*****************************************************
c    ************************************************
c    ************************************************
```

```
c    ten pt gaussian quadrature
c    taken from Press, Flannery, Teukolsky and Vetterling
c    (1986), Numerical Recipes, Cambridge Univ. Press, NY.
c***************************************************
c
c
      subroutine qgaus(j,flg,a,b,ss)
      real*8 x(5),w(5),ss,xm,xr,dx,t1,t2,a,b
      data x/.1488743389d0,.4333953941d0,.6794095682d0,
     &.8650633666d0,.9739065285d0/
      data w/.2955242247d0,.2692667193d0,.2190863625d0,
     &.1494513491d0,.0666713443d0/
      xm=0.5d0*(b+a)
      xr=0.5d0*(b-a)
      ss=0.d0
      do 11 k=1,5
        dx=xr*x(k)
c
c    the 2nd arg is simply the flag for error type
c    for the cost probability component
c
        call func(j,flg,xm+dx,t1)
        call func(j,flg,xm-dx,t2)
        ss=ss+w(k)*(t1+t2)
c
c
 11   continue
      ss=xr*ss
      return
      end
c***************************************************
c    ***********************************************
c    ***********************************************
c    subroutine to set a (-infinity) and b (infinity)
c    practical limits for the function used in order
c    to utilize gaussian quadrature
c    arg j defines which n (not nmax) is current
c    j passes from management program
c    stores high value in pk, low value in trof
c    checks for trof as a fraction of pk to quit
c***************************************************
      subroutine setabx(a,b,j)
      real*8 x,ffunc,pk,trof,a,b,lim1,lim2
      dimension thta(4),tau(4)
      common /parms/ tau,sme,thta,u
c    reject probability
c    must set b (upper), unknown involving c is lower limit
c    start at u, work up
c    lims 1 and 2 account for the sign of the spec and
c    cutoff values
c
      lim1=0.d0
      lim2=2.*u
      if(u/abs(u).lt.0.d0)then
```

```
      lim1=2.*u
      lim2=0.d0
     endif
     sgn=1.
     irev=0
8    continue
     flg=1.
     x=dble(u)
     call func(j,flg,x,ffunc)
9    continue
     pk=ffunc
     trof=ffunc
     if(ffunc.eq.0.d0)then
       x=x-sgn*dble(sme)
       goto 12
     endif
     x=x+sgn*dble(sme)
10   continue
     call func(j,flg,x,ffunc)
     if(ffunc.gt.pk)then
       pk=ffunc
     endif
     if(ffunc.lt.trof)then
       trof=ffunc
       b=x
       if(trof.le.(.000001d0*pk))then
         goto 20
       endif
     endif
     x=x+sgn*dble(sme)
     goto 10
12   continue
     call func(j,flg,x,ffunc)
     if(ffunc.gt.0.d0)then
       b=x
       if(irev.eq.0)goto 20
       goto 9
      else
       if(irev.eq.0.and.x.lt.lim1)then
         irev=1
         sgn=-1.*sgn
         goto 8
       endif
       if(irev.eq.1.and.x.gt.lim2)then
         b=lim1
         goto 20
       endif
       x=x-sgn*dble(sme)
       goto 12
     endif
20   continue
     irev=0
     sgn=.5
22   continue
```

```
c    accept probability
c    need a, unknown involving c is upper limit
c    start at u, work way down
     flg=2.
     x=dble(u)
27   continue
     call func(j,flg,x,ffunc)
28   continue
     pk=ffunc
     trof=ffunc
     if(ffunc.eq.0.d0)then
        x=x+sgn*dble(sme)
        goto 32
     endif
     x=x-sgn*dble(sme)
30   continue
     call func(j,flg,x,ffunc)
     if(ffunc.gt.pk)then
        pk=ffunc
     endif
     if(ffunc.lt.trof)then
        trof=ffunc
        a=x
        if(trof.le.(.000001d0*pk))then
           goto 40
        endif
     endif
     x=x-sgn*dble(sme)
     goto 30
32   continue
     call func(j,flg,x,ffunc)
     if(ffunc.gt.0.d0)then
        a=x
        if(irev.eq.0)goto 40
        goto 28
      else
        if(irev.eq.0.and.x.gt.lim2)then
           irev=1
           sgn=-1.*sgn
           goto 22
        endif
        if(irev.eq.1.and.x.lt.lim1)then
           a=lim2
           goto 40
        endif
        x=x+sgn*dble(sme)
        goto 32
     endif
40   continue
     return
     end
c**************************************************
c    ************************************************
c    ************************************************
```

```
c     this sub created from the necessity of covering
c     all possible values of cutoffs which a user may
c     give in requesting an expected value of his/her
c     inspection program.Must known all limits of the
c     conditional false disposition curves.
c     j is observation number from management program
c     stores high value in pk, low value in trof
c     checks for trof as a fraction of pk to quit
c*************************************************
      subroutine setmor(ap,bp,j)
      real*8 x,ffunc,pk,trof,ap,bp,lim1,lim2
      dimension thta(4),tau(4)
      common /parms/ tau,sme,thta,u
c     type i error
c     must set bp, looking for the curve's lower limit
c     start at u, work down (to start)
c
c     irev is the flag which tells if I have to look in
c     both directions
c     lims 1 and 2 account for the sign of the spec and
c     cutoff values
      lim1=0.d0
      lim2=2.*u
      if(u/abs(u).lt.0.d0)then
       lim1=2.*u
       lim2=0.d0
      endif
      irev=0
      sgn=.5
5      continue
      flg=1.
      x=dble(u)
      call func(j,flg,x,ffunc)
8      continue
      pk=ffunc
      trof=ffunc
      if(ffunc.eq.0.d0)then
         x=x+sgn*dble(sme)
         goto 12
      endif
      x=x-sgn*dble(sme)
10      continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.pk)then
         pk=ffunc
      endif
      if(ffunc.lt.trof)then
         trof=ffunc
         bp=x
         if(trof.le.(.0001d0*pk))then
           goto 20
         endif
      endif
      x=x-sgn*dble(sme)
```

```
      goto 10
12    continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.0.d0)then
        bp=x
        if(irev.eq.0)goto 20
        goto 8
       else
        if(irev.eq.0.and.x.gt.lim2)then
            irev=1
            sgn=-1.*sgn
            goto 5
        endif
        if(irev.eq.1.and.x.lt.lim1)then
            bp=lim2
            goto 20
        endif
        x=x+sgn*dble(sme)
        goto 12
      endif
20    continue
      irev=0
      sgn=.5
25    continue
c     type ii
c     need ap, looking for upper limit
c     start at u, work way up (to start)
      flg=2.
      x=dble(u)
27    continue
      call func(j,flg,x,ffunc)
28    continue
      pk=ffunc
      trof=ffunc
      if(ffunc.eq.0.d0)then
        x=x-sgn*dble(sme)
        goto 32
      endif
      x=x+sgn*dble(sme)
30    continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.pk)then
        pk=ffunc
      endif
      if(ffunc.lt.trof)then
        trof=ffunc
        ap=x
        if(trof.le.(.0001d0*pk))then
          goto 40
        endif
      endif
      x=x+sgn*dble(sme)
      goto 30
32    continue
```

```
      call func(j,flg,x,ffunc)
      if(ffunc.gt.0.d0)then
        ap=x
        if(irev.eq.0)goto 40
        goto 28
       else
        if(irev.eq.0.and.x.lt.lim1)then
            irev=1
            sgn=-1.*sgn
            goto 25
        endif
        if(irev.eq.1.and.x.gt.lim2)then
            ap=lim1
            goto 40
        endif
        x=x-sgn*dble(sme)
        goto 32
      endif
40    continue
      return
      end
c****************************************************
c     **************************************************
c     **************************************************
c     input subroutine for all plan parameters
c     passes back nspc as the flg for upper/lower limit
c     u=limit
c     thta=prior mean
c     tau=prior std.dev.
c     bias=measurement error dist mean
c     sme=measurement error dist std.dev.
c     s1=sampling cost
c     a2=false acceptance cost
c     r2=false rejection cost
c
c****************************************************
      subroutine vardef(u,thta,tau,bias,sme,s1,a2,r2,nspc)
cccccccccccccccccccc
      character iopt
      character*5 spec
      spec='Upper'
      iflg=0
10    print*
      print*,'Enter the Specification Limit.'
      print*
      read(*,*,err=10)u
      if(iflg.eq.1)goto 90
15    print*
      print*,'Is this an Upper (1) or Lower (2) Spec?'
      print*,'Enter 1 or 2.'
      print*
      read(*,*,err=15)nspc
      if(nspc.ne.1.and.nspc.ne.2)then
        write(*,*)'**** Invalid Entry.  Please Reenter. ****'
```

```
      goto 15
    endif
    if(nspc.eq.2)spec='Lower'
20  print*
    print*,'Enter the value of the prior distribution mean.'
    print*
    read(*,*,err=20)thta
    if(iflg.eq.1)goto 90
30  print*
    print*,'Enter the value of the prior standard deviation.'
    print*
    read(*,*,err=30)tau
    if(tau.le.0)then
      print*
      print*,'**** Standard Deviation must be positive ****'
      goto 30
    endif
    if(iflg.eq.1)goto 90
40  print*
    print*,'Enter the value of the measurement error'
    print*,'distribution mean (bias).'
    print*,'Sign Convention: If the instrument reads higher '
    print*,'than the true value, this bias should be positive.'
    print*
    read(*,*,err=40)bias
    if(iflg.eq.1)goto 90
50  print*
    print*,'Enter the value of the measurement error'
    print*,'distribution standard deviation.'
    print*
    read(*,*,err=50)sme
    if(sme.le.0)then
      print*
      print*,'**** Standard Deviation must be positive ****'
      goto 50
    endif
    if(iflg.eq.1)goto 90
60  print*
    print*,'Enter the cost associated with a single measurement'
    print*,'iteration (S).'
    print*
    read(*,*,err=60)s1
    if(s1.lt.0)then
      print*
      print*,'**** Cost must be positive ****'
      goto 60
    endif
    if(iflg.eq.1)goto 90
70  print*
    print*,'Enter the cost associated with a false acceptance of'
    print*,'a batch of product (A).'
    print*
    read(*,*,err=70)a2
    if(a2.lt.0)then
```

```
      print*
      print*,'**** Cost must be positive ****'
      goto 70
    endif
    if(iflg.eq.1)goto 90
80  print*
    print*,'Enter the cost associated with a false rejection of'
    print*,'a batch of product (R).'
    print*
    read(*,*,err=80)r2
    if(r2.lt.0)then
      print*
      print*,'**** Cost must be positive ****'
      goto 80
    endif
90  print*
    write(*,300)spec,u
    write(*,310)thta
    write(*,320)tau
    write(*,330)bias
    write(*,340)sme
    write(*,350)s1
    write(*,360)a2
    write(*,370)r2
    print*
    print*,'Is the above information correct?'
    print*,'Enter to accept or parameter # to reenter.'
    print*
    read(*,380)iopt
    print*
    if(iopt.eq.' ')goto 1000
    if(iopt.lt.'1'.or.iopt.gt.'8')then
      print*,'**** Invalid Entry.  Please Reenter. ****'
      goto 90
    endif
    iflg=1
    if(iopt.eq.'1')then
      goto 10
    endif
    if(iopt.eq.'2')then
      goto 20
    endif
    if(iopt.eq.'3')then
      goto 30
    endif
    if(iopt.eq.'4')then
      goto 40
    endif
    if(iopt.eq.'5')then
      goto 50
    endif
    if(iopt.eq.'6')then
      goto 60
    endif
```

```
      if(iopt.eq.'7')then
        goto 70
      endif
      if(iopt.eq.'8')then
        goto 80
      endif
300   format(1x,'1  ',a5' Specification Limit=      ',f10.4)
310   format(' 2  Prior Distribution Mean=        ',f10.4)
320   format(' 3  Prior Standard Deviation=       ',f10.4)
330   format(' 4  Error Distribution Mean (Bias)= ',f10.4)
340   format(' 5  Error Distribution Std. Dev.=   ',f10.4)
350   format(' 6  Iteration Cost (S)=             ',f8.2)
360   format(' 7  False Acceptance Cost (A)=      ',f8.2)
370   format(' 8  False Rejection Cost (R)=       ',f8.2)
380   format(a1)
1000  continue
      return
      end
c*****************************************************
c     ***********************************************
c     ***********************************************
c     error function evaluation (erf)
c     modified from Stegun and Zucker
c*****************************************************
c
      subroutine errint (x,erf,erfc)
      real*8 an,bn,cons,c1,dn,erf,erfc,f,fn,fnm1,
     1 fnm2,four,gn,gnm1,gnm2,one,prev,pwr,rnbc,scf,sum,
     2 tn,toler,trrtpi,two,ulcf,ulps,wn,x,y,ysq
c
      data nbc,nbm/11,60/
      data one,two,four,ulps,cons/1.d0,2.d0,4.d0,1.d0,.83d0/
      data trrtpi/1.128379167095512574d0/
c
      rnbc=nbc
      toler=two**(-nbm)
c
c  test on zero
c
      if(x) 2,1,2
  1   erf=0.d0
      erfc=one
c
      return
c
  2   y=dabs(x)
      ysq=y**2.d0
      if(y-ulps) 3,3,4
c
c   maximum argument
c
  4   c1=two**((rnbc-one)/two)
      ulcf=cons*c1
c
```

```
      if(iopt.eq.'7')then
         goto 70
      endif
      if(iopt.eq.'8')then
         goto 80
      endif
300   format(1x,'1  ',a5' Specification Limit=      ',f10.4)
310   format(' 2  Prior Distribution Mean=        ',f10.4)
320   format(' 3  Prior Standard Deviation=       ',f10.4)
330   format(' 4  Error Distribution Mean (Bias)= ',f10.4)
340   format(' 5  Error Distribution Std. Dev.=   ',f10.4)
350   format(' 6  Iteration Cost (S)=             ',f8.2)
360   format(' 7  False Acceptance Cost (A)=      ',f8.2)
370   format(' 8  False Rejection Cost (R)=       ',f8.2)
380   format(a1)
1000  continue
      return
      end
c*****************************************************
c     ************************************************
c     ************************************************
c     error function evaluation (erf)
c     modified from Stegun and Zucker
c*****************************************************
c
      subroutine errint (x,erf,erfc)
      real*8 an,bn,cons,c1,dn,erf,erfc,f,fn,fnm1,
     1 fnm2,four,gn,gnm1,gnm2,one,prev,pwr,rnbc,scf,sum,
     2 tn,toler,trrtpi,two,ulcf,ulps,wn,x,y,ysq
c
      data nbc,nbm/11,60/
      data one,two,four,ulps,cons/1.d0,2.d0,4.d0,1.d0,.83d0/
      data trrtpi/1.128379167095512574d0/
c
      rnbc=nbc
      toler=two**(-nbm)
c
c test on zero
c
      if(x) 2,1,2
 1    erf=0.d0
      erfc=one
c
      return
c
 2    y=dabs(x)
      ysq=y**2.d0
      if(y-ulps) 3,3,4
c
c  maximum argument
c
 4    c1=two**((rnbc-one)/two)
      ulcf=cons*c1
c
```

```
      if(iopt.eq.'7')then
        goto 70
      endif
      if(iopt.eq.'8')then
        goto 80
      endif
300   format(1x,'1  ',a5' Specification Limit=      ',f10.4)
310   format(' 2  Prior Distribution Mean=        ',f10.4)
320   format(' 3  Prior Standard Deviation=       ',f10.4)
330   format(' 4  Error Distribution Mean (Bias)= ',f10.4)
340   format(' 5  Error Distribution Std. Dev.=   ',f10.4)
350   format(' 6  Iteration Cost (S)=             ',f8.2)
360   format(' 7  False Acceptance Cost (A)=      ',f8.2)
370   format(' 8  False Rejection Cost (R)=       ',f8.2)
380   format(a1)
1000  continue
      return
      end
c*****************************************************
c     ************************************************
c     ************************************************
c     error function evaluation (erf)
c     modified from Stegun and Zucker
c*****************************************************
c
      subroutine errint (x,erf,erfc)
      real*8 an,bn,cons,c1,dn,erf,erfc,f,fn,fnm1,
     1 fnm2,four,gn,gnm1,gnm2,one,prev,pwr,rnbc,scf,sum,
     2 tn,toler,trrtpi,two,ulcf,ulps,wn,x,y,ysq
c
      data nbc,nbm/11,60/
      data one,two,four,ulps,cons/1.d0,2.d0,4.d0,1.d0,.83d0/
      data trrtpi/1.128379167095512574d0/
c
      rnbc=nbc
      toler=two**(-nbm)
c
c test on zero
c
      if(x) 2,1,2
 1    erf=0.d0
      erfc=one
c
      return
c
 2    y=dabs(x)
      ysq=y**2.d0
      if(y-ulps) 3,3,4
c
c  maximum argument
c
 4    c1=two**((rnbc-one)/two)
      ulcf=cons*c1
c
```

```
c**********************************************************
c   modified from:
c   Olsson, D. M., "A Sequential Simplex Program for
c   Solving Minimization Problems," JQT, V. 6, No. 1,
c   pp. 53-57, Jan. 1974.
c   and from
c   Ho, C., "The Economic Design and Evaluation of Three
c   Variables Control Charts", Ph.D. Dissertation, O.S.U
c   July, 1992.
c**********************************************************
      real*8 start(n),step(n),xmin(n),xsec(n),ynewlo,
     &ysec,reqmin,p(20,21),pstar(20),p2star(20),
     &pbar(20),y(20),z,ylo,rcoeff,ystar,ecoeff,
     &y2star,ccoeff,f,dabit,dchk,coord1,coord2,pclo(9),pcs(9)
      data rcoeff/1.0d0/,ecoeff/2.0d0/,ccoeff/0.5d0/
      kcount=icount
      icount=0
c**********************************************************
c   initialization
c**********************************************************
      do 60 i=1,n
        xmin(i)=0.0d0
        xsec(i)=0.0d0
   60 continue
      ynewlo=0.0d0
      ysec=0.0d0
      if (reqmin.le.0.0d0) icount=icount-1
      if (n.le.0) icount=icount-10
      if (n.gt.20) icount=icount-10
      if (icount.lt.0)then
         print*,'iterations expired'
         return
      endif
      dabit=2.04607d-35
      bignum=1.0d30
      konvge=5
      xn=float(n)
      nn=n+1
c**********************************************************
c   construction of simplex
c**********************************************************
 1001 do 1 i=1,n
    1 p(i,nn)=start(i)
c
      if(n.eq.2)call clg2a2(start,f,pcs)
      if(n.eq.4)call clg3out(start,f,pcs)
c      print*,'nm f= ',f
      y(nn)=f
      icount=icount+1
      do 2 j=1,n
        dchk=start(j)
        start(j)=dchk+step(j)
        do 3 i=1,n
    3   p(i,j)=start(i)
```

```
c
      if(n.eq.2)call clg2a2(start,f,pcs)
      if(n.eq.4)call clg3out(start,f,pcs)
c       print*,'nm f= ',f
      y(j)=f
      icount=icount+1
    2 start(j)=dchk
c***********************************************************
c     simplex construction complete
c***********************************************************
c     find highest and lowest y value
c     ynewlo indicates the vertex of the
c     simplex to be replaced
c***********************************************************
 1000 ylo=y(1)
      ynewlo=ylo
      ilo=1
      ihi=1
      do 5 i=2,nn
        if (y(i).ge.ylo) go to 4
        ylo=y(i)
        ilo=i
    4   if (y(i).le.ynewlo) go to 5
        ynewlo=y(i)
        ihi=i
    5 continue
c***********************************************************
c     perform convergence checks on function
c***********************************************************
      dchk=(ynewlo+dabit)/(ylo+dabit)-1.0d0
      if (dabs(dchk).lt.reqmin) go to 900
      konvge=konvge-1
      if (konvge.ne.0) go to 2020
      konvge=5
c***********************************************************
c     check convergence of coordinate
c     only  every 5 simplex
c***********************************************************
      do 2015 i=1,n
        coord1=p(i,1)
        coord2=coord1
        do 2010 j=2,nn
          if (p(i,j).ge.coord1) go to 2005
          coord1=p(i,j)
 2005     if (p(i,j).le.coord2) go to 2010
          coord2=p(i,j)
 2010   continue
        dchk=(coord2+dabit)/(coord1+dabit)-1.0d0
        if (dabs(dchk).gt.reqmin) go to 2020
 2015 continue
      go to 900
 2020 if (icount.ge.kcount) go to 900
c***********************************************************
c     calculate pbar, the centroid of the simplex
```

```
c     vertices except thjat with y value ynewlo
c*************************************************************
      do 7 i=1,n
        z=0.0d0
        do 6 j=1,nn
          z=z+p(i,j)
    6   continue
        z=z-p(i,ihi)
    7 pbar(i)=z/float(n)
c*************************************************************
c     reflection through the centroid
c*************************************************************
      do 8 i=1,n
    8 pstar(i)=(1.0d0+rcoeff)*pbar(i)-rcoeff*p(i,ihi)
c
      if(n.eq.2)call clg2a2(pstar,f,pcs)
      if(n.eq.4)call clg3out(pstar,f,pcs)
c      print*,'nm f= ',f
      ystar=f
      icount=icount+1
      if (ystar.ge.ylo) go to 12
      if (icount.ge.kcount) go to 19
c*************************************************************
c     successful reflection, so extension
c*************************************************************
      do 9 i=1,n
    9 p2star(i)=ecoeff*pstar(i)+(1.0d0-ecoeff)*pbar(i)
c
      if(n.eq.2)call clg2a2(p2star,f,pcs)
      if(n.eq.4)call clg3out(p2star,f,pcs)
      y2star=f
      icount=icount+1
c*************************************************************
c     retain extension or contraction
c*************************************************************
      if (y2star.ge.ystar) go to 19
   10 do 11, i=1,n
   11 p(i,ihi)=p2star(i)
      y(ihi)=y2star
      go to 1000
c*************************************************************
c     no extension
c*************************************************************
   12 l=0
      do 13 i=1,nn
        if (y(i).gt.ystar) l=l+1
   13 continue
      if (l.gt.1) go to 19
      if (l.eq.0) go to 15
c*************************************************************
c     contraction on the reflection side of the centroid
c*************************************************************
      do 14 i=1,n
   14 p(i,ihi)=pstar(i)
```

```
    y(ihi)=ystar
c***********************************************************
c    contraction on the y(ihi) side of the centroid
c***********************************************************
  15 if (icount.ge.kcount) go to 900
    do 16 i=1,n
  16 p2star(i)=ccoeff*p(i,ihi)+(1.0d0-ccoeff)*pbar(i)
c
    if(n.eq.2)call clg2a2(p2star,f,pcs)
    if(n.eq.4)call clg3out(p2star,f,pcs)
c      print*,'nm f= ',f
    y2star=f
    icount=icount+1
    if (y2star.lt.y(ihi)) go to 10
c***********************************************************
c    contract the whole simplex
c***********************************************************
    do 18 j=1,nn
      do 17 i=1,n
        p(i,j)=(p(i,j)+p(i,ilo))*0.5d0
  17    xmin(i)=p(i,j)
c
      if(n.eq.2)call clg2a2(xmin,f,pcs)
    if(n.eq.4)call clg3out(xmin,f,pcs)
c      print*,'nm f= ',f
      Y(j)=f
  18 continue
    icount=icount+nn
    if (icount.lt.kcount) go to 1000
    go to 900
c**********************************************************
c    retain reflection
c**********************************************************
  19 continue
    do 20 i=1,n
  20 p(i,ihi)=pstar(i)
    y(ihi)=ystar
    go to 1000
 900 do 23 j=1,nn
      do 22 i=1,n
  22    xmin(i)=p(i,j)
c
      if(n.eq.2)call clg2a2(xmin,f,pcs)
    if(n.eq.4)call clg3out(xmin,f,pcs)
c      write(*,*)"nm f= ",f
      y(j)=f
  23 continue
    ynewlo=bignum
    do 24 j=1,nn
      if (y(j).ge.ynewlo) go to 24
      ynewlo=y(j)
      pclo(1)=pcs(1)
      pclo(2)=pcs(2)
      pclo(3)=pcs(3)
```

```
      pclo(4)=pcs(4)
      pclo(5)=pcs(5)
      pclo(6)=pcs(6)
      pclo(7)=pcs(7)
      pclo(8)=pcs(8)
      pclo(9)=pcs(9)
      ibest=j
   24 continue
     y(ibest)=bignum
     ysec=bignum
     do 25 j=1,nn
      if (y(j).ge.ysec) go to 25
      ysec=y(j)
      isec=j
   25 continue
     do 26 i=1,n
      xmin(i)=p(i,ibest)
      xsec(i)=p(i,isec)
   26 continue
c     write(*,*)"ynewlo= ",ynewlo
      return
      end
c*************************************************
c     ***********************************************
c     ***********************************************
c     this program to calculate expected total cost of a
c     sampling plan, given user inputted cutoff values
c
c     NOTE:  Do not handle bias in this program module
c     (although it is entered as a param) because I assume
c     that the measurements and cutoffs will have bias built
c     into them (if existing).
c     igo=selection from main
c*************************************************
c
      subroutine calcst(igo)
c
      real*8 c1(2),c2(2),c3,cmax,pcs(9),ctmp,tcst
      character*1 iopt
      character*5 spec
      dimension thta(4),tau(4)
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u
      common /cult/ cmax
      spec='Upper'
5     continue
      iopt=' '
      print*
      print*
      write(*,*)'This program module calculates the expected'
      write(*,*)'total cost of a given sequential sampling'
      write(*,*)'plan.  The user must supply the sequential'
      write(*,*)'decision cutoff values.'
      write(*,*)
```

```
      write(*,*)'Measurement System Parameter Entry:'
      write(*,*)
      call vardef(u,thta,tau,bias,sme,s1,a2,r2,nspc)
      write(*,*)
10    write(*,*)
      tcst=0.d0
      write(*,*)'What is the maximum number of iterations'
      write(*,*)'for the plan? (1, 2 or 3)'
      write(*,*)
      read(*,*,err=10)nmx
      print*
      if(nmx.lt.1.or.nmx.gt.3)then
        print*,'**** Iteration Limit out of range.
     &  Please Reenter. ****'
        print*
        goto 10
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c     this to account for lower spec by symmetry, only
c     Use given spec, but find for symmetrical prior mean
      if(nspc.eq.2)then
        thta(1)=2*u-thta(1)
        spec='Lower'
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      if(nmx.eq.1)then
20        print*,'Enter the value of C1'
        print*
        read(*,*,err=20)c1(1)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
        c1(2)=c1(1)
        goto 200
      endif
30    print*,'Enter the value of C1,L'
      print*
      read(*,*,err=30)c1(1)
      print*
40    print*,'Enter the value of C1,H'
      print*
      read(*,*,err=40)c1(2)
      print*
      if(c1(2).le.c1(1))then
        print*,'**** C1,L cannot exceed C1,H ****'
        print*
        goto 30
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      if(nmx.eq.2)then
50        print*,'Enter the value of C2'
        print*
```

```
        read(*,*,err=50)c2(1)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
        c2(2)=c2(1)
        goto 200
      endif
60    print*,'Enter the value of C2,L'
      print*
      read(*,*,err=60)c2(1)
      print*
70    print*,'Enter the value of C2,H'
      print*
      read(*,*,err=70)c2(2)
      print*
      if(c2(2).le.c2(1))then
        print*,'**** C2,L cannot exceed C2,H ****'
        print*
        goto 60
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
80    print*,'Enter the value of C3'
      print*
      read(*,*,err=80)c3
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      print*
200   continue
c
      print*
      write(*,400)nmx
      write(*,*)
      if(nmx.eq.1)then
        write(*,410)c1(1)
        goto 210
      endif
      write(*,420)c1(1),c1(2)
      if(nmx.eq.2)then
        write(*,430)c2(1)
        goto 210
      endif
      write(*,440)c2(1),c2(2)
      write(*,450)c3
210   continue
      print*
      print*,'Is the above information correct?'
      print*,'Y to accept or N to reenter cutoff values.'
      print*
      read(*,460)iopt
      print*
      if(iopt.ne.'y'.and.iopt.ne.'Y'.and.iopt.ne.'n'.
     &    and.iopt.ne.'N')then
        print*,'**** Invalid Entry.  Please Reenter. ****'
        print*
```

```
      goto 200
    endif
    if(iopt.eq.'N'.or.iopt.eq.'n')goto 10
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c    this to account for lower spec by symmetry, only
c    Use given spec, but find for symmetrical prior mean
    if(nspc.eq.2)then
      ctmp=c1(1)
      c1(1)=2.*u-c1(2)
      c1(2)=2.*u-ctmp
      ctmp=c2(1)
      c2(1)=2.*u-c2(2)
      c2(2)=2.*u-ctmp
      c3=2.*u-c3
    endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c
c
220  continue
    call errchk1(c1,pcs)
    if(nmx.gt.1)then
      call errchk2(nmx,c1,c2,c3,pcs)
    endif
    if(nspc.eq.2)then
      thta(1)=2*u-thta(1)
      ctmp=c1(1)
      c1(1)=2.*u-c1(2)
      c1(2)=2.*u-ctmp
      ctmp=c2(1)
      c2(1)=2.*u-c2(2)
      c2(2)=2.*u-ctmp
      c3=2.*u-c3
    endif
    print*
    print*,'Expected Costs of Sampling Plan:'
    write(*,*)
    do 300 i=1,nmx
    write(*,470)i,pcs(3.d0*(i-1)+1)
    write(*,475)i,pcs(3.d0*(i-1)+2)
    write(*,480)i,pcs(3.d0*(i-1)+3)
    tcst=tcst+pcs(3*(i-1)+1)+pcs(3*(i-1)+2)+pcs(3*(i-1)+3)
300  continue
    write(*,*)
    write(*,650)tcst
    print*
305  iopt=' '
310  print*,'Send output to file? (Y/N)'
    read(*,460)iopt
    if(iopt.ne.'Y'.and.iopt.ne.'N'.and.iopt.ne.'y'.and.
   &iopt.ne.'n')goto 310
    if(iopt.eq.'n'.or.iopt.eq.'N')goto 315
    write(2,510)thta(1)
```

```
      write(2,520)tau(1)
      write(2,530)spec,u
      write(2,540)sme
      write(2,550)bias
      write(2,560)
      write(2,570)s1
      write(2,580)a2
      write(2,590)r2
      write(2,600)
      if(nmx.eq.2)goto 311
      if(nmx.eq.1)goto 312
      write(2,602)c3
      write(2,604)c2(1)
311   continue
      write(2,610)c2(2)
      write(2,614)c1(1)
312   continue
      write(2,616)c1(2)
      write(2,618)
      write(2,620)pcs(1)
      write(2,622)pcs(2)
      write(2,624)pcs(3)
      if(nmx.eq.1)goto 313
      write(2,626)pcs(4)
      write(2,628)pcs(5)
      write(2,630)pcs(6)
      if(nmx.eq.2)goto 313
      write(2,632)pcs(7)
      write(2,634)pcs(8)
      write(2,636)pcs(9)
313   continue
      write(2,650)tcst
315   iopt=' '
      print*
      print*,'Would you like to input another set of cutoffs?'
      print*,'Enter Y or N.'
      print*
      read(*,460)iopt
      if(iopt.eq.'Y'.or.iopt.eq.'y')goto 10
      if(iopt.eq.'n'.or.iopt.eq.'N')goto 390
      goto 315
390   continue
400   format(' Maximum iterations = ',i1)
410   format(' C1 =  ',f11.4)
420   format(' C1,L = ',f11.4,5x,'C1,H = ',f11.4)
430   format(' C2 =  ',f11.4)
440   format(' C2,L = ',f11.4,5x,'C2,H = ',f11.4)
450   format(' C3 =  ',f11.4)
460   format(a1)
470   format('   Sampling on     ',i1,' =  ',f12.4)
475   format('   False Accept on ',i1,' =  ',f12.4)
480   format('   False Reject on ',i1,' =  ',f12.4)
510   format(' Prior Mean =          ',f16.8)
520   format(' Prior Std. Dev =      ',f16.8)
```

```
530  format(1x,a5,' Specification =  ',f16.8)
540  format(' Meas. Error Std. Dev. =   ',f13.8)
550  format(' Meas. Error Mean (Bias) = ',f13.8)
560  format(' Input Costs:')
570  format('    Iteration (S) =     ',f12.4)
580  format('    False Accept (A) =  ',f12.4)
590  format('    False Reject (R) =  ',f12.4)
600  format(' **************************************')
602  format(' Cutoff C3 =            ',f16.8)
604  format(' Cutoff C2,L =          ',f16.8)
610  format(' Cutoff C2,H =          ',f16.8)
614  format(' Cutoff C1,L =          ',f16.8)
616  format(' Cutoff C1,H =          ',f16.8)
618  format(' Expected Plan Costs:')
620  format('   Sampling on 1 =      ',f12.4)
622  format('   False Accept on 1 =  ',f12.4)
624  format('   False Reject on 1 =  ',f12.4)
626  format('   Sampling on 2 =      ',f12.4)
628  format('   False Accept on 2 =  ',f12.4)
630  format('   False Reject on 2 =  ',f12.4)
632  format('   Sampling on 3 =      ',f12.4)
634  format('   False Accept on 3 =  ',f12.4)
636  format('   False Reject on 3 =  ',f12.4)
650  format(' Expected Total Cost =  ',f12.4)
     igo=3
     return
     end
c****************************************************
c    ***********************************************
c    ***********************************************
c    error checking and cost calculating for the
c    case of user-input C1 and program cost request
c****************************************************
c
     subroutine errchk1(c1,pcs)
     real*8 pcs(9),a,b,ap,bp,c1(2),rej,acc
     real*8 x1,c1l,c1h
     dimension thta(4),tau(4)
     common /costs/ s1,a2,r2
     common /parms/ tau,sme,thta,u
     common /ncur/ j
     c1l=c1(1)
     c1h=c1(2)
     j=1
cccccccccccccccccccccccccccccccccccccccccccccccccc
     tau(2)=sqrt(1./(1./sme**2.+1./tau(1)**2.))
c    these routines set the infinity limits for the functions
     call setabx(a,b,j)
     call setmor(ap,bp,j)
cccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccc
4    continue
     tau(3)=sqrt(1./(1./sme**2.+1./tau(2)**2.))
c    start x1 at a, work up
```

```
      x1=a
c     sequence that checks if cutoffs are beyond inf limits
      if(b.lt.a)then
      if(c1l.lt.a)then
         flg=1.
         acc=0.d0
         if(c1h.lt.bp)then
            call qgaus(j,flg,bp,b,rej)
            goto 300
         endif
         if(c1l.gt.b)then
            rej=0.d0
            goto 300
         endif
         call qgaus(j,flg,c1h,b,rej)
      else
         flg=2.
         rej=0.d0
         if(c1l.gt.ap)then
            call qgaus(j,flg,a,ap,acc)
            goto 300
         else
            call qgaus(j,flg,a,c1l,acc)
         endif
      endif
      else
      if(c1l.lt.a)then
         flg=1.
         acc=0.d0
         if(c1h.lt.bp)then
            call qgaus(j,flg,bp,b,rej)
            goto 300
         endif
         if(c1h.lt.b)then
            call qgaus(j,flg,c1h,b,rej)
c              print*,'region 2 c1h= ',c1h
c              print*,'b= ',b
            goto 300
         endif
         rej=0.d0
      else
         if(c1h.lt.b)then
            flg=1.
            call qgaus(j,flg,c1h,b,rej)
               else
            rej=0.d0
         endif
         flg=2.
         if(c1l.gt.ap)then
            call qgaus(j,flg,a,ap,acc)
         else
            call qgaus(j,flg,a,c1l,acc)
         endif
      endif
```

```
      endif
300   continue
      pcs(1)=dble(s1)
      pcs(2)=dble(a2)*acc
      pcs(3)=dble(r2)*rej
      return
      end
c****************************************************
c     **********************************************
c     **********************************************
c     error checking and cost calculating for the
c     case of user-input C1 and program cost request
c     c1 array carries c1l and c1h
c     c2 array carries c2l and c2h
c     pcs carries cost components
c****************************************************
c
      subroutine errchk2(nmx,c1,c2,c3,pcs)
      real*8 a,b,ap,bp,c1(2),c2(2),c3,rej,acc
      dimension thta(4),tau(4),x(2),p(2)
      real*8 tc4int,tc5int,tc6int,pcs(9),tc7,tc8,tc9
      real*8 tc7int,tc8int,tc9int,bignum
      real*8 tc4pre,tc5pre,tc6pre,tc7pre,tc8pre,tc9pre
      real*8  c1l,c1h,c2l,c2h,delx,x1,x1pre
      real*8 alo,blo,ahi,bhi
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u
      common /inf2/ a,b
      common /ncur/ j
      bignum=9.99d+55
      tc4int=0.d0
      tc5int=0.d0
      tc6int=0.d0
      tc7int=0.d0
      tc8int=0.d0
      tc9int=0.d0
      istp=0
cccccccccccccccccccccccccccccccccccccccccccccccccc
      tau(2)=sqrt(1./(1./sme**2.+1./tau(1)**2.))
      tau(3)=sqrt(1./(1./sme**2.+1./tau(2)**2.))
cccccccccccccccccccccccccccccccccccccccccccccccccc
      c1l=c1(1)
      c1h=c1(2)
      c2l=c2(1)
      c2h=c2(2)
c     delx is step size
c     j is iteration number
      delx=(c1h-c1l)/100.d0
      j=2
c
c     this sequence finds the +/- infinity limits for the
c     conditional false dispositions at n=2, based on the
c     possible extreme values of x1 ==>C1l and C1h
c
```

```
      call post(j,c1l)
      call setabx(alo,blo,j)
      call post(j,c1h)
      call setabx(ahi,bhi,j)
      a=dmin1(alo,ahi)
      b=dmax1(blo,bhi)
c
      call post(j,c1l)
      call setmor(alo,blo,j)
      call post(j,c1h)
      call setmor(ahi,bhi,j)
      ap=dmax1(alo,ahi)
      bp=dmin1(blo,bhi)
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
      x1=c1l
20    continue
      j=2
      print*,' *******   working...'
      call post(j,x1)
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
      if(b.lt.a)then
      if(2.*c2l-x1.lt.a)then
        flg=1.
        acc=0.d0
        if(2*c2h-x1.lt.bp)then
          call qgaus(j,flg,bp,b,rej)
          goto 70
        endif
        if(2.*c2l-x1.gt.b)then
          rej=0.d0
          goto 70
        endif
        call qgaus(j,flg,2*c2h-x1,b,rej)
      else
        flg=2.
        rej=0.d0
        if(2.*c2l-x1.gt.ap)then
          call qgaus(j,flg,a,ap,acc)
          goto 70
        else
         call qgaus(j,flg,a,2*c2l-x1,acc)
        endif
      endif
      else
      if(2*c2l-x1.lt.a)then
        flg=1.
        acc=0.d0
        if(2.*c2h-x1.lt.bp)then
          call qgaus(j,flg,bp,b,rej)
          goto 70
        endif
        if(2.*c2h-x1.lt.b)then
```

```
          call qgaus(j,flg,2.*c2h-x1,b,rej)
          goto 70
        endif
        rej=0.d0
        goto 70
      else
        if(2.*c2h-x1.lt.b)then
          flg=1.
          call qgaus(j,flg,2.*c2h-x1,b,rej)
            else
          rej=0.d0
        endif
        flg=2.
        if(2.*c2l-x1.gt.ap)then
          call qgaus(j,flg,a,ap,acc)
          goto 70
        else
          call qgaus(j,flg,a,2.*c2l-x1,acc)
          goto 70
        endif
      endif
      endif
70   continue
      cost4=s1
      costa5=acc*a2
      costr6=rej*r2
      if(nmx.gt.2)then
        call errchk3(c2,x1,c3,tc7,tc8,tc9)
      endif
c****************************************************
      if (istp.eq.0)then
        tc4pre=cost4
        tc5pre=costa5
        tc6pre=costr6
        tc7pre=tc7
        tc8pre=tc8
        tc9pre=tc9
        x1pre=x1
        istp=1
          else
c    marginal on x1 is N(theta1,sme^2+tau1^2)
        x(1)=(x1pre-thta(1))/sqrt(tau(1)**2.+sme**2.)
        x(2)=(x1-thta(1))/sqrt(tau(1)**2.+sme**2.)
        call normal(x,p)
        prob=p(2)-p(1)
        tc4int=tc4int+prob*((cost4+tc4pre)/2.d0)
        tc5int=tc5int+prob*((costa5+tc5pre)/2.d0)
        tc6int=tc6int+prob*((costr6+tc6pre)/2.d0)
        tc7int=tc7int+prob*((tc7+tc7pre)/2.d0)
        tc8int=tc8int+prob*((tc8+tc8pre)/2.d0)
        tc9int=tc9int+prob*((tc9+tc9pre)/2.d0)
        tc4pre=cost4
        tc5pre=costa5
        tc6pre=costr6
```

```
      tc7pre=tc7
      tc8pre=tc8
      tc9pre=tc9
      x1pre=x1
    endif
c**************************************************
    x1=x1+delx
c   this next chunk of code attempts to account for the
c   slack between the last x1 and c1h
    if (x1.gt.c1h)then
     tc4int=tc4int+((c1h-x1pre)/delx)*prob*((cost4+tc4pre)/2.d0)
     tc5int=tc5int+((c1h-x1pre)/delx)*prob*((costa5+tc5pre)/2.d0)
     tc6int=tc6int+((c1h-x1pre)/delx)*prob*((costr6+tc6pre)/2.d0)
     tc7int=tc7int+((c1h-x1pre)/delx)*prob*
   &      ((tc7+tc7pre)/2.d0)
     tc8int=tc8int+((c1h-x1pre)/delx)*prob*
   &      ((tc8+tc8pre)/2.d0)
     tc9int=tc9int+((c1h-x1pre)/delx)*prob*
   &      ((tc9+tc9pre)/2.d0)
      goto 80
    endif
    goto 20
80  continue
    pcs(4)=tc4int
    pcs(5)=tc5int
    pcs(6)=tc6int
    pcs(7)=tc7int
    pcs(8)=tc8int
    pcs(9)=tc9int
    return
    end
c*************************************************
c   *********************************************
c   *********************************************
c   error checking and cost calculating for the
c   case of user-input C1 and program cost request
c   c2 carries c2l and c2h
c   pcs carries cost components
c   x1 is obs 1 value at which function is evaluated
c*************************************************
c
    subroutine errchk3(c2,x1,c3,tc7int,tc8int,tc9int)
    real*8 a,b,ap,bp,alst,blst,c2(2),rej,acc
    dimension thta(4),tau(4),x(2),p(2)
    real*8 tc7int,tc8int,tc9int
    real*8 tc7pre,tc8pre,tc9pre,delx2,x1,x2
    real*8  c2l,c2h,c3,x2pre
    real*8 alo,blo,ahi,bhi
    common /costs/ s1,a2,r2
    common /parms/ tau,sme,thta,u
    common /inf2/ alst,blst
    common /ncur/ j
    tc7int=0.d0
    tc8int=0.d0
```

```
      tc9int=0.d0
      istp=0
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      tau(4)=sqrt(1./(1./sme**2.+1./tau(3)**2.))
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      c2l=c2(1)
      c2h=c2(2)
c     delx2 is step size
c     j is iteration number
      delx2=(blst-alst)/100.d0
      j=3
c
c     this sequence finds the +/- infinity limits for the
c     conditional false dispositions at n=2, based on the
c     possible extreme values of x1 ==>C1l and C1h
c
      call post(j,2.d0*c2l-x1)
      call setabx(alo,blo,j)
      call post(j,2.d0*c2l-x1)
      call setabx(ahi,bhi,j)
      a=dmin1(alo,ahi)
      b=dmax1(blo,bhi)
c
      call post(j,2.d0*c2l-x1)
      call setmor(alo,blo,j)
      call post(j,2.d0*c2l-x1)
      call setmor(ahi,bhi,j)
      ap=dmax1(alo,ahi)
      bp=dmin1(blo,bhi)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      x2=2.d0*c2l-x1
20    continue
      call post(3,x2)
      if(c3.lt.a)then
         acc= 0.d0
        else
         flg=2.
         if(c3.gt.ap)then
            call qgaus(j,flg,a,ap,acc)
           else
            call qgaus(j,flg,a,3.d0*c3-x1-x2,acc)
         endif
      endif
      if(c3.gt.b)then
         rej= 0.d0
        else
         flg=1.
         if(c3.lt.bp)then
            call qgaus(j,flg,bp,b,rej)
           else
            call qgaus(j,flg,3.d0*c3-x1-x2,b,rej)
         endif
      endif
```

```
      cost7=s1
      costa8=acc*a2
      costr9=rej*r2
c*****************************************************
      if (istp.eq.0)then
        tc7pre=dble(cost7)
        tc8pre=dble(costa8)
        tc9pre=dble(costr9)
        x2pre=x2
        istp=1
          else
c     marginal on x2|x1 is N(theta2,sme^2+tau2^2)
        x(1)=(x2pre-thta(2))/sqrt(tau(2)**2.+sme**2.)
        x(2)=(x2-thta(2))/sqrt(tau(2)**2.+sme**2.)
        call normal(x,p)
        prob=p(2)-p(1)
        tc7int=tc7int+prob*((cost7+tc7pre)/2.d0)
        tc8int=tc8int+prob*((costa8+tc8pre)/2.d0)
        tc9int=tc9int+prob*((costr9+tc9pre)/2.d0)
        tc7pre=cost7
        tc8pre=costa8
        tc9pre=costr9
        x2pre=x2
      endif
c*****************************************************
      x2=x2+delx2
c     this next chunk of code accounts for the
c     slack between the last x1 and c1h
      if (x2.gt.2.*c2h-x1)then
       tc7int=tc7int+((2.*c2h-x1-x2pre)/delx2)*prob*
     &       ((cost7+tc7pre)/2.d0)
       tc8int=tc8int+((2.*c2h-x1-x2pre)/delx2)*prob*
     &       ((costa8+tc8pre)/2.d0)
       tc9int=tc9int+((2.*c2h-x1-x2pre)/delx2)*prob*
     &       ((costr9+tc9pre)/2.d0)
        goto 80
      endif
      goto 20
80    continue
      return
      end
c     *********************************************
c     *********************************************
c     *********************************************
c
c     decision sub to disposition batch of unknown sige
c
c     bias=measurement error mean
c     beta=prob of being wrong when actual>the1
c     alpha=prob of being wrong when actual<the0
c     del=mu/sig=upper indiff limit
c     delp=mu'/sig=lower indiff limit
c     xobs()=array of observed measurements, limit=50
c        bias subtracted from xobs() prior to calculating
```

```
c    xadj()=array of adjusted measurements (by USpec),limit=10
c    dlna=lnA
c    dlnb=lnB
c    dllr=ln of the likelihood ratio
c    sumx=sum of the xadj array, from 1 to n
c    sumx2=sum of the sqs of the xadj array, from 1 to n
c*************************************************
c
      subroutine unkn(igo)
c
      real*8 sumx,sumx2,thep,xobs(50),xadj(50),
     &un,gdelu,gdelpu,dllr
      real*8 delu,delpu,fdu1,fdu2,fdpu1,fdpu2,gaman2,gamn1
      character iopt
      character*28 tag
      un=0.d0
      nsgs=1
2       print*
      write(*,*)'What is the maximum number of iterations which'
      write(*,*)'you wish to make (cannot exceed 50)?'
      print*
      read(*,*,err=2)nunk
      if(nunk.le.1.or.nunk.gt.50)then
         print*
         write(*,*)'**** Invalid Entry.  Please Reenter. ****'
         goto 2
      endif
      do 10 j=1,nunk
       xobs(j)=0.0d0
       xadj(j)=0.0d0
10      continue
      iopt='0'
      sumx=0.0d0
      sumx2=0.0d0
      print*
      write(*,*)'Enter the Specification Limit.'
      print*
      read(*,*,err=500)thep
      print*
12       write(*,*)'Enter the ratio of mean/standard deviation to be'
      write(*,*)'tested for the null hypothesis.'
      print*
      read(*,*,err=500)del
      print*
14       write(*,*)'Enter the ratio of mean/standard deviation to be'
      write(*,*)'tested for the alternative hypothesis.'
      print*
      read(*,*,err=500)delp
      print*
      write(*,*)'Enter Alpha, acceptable Type I error probability,'
      write(*,*)'associated with a true null hypothesis'
      write(*,*)'(0 to 1).'
      print*
      read(*,*,err=500)alpha
```

```
      if(alpha.lt.0.or.alpha.gt.1.)goto 500
      print*
      write(*,*)'Enter Beta, acceptable Type II error probability,'
      write(*,*)'associated with a true alternative hypothesis'
      write(*,*)'(0 to 1).'
      print*
      read(*,*,err=500)beta
      if(beta.lt.0.or.beta.gt.1.)goto 500
      print*
      print*,'Enter the measurement error bias.'
      print*,'Sign Convention: If the instrument reads higher '
      print*,'than the true value, this bias should be positive.'
      print*
      read(*,*,err=500)bias
      print*
20    continue
      print*
      write(*,100)thep
      write(*,110)del
      write(*,120)delp
      write(*,130)alpha
      write(*,140)beta
      write(*,145)bias
      write(*,*)
      write(*,*)'Is the above information correct?'
      write(*,*)'Enter to accept, or # of parameter to reenter.'
      print*
      read(*,150)iopt
      if(iopt.eq.' ')goto 40
      if(iopt.gt.'6')goto 600
        if(iopt.eq.'1')tag='Specification Limit'
        if(iopt.eq.'2')tag='Null Hyp. Ratio of Mean/Std. Dev.'
        if(iopt.eq.'3')tag='Alt. Hyp. Ratio of Mean/Std. Dev.'
        if(iopt.eq.'4')tag='Alpha'
        if(iopt.eq.'5')tag='Beta'
        if(iopt.eq.'6')tag='Measurement Bias'
      goto 700

40    continue
      dlna=log((1.0-beta)/alpha)
      dlnb=log(beta/(1.0-alpha))
      do 80 n=1,nunk
       print*
       write(*,*)'Enter measurement observation #',n
       print*
       read(*,*)xobs(n)
       xobs(n)=xobs(n)-bias
       xadj(n)=xobs(n)-thep
       sumx=sumx+xadj(n)
       sumx2=sumx2+xadj(n)**2.d0
       if(sumx2.gt.0.)then
        un=sumx/dsqrt(sumx2)
       endif
       delpu=delp*un
```

```
      delu=del*un
c
c      sub calls
c
     call sumch(n,1.d0,.5d0*delu**2.d0,fdu1)
     call sumch(n,1.d0,.5d0*delpu**2.d0,fdpu1)
     call sumch(n+1,0.d0,.5d0*delu**2.d0,fdu2)
     call sumch(n+1,0.d0,.5d0*delpu**2.d0,fdpu2)
     call gamn(n,gamn1,gaman2)
c
     gdelu=dlog(fdu1+dsqrt(2.d0)*delu*fdu2*gamn1/gaman2)
     gdelpu=dlog(fdpu1+dsqrt(2.d0)*delpu*fdpu2*gamn1/gaman2)
c
c
      dllr=gdelpu-gdelu-.5d0*n*((delp**2.d0)-(del**2.0d0))
      if(dllr.lt.dlnb)then
        print*
        write(*,160)dllr
        write(*,170)dlnb
        write(*,*)'********************************************'
        write(*,*)'****** Do Not Reject Null Hypothesis ******'
        write(*,*)'********************************************'
        print*
        goto 1000
      endif
      if(dllr.gt.dlna)then
        print*
        write(*,180)dllr
        write(*,190)dlna
        write(*,*)'************************************'
        write(*,*)'****** Reject Null Hypothesis ******'
        write(*,*)'************************************'
        print*
        goto 1000
      endif
80    continue
     print*
     write(*,*)'No decision reached.  '
     write(*,155)dllr
     write(*,157)dlna
     write(*,158)dlnb
     goto 1000
c
c
100   format(' 1  Specification Limit=               ',f6.2)
110   format(' 2  Null Hyp. Ratio of Mean/Std. Dev.= ',f6.2)
120   format(' 3  Alt. Hyp. Ratio of Mean/Std. Dev.= ',f6.2)
130   format(' 4  Alpha=                             ',f4.2)
140   format(' 5  Beta=                              ',f4.2)
145   format(' 6  Measurement Bias=                  ',f6.2)
150   format(a1)
155   format(' Ln of likelihood ratio= ',f11.4)
157   format(' Upper limit (lnA) = ',f11.4)
158   format(' Lower limit (lnB) = ',f11.4)
```

```
160  format(' Ln of Likelihood ratio, ',f11.4,', less than')
170  format(' ln of B, ',f11.4,'.')
180  format(' Ln of Likelihood ratio, ',f11.4,', greater than')
190  format(' ln of A, ',f11.4,'.')
c
c
500  print*
    write(*,*)'**** Invalid Entry.  Please Reenter Parameters. ****'
    goto 2
600  print*
    write(*,*)'**** Invalid Entry.  Please Reenter. ****'
    goto 20
700  print*
    write(*,*)'Enter ',tag,'.'
      print*
      read(*,*)crct
      if(iopt.eq.'1')thep=crct
      if(iopt.eq.'6')bias=crct
      if(iopt.eq.'2')del=crct
      if(iopt.eq.'3')delp=crct
      if(iopt.eq.'4')then
        if(crct.ge.0.and.crct.le.1.)then
          alpha=crct
         else
          goto 700
        endif
      endif
      if(iopt.eq.'5')then
        if(crct.ge.0.and.crct.le.1.)then
          beta=crct
         else
          goto 700
        endif
      endif
    goto 20
1000 continue
    igo=3
    print*,'Enter to continue'
    read(*,150)iopt
    return
    end
c****************************************************
c   ***********************************************
c   ***********************************************
c   summation program for con hyp
c   both odd and even n
c   if ginc=1., then gamma=1/2
c   if ginc=0., then gamma =3/2
c****************************************************
    subroutine sumch(n,ginc,x,f)
    real*8 jfct,gamg,gamgj,gama,gamaj,f,x,fold,ginc
    if(ginc.eq.0d0)then
      gamg=.5d0*dsqrt(3.141593d0)
        else
```

```
      gamg=dsqrt(3.141593d0)
    endif
2     gamgj=gamg
    jfct=1.d0
    ia=n/2
    if(mod(n,2).eq.0)then
      gama=1.d0
      do 10 i=1,ia-1
        gama=gama*i
10      continue
        else
      gama=dsqrt(3.141593d0)
      do 15 i=1,ia
        gama=(i-.5d0)*gama
15      continue
    endif
    gamaj=gama
c
c     for j=0, f=x
c
    f=1.d0
    fold=f
    do 20 j=1,100
      jfct=jfct*j
      gamgj=gamgj*(j+.5d0-ginc)
      if(mod(n,2).eq.0)then
        gamaj=gamaj*(ia+j-1)
          else
        gamaj=gamaj*(ia+j-.5d0)
      endif
      f=f+(gamg*gamaj*x**j)/(gama*gamgj*jfct)
      if(dabs(f-fold).le.1.e-8)then
        goto 22
      endif
      fold=f
20    continue
22    continue
    return
    end
c**************************************************
c     *********************************************
c     *********************************************
c     subroutine to solve the gamma function
c**************************************************
c
    subroutine gamn(n,gamn1,gaman2)
    real*8 gamn1,gamano2,gamane2,gaman2
2     gamano2=dsqrt(3.141593d0)
    gamane2=1.d0
    ia=n/2
    do 5 i=1,n
      if(mod(i,2).eq.0.and.i.gt.2)then
        gamane2=gamane2*(i/2.d0-1.d0)
      endif
```

```
      if(mod(i,2).gt.0.and.i.gt.1)then
         gamano2=gamano2*(i/2.d0-1.d0)
      endif
5    continue
     if(mod(n,2).eq.0)then
       gamn1=(ia-.5d0)*dsqrt(3.141593d0)
       gaman2=gamane2
       do 10 i=1,ia-1
c    gamn1 is for unk2 --(n+1)is odd
         gamn1=gamn1*(i-.5d0)
10      continue
         else
c    gamn1 for unk2 -- (n+1) even
       gamn1=1.d0
       gaman2=gamano2
       do 15 i=1,ia
         gamn1=gamn1*i
15      continue
     endif
c
     return
     end
c***************************************************
c    ***********************************************
c    ***********************************************
c    program known
c    decision sub to disposition batch of known sige
c
c    sige=measurement error standard deviation
c    bias=measurement error mean
c    the0=accept indifference limit, theta0
c    the1=reject indifference limit, theta1
c    thep= specification limit, thetaprime
c    beta=prob of being wrong when actual>the1
c    alpha=prob of being wrong when actual<the0
c    xobs()=array of observed measurements, limit=50
c       this array of observations is adjusted for bias
c    dlna=lnA
c    dlnb=lnB
c    dllr=ln of the likelihood ratio
c    sumx=sum of the xobs array, from 1 to n
c***************************************************
     subroutine known(igo)
c
     dimension xobs(50)
     character iopt
     character*25 tag,tag1,tag2
     character*5 tag3
     bias=0.
     iflg=0
     nsgn=1
     tag1='less '
     tag2='greater'
     tag3='Upper'
```

```
2    print*
     write(*,*)'What is the maximum number of iterations which'
     write(*,*)'you wish to make (cannot exceed 50)?'
     print*
     read(*,*,err=2)k
     if(k.le.0.or.k.gt.50)then
        print*
        write(*,*)'**** Invalid Entry.  Please Reenter. ****'
        go to 2
     endif
     do 3 j=1,k
     xobs(j)=0.0
3    continue
4    iopt='0'
     sumx=0.0
     print*
10   write(*,*)'Enter the standard deviation of the measurement'
     write(*,*)'error distribution.'
     print*
     read(*,*,err=130)sige
     if(sige.le.0.)then
        print*
        write(*,*)'**** Standard Deviation must be positive. ****'
        goto 4
     endif
     if(iflg.eq.1)goto 50
     print*
     print*,'Enter the measurement error bias.'
     print*,'Sign Convention: If the instrument reads higher '
     print*,'than the true value, this bias should be positive.'
     print*
     read(*,*,err=130)bias
     print*
20   print*
     write(*,*)'Enter the Specification Limit.'
     print*
     read(*,*,err=130)thep
     print*
30   write(*,*)'Is this an Upper (1) or Lower (2) Spec?'
     print*
     write(*,*)'Enter 1 or 2.'
     print*
     read(*,*,err=30)nspc
     if(nspc.ne.1.and.nspc.ne.2)then
        print*
        write(*,*)'**** Invalid Entry.  Please Reenter. ****'
        print*
        goto 30
     endif
     if(nspc.eq.2)then
        nsgn=-1
        tag1='greater'
        tag2='less'
        tag3='Lower'
```

```
      endif
      thep=nsgn*thep
34    print*
      write(*,*)'Enter the Acceptance Indifference Limit.'
      write(*,*)'(Beyond which acceptance is preferred)'
      print*
      read(*,*,err=130)the0
      the0=nsgn*the0
      if(the0.gt.thep)then
        print*
        write(*,84)tag1
        goto 20
      endif
      print*
36      write(*,*)'Enter the Rejection Indifference Limit.'
      write(*,*)'(Beyond which rejection is preferred)'
      print*
      read(*,*,err=130)the1
      the1=nsgn*the1
      if(the1.lt.thep)then
        print*
        write(*,85)tag2
        goto 20
      endif
      print*
40    print*
      write(*,*)'Enter Alpha, Type I Error probability (0 to 1).'
      print*
      read(*,*,err=130)alpha
      print*
      if(alpha.lt.0.or.alpha.gt.1.)then
         print*
         write(*,*)'**** Invalid Entry.  Please Reenter. ****'
         goto 40
      endif
      print*
      if(iflg.eq.1)goto 50
45    print*
      write(*,*)'Enter Beta, Type II Error probability (0 to 1).'
      print*
      read(*,*,err=130)beta
      print*
      if(beta.lt.0.or.beta.gt.1.)then
         print*
         write(*,*)'**** Invalid Entry.  Please Reenter. ****'
         goto 45
      endif
      print*
      if(iflg.eq.1)goto 50
50    continue
      print*
      write(*,92)sige
      write(*,102)bias
      write(*,93)tag3,real(nsgn)*thep
```

```
      write(*,94)real(nsgn)*the0
      write(*,95)real(nsgn)*the1
      write(*,96)alpha
      write(*,97)beta
60    write(*,*)
      write(*,*)'Is the above information correct?'
      write(*,*)'Enter to accept, or # of parameter to reenter. '
      print*
      read(*,91)iopt
      if(iopt.eq.' ')goto 70
      if(iopt.gt.'7')goto 120
        if(iopt.eq.'1')tag='Standard Deviation'
        if(iopt.eq.'3')tag='Specification Limit'
        if(iopt.eq.'4')tag='Accept Indifference Limit'
        if(iopt.eq.'5')tag='Reject Indifference Limit'
        if(iopt.eq.'6')tag='Alpha'
        if(iopt.eq.'7')tag='Beta'
        if(iopt.eq.'2')tag='Measurement bias'
        iflg=1
        goto 110
c
c
70    continue
      dlna=log((1.0-beta)/alpha)
      dlnb=log(beta/(1.0-alpha))
      do 80 j=1,k
      print*
      write(*,*)'Enter measurement observation #',j
      print*
      read(*,*)xobs(j)
      xobs(j)=nsgn*xobs(j)-bias
      sumx=sumx+xobs(j)
      dllr=((the1-the0)/sige**2.0)*sumx+(j/(2.0*sige**2.0))*
     \(the0**2.0-the1**2.0)
      if(dllr.lt.dlnb)then
        print*
        write(*,98)dllr
        write(*,99)dlnb
        write(*,*)'********************************'
        write(*,*)'********  Accept batch  ********'
        write(*,*)'********************************'
        print*
        goto 140
        elseif(dllr.ge.dlna)then
          print*
          write(*,100)dllr
          write(*,101)dlna
          write(*,*)'********************************'
          write(*,*)'********  Reject batch  ********'
          write(*,*)'********************************'
          print*
          goto 140
      endif
80    continue
```

```
      print*
      write(*,*)'Maximum number of iterations reached.'
      write(*,*)'Log of likelihood ratio = ',dllr
      write(*,*)'Acceptance limit = ',dlnb
      write(*,*)'Rejection limit = ',dlna
      write(*,*)'Wald Truncation Rule calls for'
      if(dllr.lt.0.)then
        write(*,*)'Acceptance.'
      else
        write(*,*)'Rejection.'
      endif
      goto 140
c
c
84    format(' **** Limit must be ',a7,' than or equal to spec. ****')
85    format(' **** Limit must be ',a7,' than or equal to spec. ****')
90    format(' ',f6.2)
91    format(a1)
92    format(' 1  Error Standard Deviation=  ',f6.2)
93    format(' 3  ',a5,' Specification=       ',f6.2)
94    format(' 4  Accept Indifference Limit= ',f6.2)
95    format(' 5  Reject Indifference Limit= ',f6.2)
96    format(' 6  Alpha=                     ',f4.2)
97    format(' 7  Beta=                      ',f4.2)
98    format(' Ln of likelihood ratio, ',f10.4,', less than')
99    format(' ln of B, ',f8.4,'.')
100   format(' Ln of likelihood ratio, ',f10.4,', greater than')
101   format(' ln of A, ',f8.4,'.')
102   format(' 2  Measurement Bias=          ',f6.2)
c
110   print*
      write(*,*)'Enter ',tag
      print*
      read(*,*,err=110)crct
        if(iopt.eq.'3')then
          crct=nsgn*crct
          if(crct.le.the1.and.crct.ge.the0)then
            thep=crct
           else
            goto 110
          endif
        endif
        if(iopt.eq.'2')bias=crct
        if(iopt.eq.'1')goto 10
        if(iopt.eq.'4')then
          crct=nsgn*crct
          if(crct.le.thep)then
            the0=crct
           else
            print*
            write(*,84)tag1
            write(*,*)'to ',tag3,' Specification.'
            goto 110
          endif
```

```
        endif
        if(iopt.eq.'5')then
          crct=nsgn*crct
          if(crct.ge.thep)then
            the1=crct
           else
            print*
            write(*,85)tag2
            write(*,*)'to ',tag3,' Specification.'
            goto 110
          endif
        endif
        if(iopt.eq.'6')goto 40
        if(iopt.eq.'7')goto 45
      goto 50
120   print*
      write(*,*)'**** Invalid Entry.  Please Reenter. ****'
      goto 50
130   print*
      write(*,*)'**** Invalid Entry.  Please Reenter Parameters. ****'
      write(*,*)
      goto 4
140   continue
      igo=3
      print*,'Enter to continue'
      read(*,91)iopt
      return
      end
```

APPENDIX B

ECONOMIC COMPUTER PROGRAMS UTILIZING THE PRIOR COSTING APPROACH

(FORTRAN Code Listings)

```
c*****************************************************
c*****************************************************
c*****************************************************
c    main for approach 1 (prior), nmax=1
c    "n" is number of unknowns, in this case, only the
c    single value of c1=c1l=c1h
c
      dimension thta(4),tau(4)
      real*8 start(10),step(10),xmin(20),xsec(20),tcnewlo,
     &tcsec,reqmin,a,b,pcs(3)
      character iopt
      character*5 spec
      common /costs/ s1,a2,r2,nspc
      common /parms/ tau,sme,thta,u,a,b
      common /ncur/ j
      spec='Upper'
      n=1
      call vardef(u,thta(1),tau(1),bias,sme,s1,a2,r2,nspc)
      if(nspc.eq.2)spec='Lower'
      reqmin=.000001d0
      icount=200
      j=1
      call setabx(a,b,j)
      if(b.le.a)then
        tcnewlo=s1
        pcs(1)=s1
        pcs(2)=0.d0
        pcs(3)=0.d0
        xmin(1)=b+(.5*(a-b))
        icount=0
        goto 80
      endif
      step(1)=(b-a)/2.d0
      start(1)=(b+a)/2.d0
c
      call nelmin(n,start,xmin,xsec,tcnewlo,
     &tcsec,reqmin,step,icount,pcs)
80    continue
      print*
      write(*,610)xmin(1),xmin(1)-bias
      write(*,612)
      write(*,614)pcs(1)
      write(*,616)pcs(2)
      write(*,618)pcs(3)
      write(*,620)tcnewlo
c     write(*,*)'next highest = ',tcsec
c     write(*,*)'Trials used= ',icount
90    print*,'Send output to file? (Y/N)'
      read(*,500)iopt
      if(iopt.ne.'Y'.and.iopt.ne.'N'.and.iopt.ne.'y'.and.
     &iopt.ne.'n')goto 90
      if(iopt.eq.'n'.or.iopt.eq.'N')goto 200
      write(2,510)thta(1)
```

```
      write(2,520)tau(1)
      write(2,530)spec,u
      write(2,540)sme
      write(2,550)bias
      write(2,560)
      write(2,570)s1
      write(2,580)a2
      write(2,590)r2
      write(2,600)
      write(2,610)xmin(1),xmin(1)-bias
      write(2,612)
      write(2,614)pcs(1)
      write(2,616)pcs(2)
      write(2,618)pcs(3)
      write(2,620)tcnewlo
200   continue
500   format(a1)
510   format(' Prior Mean =           ',f16.8)
520   format(' Prior Std. Dev =       ',f16.8)
530   format(1x,a5,' Specification = ',f16.8)
540   format(' Measurement Error Std. Dev. = ',f16.8)
550   format(' Measurement Error Mean (Bias = ',f16.8)
560   format(' Input Costs:')
570   format('    Iteration (S) =     ',f12.4)
580   format('    False Accept (A) =  ',f12.4)
590   format('    False Reject (R) =  ',f12.4)
600   format(' *************************************')
610   format(' Zero-Bias C1 = ',f16.8,5x,'Bias Adj C1 = ',f16.8)
612   format(' Expected Plan Costs:')
614   format('    Sampling on 1 =     ',f12.4)
616   format('    False Accept on 1 = ',f12.4)
618   format('    false Reject on 1 = ',f12.4)
620   format(' Expected Total Cost =  ',f12.4)
      return
      end
c*****************************************************
c*****************************************************
c*****************************************************
c     sub for input of all sampling paramters
c
      subroutine vardef(u,thta,tau,bias,sme,s1,a2,r2,nspc)
ccccccccccccccccccc
      character iopt
      character*5 spec
      spec='Upper'
      iflg=0
10    print*
      print*,'Enter the Specification Limit.'
      print*
      read(*,*,err=10)u
      if(iflg.eq.1)goto 90
15    print*
      print*,'Is this an Upper (1) or Lower (2) Spec?'
      print*,'Enter 1 or 2.'
```

```
      print*
      read(*,*,err=15)nspc
      if(nspc.ne.1.and.nspc.ne.2)then
        write(*,*)'Invalid Entry.  Try again.'
        goto 15
      endif
      if(nspc.eq.2)spec='Lower'
20    print*
      print*,'Enter the value of the prior distribution mean.'
      print*
      read(*,*,err=20)thta
      if(iflg.eq.1)goto 90
30    print*
      print*,'Enter the value of the prior standard deviation.'
      print*
      read(*,*,err=30)tau
      if(tau.lt.0)then
        print*
        print*,'**** Standard Deviation cannot be negative ****'
        goto 30
      endif
      if(iflg.eq.1)goto 90
40    print*
      print*,'Enter the value of the measurement error'
      print*,'distribution mean (bias).'
      print*,'Sign Convention: If the instrument reads higher '
      print*,'than the true value, this bias should be positive.'
      print*
      read(*,*,err=40)bias
      if(iflg.eq.1)goto 90
50    print*
      print*,'Enter the value of the measurement error'
      print*,'distribution standard deviation.'
      print*
      read(*,*,err=50)sme
      if(sme.lt.0)then
        print*
        print*,'**** Standard Deviation must be positive ****'
        goto 50
      endif
      if(iflg.eq.1)goto 90
60    print*
      print*,'Enter the cost associated with a single measurement'
      print*,'iteration (S).'
      print*
      read(*,*,err=60)s1
      if(s1.lt.0)then
        print*
        print*,'**** Cost must be positive ****'
        goto 60
      endif
      if(iflg.eq.1)goto 90
70    print*
      print*,'Enter the cost associated with a false acceptance of'
```

```
      print*,'a batch of product (A).'
      print*
      read(*,*,err=70)a2
      if(a2.lt.0)then
        print*
        print*,'**** Cost must be positive ****'
        goto 70
      endif
      if(iflg.eq.1)goto 90
80    print*
      print*,'Enter the cost associated with a false rejection of'
      print*,'a batch of product (R).'
      print*
      read(*,*,err=80)r2
      if(r2.lt.0)then
        print*
        print*,'**** Cost must be positive ****'
        goto 80
      endif
90    print*
      write(*,300)spec,u
      write(*,310)thta
      write(*,320)tau
      write(*,330)bias
      write(*,340)sme
      write(*,350)s1
      write(*,360)a2
      write(*,370)r2
      print*
      print*,'Is the above information correct?'
      print*,'Enter to accept or parameter # to reenter.'
      print*
      read(*,380)iopt
      print*
      if(iopt.eq.' ')goto 1000
      if(iopt.lt.'1'.or.iopt.gt.'8')then
        print*,'Invalid Entry.  Please Reenter.'
        goto 90
      endif
      iflg=1
      if(iopt.eq.'1')then
        goto 10
      endif
      if(iopt.eq.'2')then
        goto 20
      endif
      if(iopt.eq.'3')then
        goto 30
      endif
      if(iopt.eq.'4')then
        goto 40
      endif
      if(iopt.eq.'5')then
        goto 50
```

```
      endif
      if(iopt.eq.'6')then
        goto 60
      endif
      if(iopt.eq.'7')then
        goto 70
      endif
      if(iopt.eq.'8')then
        goto 80
      endif
300   format(1x,'1  ',a5' Specification Limit=      ',f10.4)
310   format(' 2  Prior Distribution Mean=        ',f10.4)
320   format(' 3  Prior Standard Deviation=       ',f10.4)
330   format(' 4  Error Distribution Mean (Bias)= ',f10.4)
340   format(' 5  Error Distribution Std. Dev.=   ',f10.4)
350   format(' 6  Iteration Cost (S)=            ',f8.2)
360   format(' 7  False Acceptance Cost (A)=      ',f8.2)
370   format(' 8  False Rejection Cost (R)=       ',f8.2)
380   format(a1)
1000  continue
      return
      end
c*****************************************************
c*****************************************************
c*****************************************************
c     subroutine to set a (-infinity) and b (infinity)
c     practical limits for the function used in order
c     to utilize gaussian quadrature
c     add arg j to define which n (not nmax) is current
c     j passes from management program
      subroutine setabx(a,b,j)
      real*8 x,ffunc,pk,trof,a,b
      dimension thta(4),tau(4)
c
      common /parms/ tau,sme,thta,u
c     reject prob
c     must set b, unknown involving c is lower limit
c     start at u, work up
c
      flg=1.
      x=u
      call func(j,flg,x,ffunc)
      pk=ffunc
      trof=ffunc
      if(ffunc.eq.0.)then
        x=x-sme
        goto 12
      endif
      x=x+sme
 10   continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.pk)then
        pk=ffunc
      endif
```

```
      if(ffunc.lt.trof)then
        trof=ffunc
        b=x
        if(trof.le.(.000001d0*pk))then
          goto 20
        endif
      endif
      x=x+sme
      goto 10
12    continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.0.)then
        b=x
        goto 20
       else
        x=x-sme
        goto 12
      endif
20    continue
c     accept prob
c     need a, unknown involving c is upper limit
c     start at u, work way down
      flg=2.
      x=u
27    continue
      call func(j,flg,x,ffunc)
      pk=ffunc
      trof=ffunc
      if(ffunc.eq.0.)then
        x=x+sme
        goto 32
      endif
      x=x-sme
30    continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.pk)then
        pk=ffunc
      endif
      if(ffunc.lt.trof)then
        trof=ffunc
        a=x
        if(trof.le.(.000001d0*pk))then
          goto 40
        endif
      endif
      x=x-sme
      goto 30
32    continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.0.)then
        a=x
        goto 40
       else
        x=x+sme
```

```
      goto 32
    endif
40  continue
    return
    end
c   ten pt gaussian quadrature
c*******************************************************
c*******************************************************
c*******************************************************
c   taken from Press, Flannery, Teukolsky and Vetterling
c   (1986), Numerical Recipes, Cambridge Univ. Press, NY.
c   This function is
c   called from fn, which serves as an intermediate
c   sub between nelmin and qgaus.
c
c
    subroutine qgaus(j,flg,a,b,ss)
    real*8 x(5),w(5),ss,xm,xr,dx,t1,t2,a,b
    data x/.1488743389d0,.4333953941d0,.6794095682d0,
   &.8650633666d0,.9739065285d0/
    data w/.2955242247d0,.2692667193d0,.2190863625d0,
   &.1494513491d0,.0666713443d0/
    xm=0.5d0*(b+a)
    xr=0.5d0*(b-a)
    ss=0.d0
    do 11 k=1,5
      dx=xr*x(k)
c
c
      call func(j,flg,xm+dx,t1)
      call func(j,flg,xm-dx,t2)
      ss=ss+w(k)*(t1+t2)
c
 11 continue
    ss=xr*ss
    return
    end
c*******************************************************
c*******************************************************
c*******************************************************
c   Intermediate sub between nelmin and qgaus
c   called from nelmin
c   sub fn takes care of the accounting in solving
c   each leg of the tc equation (using qgaus) with
c   estimates provided by nelmin.
c   should receive the array of unknowns, must pass a
c   function value back to nelmin
c
c   c is the unknown value of c1
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
    subroutine fn(c,from,pcs)
    real*8 c,from,a,b,pcs(3)
    real*8 acc,rej,prb(2),step(10)
    dimension tau(4),thta(4)
```

```
      common /costs/ s1,a2,r2,nspc
      common /parms/ tau,sme,thta,u,a,b,step
      common /ncur/ j
c
c     this is accept error for upper spec, reject for lower
c     c1 is upper limit (-inf,lower)
c
      if(c.lt.a)then
        prb(1)=100.d0
        goto 88
       else
        flg=2.
        call qgaus(j,flg,a,c,prb(1))
      endif
c
c     reject for upper spec, accept for lower
c     c1 is lower limit (inf, higher)
c     ==>a=c(1), b=...must be close enough to c1 for 10pt.
c
      if(c.gt.b)then
         prb(2)=100.d0
         goto 88
        else
         flg=1.
         call qgaus(j,flg,c,b,prb(2))
      endif
c
c
c     should now have the two integral terms
c     to plug in with costs and send back to nelmin in the
c     form of the total cost function value
c
88    continue
      if(nspc.eq.1)then
         acc=prb(1)
         rej=prb(2)
        else
         acc=prb(2)
         rej=prb(1)
      endif
      pcs(1)=s1
      pcs(2)=a2*acc
      pcs(3)=r2*rej
      from=pcs(1)+pcs(2)+pcs(3)
100   continue
      return
      end
c*****************************************************
c*****************************************************
c*****************************************************
ccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     functions obtained from mcad
c     calls sub errint from stegun and zucker
c     must bring in a x to this sub, which is obtained
```

```
c     from gaussian quadrature
c
c
      subroutine func(j,flg,x,ffunc)
c
      real*8 x,ffunc,erfarg,erfnum,erfden,ffjnk,erf,erfc
      dimension tau(4),thta(4)
      common /parms/ tau,sme,thta,u,a,b
c
c
c     next eqns common to both errors (type I and II)
c
c
      erfnum=-u*tau(j)**2.-u*sme**2.+sme**2.*
     &thta(j)+tau(j)**2.*x
      erfden=sme*(tau(j)*sqrt(tau(j)**2.+sme**2.))
      erfarg=.70710678811865475244d0*(erfnum/erfden)
      call errint(erfarg,erf,erfc)
c
      ffjnk=exp(-.5*(x-thta(j))**2./(tau(j)**2.+sme**2.))/
     &sqrt(tau(j)**2.+sme**2.)
c
c     this ffjnk for x from u to infinity (out of up spec)
c     error is in accepting batch type ii
c
      if (flg.eq.1.) go to 21
      ffunc=.19947114020071633897d0*(1.d0+erf)*ffjnk
      go to 25
c
c     this ffjnk for x from -infinity to u (in upr spec)
c     error is in rejecting batch type i
c
   21 ffunc=-.19947114020071633897d0*(erf-1.d0)*ffjnk
c
   25 continue
      return
      end

c****************************************************
c     ************************************************
c     ************************************************
c     error function evaluation (erf)
c     modified from Stegun and Zucker
c****************************************************
c
      subroutine errint (x,erf,erfc)
      real*8 an,bn,cons,c1,dn,erf,erfc,f,fn,fnm1,
     1 fnm2,four,gn,gnm1,gnm2,one,prev,pwr,rnbc,scf,sum,
     2 tn,toler,trrtpi,two,ulcf,ulps,wn,x,y,ysq
c
      data nbc,nbm/11,60/
      data one,two,four,ulps,cons/1.d0,2.d0,4.d0,1.d0,.83d0/
      data trrtpi/1.1283791670955125 74d0/
c
```

```
      rnbc=nbc
      toler=two**(-nbm)
c
c  test on zero
c
      if(x) 2,1,2
 1    erf=0.d0
      erfc=one
c
      return
c
 2    y=dabs(x)
      ysq=y**2.d0
      if(y-ulps) 3,3,4
c
c  maximum argument
c
 4    c1=two**((rnbc-one)/two)
      ulcf=cons*c1
c
c  scale factor
c
      scf=two**(c1**2.d0-rnbc)
c
c  limiting value
c
      if(y-ulcf) 10,10,11
c
 11   erf=one
      erfc=0.d0
      go to 7
c
c  method -- power series
c
 3    sum=0.d0
      dn=one
      tn=one
      pwr=two*ysq
 6    dn=dn+two
      tn=pwr*tn/dn
      sum=tn+sum
c
c  tolerance check
c
      if(tn-toler)  5,6,6
c
 5    erf=(sum+one)*trrtpi*y*dexp(-ysq)
      erfc=one-erf
c
c  negative argument
c
 7    if(x) 8,9,9
 8    erf=-erf
      erfc=two-erfc
```

```
  9  return
c
c  method-- continued fraction
c
 10  fnm2=0.d0
     gnm2=one
     fnm1=two*y
     gnm1=two*ysq+one
c
     prev=fnm1/gnm1
     wn=one
     bn=gnm1+four
 14  an=-wn*(wn+one)
     fn=bn*fnm1+an*fnm2
     gn=bn*gnm1+an*gnm2
     f=fn/gn
c
c  tolerance check
c
     if(dabs(one-(f/prev))-toler) 12,12,13
 13  if(prev-f) 17,17,18
c  both fn and gn must be tested if abs(x) .lt. .61
 17  if(gn.lt.scf) go to 16
c
c  scaling
c
 15  fn=fn/scf
     gn=gn/scf
     fnm1=fnm1/scf
     gnm1=gnm1/scf
 16  fnm2=fnm1
     gnm2=gnm1
     fnm1=fn
     gnm1=gn
     wn=wn+two
     bn=bn+four
     prev=f
     go to 14
 18  f=prev
 12  erfc=f*dexp(-ysq)*trrtpi/two
     erf=one-erfc
c
     go to 7
     end
c*****************************************************
c*****************************************************
c*****************************************************
     subroutine nelmin(n,start,xmin,xsec,ynewlo,
    &ysec,reqmin,step,icount,pclo)
c**********************************************************
c    modified from:
c    Olsson, D. M., "A Sequential Simplex Program for
c    Solving Minimization Problems," JQT, V. 6, No. 1,
c    pp. 53-57, Jan. 1974.
```

```
c    and from
c    Ho, C., "The Economic Design and Evaluation of Three
c    Variables Control Charts", Ph.D. Dissertation, O.S.U
c    July, 1992.
c************************************************************
      real*8 start(n),step(n),xmin(n),xsec(n),ynewlo,
     &ysec,reqmin,p(20,21),pstar(20),p2star(20),
     &pbar(20),y(20),z,ylo,rcoeff,ystar,ecoeff,
     &y2star,ccoeff,f,dabit,dchk,coord1,coord2,pcs(3),pclo(3)
      data rcoeff/1.0d0/,ecoeff/2.0d0/,ccoeff/0.5d0/
      kcount=icount
      icount=0
c************************************************************
c    initialization
c************************************************************
      do 60 i=1,n
        xmin(i)=0.0d0
        xsec(i)=0.0d0
   60 continue
      ynewlo=0.0d0
      ysec=0.0d0
      if (reqmin.le.0.0d0) icount=icount-1
      if (n.le.0) icount=icount-10
      if (n.gt.20) icount=icount-10
      if (icount.lt.0) return
      dabit=2.04607d-35
      bignum=1.0d30
      konvge=5
      xn=float(n)
      nn=n+1
c************************************************************
c    construction of simplex
c************************************************************
 1001 do 1 i=1,n
    1 p(i,nn)=start(i)
c
      call fn(start,f,pcs)
c
      y(nn)=f
      icount=icount+1
      do 2 j=1,n
        dchk=start(j)
        start(j)=dchk+step(j)
        do 3 i=1,n
    3   p(i,j)=start(i)
c
      call fn(start,f,pcs)
c
      y(j)=f
      icount=icount+1
    2 start(j)=dchk
c************************************************************
c    simplex construction complete
c************************************************************
```

```
c    find highest and lowest y value
c    ynewlo indicates the vertex of the
c    simplex to be replaced
c***********************************************************
 1000 ylo=y(1)
    ynewlo=ylo
    ilo=1
    ihi=1
    do 5 i=2,nn
     if (y(i).ge.ylo) go to 4
     ylo=y(i)
     ilo=i
   4  if (y(i).le.ynewlo) go to 5
     ynewlo=y(i)
     ihi=i
   5 continue
c***********************************************************
c    perform convergence checks on function
c***********************************************************
    dchk=(ynewlo+dabit)/(ylo+dabit)-1.0d0
    if (dabs(dchk).lt.reqmin) go to 900
    konvge=konvge-1
    if (konvge.ne.0) go to 2020
    konvge=5
c***********************************************************
c    check convergence of coordinate
c    only  every 5 simplex
c***********************************************************
    do 2015 i=1,n
     coord1=p(i,1)
     coord2=coord1
     do 2010 j=2,nn
      if (p(i,j).ge.coord1) go to 2005
      coord1=p(i,j)
 2005     if (p(i,j).le.coord2) go to 2010
      coord2=p(i,j)
 2010   continue
     dchk=(coord2+dabit)/(coord1+dabit)-1.0d0
     if (dabs(dchk).gt.reqmin) go to 2020
 2015 continue
    go to 900
 2020 if (icount.ge.kcount) go to 900
c***********************************************************
c    calculate pbar, the centroid of the simplex
c    vertices except thjat with y value ynewlo
c***********************************************************
    do 7 i=1,n
     z=0.0d0
     do 6 j=1,nn
      z=z+p(i,j)
   6   continue
     z=z-p(i,ihi)
   7 pbar(i)=z/float(n)
c***********************************************************
```

```
c    reflection through the centroid
c***********************************************************
      do 8 i=1,n
    8 pstar(i)=(1.0d0+rcoeff)*pbar(i)-rcoeff*p(i,ihi)
c
      call fn(pstar,f,pcs)
c       print*,'nm f= ',f
      ystar=f
      icount=icount+1
      if (ystar.ge.ylo) go to 12
      if (icount.ge.kcount) go to 19
c**********************************************************
c    successful reflection, so extension
c**********************************************************
      do 9 i=1,n
    9 p2star(i)=ecoeff*pstar(i)+(1.0d0-ecoeff)*pbar(i)
c
      call fn(p2star,f,pcs)
      y2star=f
      icount=icount+1
c**********************************************************
c    retain extension or contraction
c**********************************************************
      if (y2star.ge.ystar) go to 19
   10 do 11, i=1,n
   11 p(i,ihi)=p2star(i)
      y(ihi)=y2star
      go to 1000
c**********************************************************
c    no extension
c**********************************************************
   12 l=0
      do 13 i=1,nn
        if (y(i).gt.ystar) l=l+1
   13 continue
      if (l.gt.1) go to 19
      if (l.eq.0) go to 15
c**********************************************************
c    contraction on the reflection side of the centroid
c**********************************************************
      do 14 i=1,n
   14 p(i,ihi)=pstar(i)
      y(ihi)=ystar
c**********************************************************
c    contraction on the y(ihi) side of the centroid
c**********************************************************
   15 if (icount.ge.kcount) go to 900
      do 16 i=1,n
   16 p2star(i)=ccoeff*p(i,ihi)+(1.0d0-ccoeff)*pbar(i)
c
      call fn(p2star,f,pcs)
c
      y2star=f
      icount=icount+1
```

```
      if (y2star.lt.y(ihi)) go to 10
c*********************************************************
c     contract the whole simplex
c*********************************************************
      do 18 j=1,nn
       do 17 i=1,n
        p(i,j)=(p(i,j)+p(i,ilo))*0.5d0
   17   xmin(i)=p(i,j)
c
       call fn(xmin,f,pcs)
Y(j)=f
   18 continue
      icount=icount+nn
      if (icount.lt.kcount) go to 1000
      go to 900
c*********************************************************
c     retain reflection
c*********************************************************
   19 continue
      do 20 i=1,n
   20 p(i,ihi)=pstar(i)
      y(ihi)=ystar
      go to 1000
  900 do 23 j=1,nn
       do 22 i=1,n
   22   xmin(i)=p(i,j)
c
       call fn(xmin,f,pcs)
c
       y(j)=f
   23 continue
      ynewlo=bignum
      do 24 j=1,nn
       if (y(j).ge.ynewlo) go to 24
       ynewlo=y(j)
       pclo(1)=pcs(1)
       pclo(2)=pcs(2)
       pclo(3)=pcs(3)
       ibest=j
   24 continue
      y(ibest)=bignum
      ysec=bignum
      do 25 j=1,nn
       if (y(j).ge.ysec) go to 25
       ysec=y(j)
       isec=j
   25 continue
      do 26 i=1,n
       xmin(i)=p(i,ibest)
       xsec(i)=p(i,isec)
   26 continue
c
      return
      end
```

```
c*******************************************************
c*******************************************************
c*******************************************************
c     Main for the prior case of nmax=2.  Calls
c     nelder-mead sub which will make a call
c     to the function for minimization.
c
c
c     for the n=1 step of nmax=2, n=2 unknowns (c1l,c1h)
c
      real*8 start(10),step(10),xmin(20),xsec(20),tcnewlo,
     &tcsec,reqmin,a,b,pcs(9),xtmp
      character iopt
      character*5 spec
      dimension thta(4),tau(4)
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a,b
cccccccccccccccccc
      spec='Upper'
      call vardef(u,thta(1),tau(1),bias,sme,s1,a2,r2,nspc)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c     this to account for lower spec by symmetry, only
c     Use given spec, but find for symmetrical prior mean
      if(nspc.eq.2)then
        thta(1)=2*u-thta(1)
        spec='Lower'
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
      j=1
      call setabx(a,b,j)
      if(b.le.a)then
        tcnewlo=s1
        pcs(1)=s1
        pcs(2)=0.d0
        pcs(3)=0.d0
        pcs(4)=0.d0
        pcs(5)=0.d0
        pcs(6)=0.d0
        xmin(1)=b+(.5*(a-b))
        icount=0
        goto 85
      endif
c     starting values for c1l=a, c1h=b, c2 split diff
      start(1)=a
      start(2)=b
      start(3)=(a+b)/2.d0
      n=3
      step(1)=(b-a)/2.d0
      step(2)=step(1)
      step(3)=step(1)
      tcint=0.d0
      reqmin=.000001d0
```

```
      icount=200
      call nelmin1(n,start,xmin,xsec,tcnewlo,
     &tcsec,reqmin,step,icount,pcs)
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c     this to account for lower spec by symmetry, only
c     change prior mean back before print
c     find c1 by symmetry around the specification
      if(nspc.eq.2)then
        xtmp=xmin(1)
        xmin(1)=2*u-xmin(2)
        xmin(2)=2*u-xtmp
        xmin(3)=2*u-xmin(3)
        thta(1)=2*u-thta(1)
      endif
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
c
85    continue
c
      write(*,610)xmin(3),xmin(3)-bias
      write(*,614)xmin(1),xmin(1)-bias
      write(*,616)xmin(2),xmin(2)-bias
      write(*,618)
      write(*,620)pcs(1)
      write(*,622)pcs(2)
      write(*,624)pcs(3)
      write(*,626)pcs(4)
      write(*,628)pcs(5)
      write(*,630)pcs(6)
      write(*,650)tcnewlo
90    print*,'Send output to file? (Y/N)'
      read(*,500)iopt
      if(iopt.ne.'Y'.and.iopt.ne.'N'.and.iopt.ne.'y'.and.
     &iopt.ne.'n')goto 90
      if(iopt.eq.'n'.or.iopt.eq.'N')goto 100
      write(2,510)thta(1)
      write(2,520)tau(1)
      write(2,530)spec,u
      write(2,540)sme
      write(2,550)bias
      write(2,560)
      write(2,570)s1
      write(2,580)a2
      write(2,590)r2
      write(2,600)
      write(2,610)xmin(3),xmin(3)-bias
      write(2,614)xmin(1),xmin(1)-bias
      write(2,616)xmin(2),xmin(2)-bias
      write(2,618)
      write(2,620)pcs(1)
      write(2,622)pcs(2)
      write(2,624)pcs(3)
      write(2,626)pcs(4)
```

```
      write(2,628)pcs(5)
      write(2,630)pcs(6)
      write(2,650)tcnewlo
100   continue
500   format(a1)
510   format(' Prior Mean =          ',f16.8)
520   format(' Prior Std. Dev =      ',f16.8)
530   format(1x,a5,' Specification = ',f16.8)
540   format(' Measurement Error Std. Dev. =   ',f16.8)
550   format(' Measurement Error Mean (Bias) = ',f16.8)
560   format(' Input Costs:')
570   format('    Iteration (S) =      ',f12.4)
580   format('    False Accept (A) =  ',f12.4)
590   format('    False Reject (R) =  ',f12.4)
600   format(' ***************************************')
610   format(' Zero-Bias C2 =  ',f16.8,5x,'Bias Adj C2 =  ',f16.8)
614   format(' Zero-Bias C1l = ',f16.8,5x,'Bias Adj C1l = ',f16.8)
616   format(' Zero-Bias C1h = ',f16.8,5x,'Bias Adj C1h = ',f16.8)
618   format(' Expected Plan Costs:')
620   format('    Sampling on 1 =      ',f12.4)
622   format('    False Accept on 1 = ',f12.4)
624   format('    False Reject on 1 = ',f12.4)
626   format('    Sampling on 2 =      ',f12.4)
628   format('    False Accept on 2 = ',f12.4)
630   format('    False Reject on 2 = ',f12.4)
650   format(' Expected Total Cost =  ',f12.4)
      return
      end
c***************************************************
c***************************************************
c***************************************************
c
c     sub which steps through the integration of the
c     n=2 function from c1l to c1h
c     c1 array holds c1l and c1h
c     pcs array holds the cost components
c
      subroutine clg2(c1,tctot,pcs)
      dimension thta(4),tau(4)
      real*8 c1(3),tcint,tctot,c1l,c1h,c2
      real*8 acc,rej,a,b,bignum,pcs(9)
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a,b
      common /ncur/ j
c
      print*,'Working...'
      c1l=c1(1)
      c1h=c1(2)
      c2=c1(3)
      tctot=0.d0
      tcint=0.d0
      bignum=9.99d+55
cccccccccccccccccc
      j=1
```

```
      if(c2.gt.c1h.or.c2.lt.c1l)then
        tctot=bignum
        goto 200
      endif
      if(c1h.gt.b)then
        rej=100.d0
       else
        flg=1.
        call qgaus(j,flg,c1h,b,rej)
      endif
      if(c1l.lt.a)then
        acc=100.d0
       else
        flg=2.
        call qgaus(j,flg,a,c1l,acc)
      endif
 50   continue
      cost1=s1
      cost2a=a2*acc
      cost3r=r2*rej
 55   continue
      if(c1l.ge.c1h)then
        tcint=bignum
       else
        call nmx2(c1l,c1h,c2,pcs(4),pcs(5),pcs(6))
      endif
 60   continue
      tctot=cost1+cost2a+cost3r+pcs(4)+pcs(5)+pcs(6)
      pcs(1)=cost1
      pcs(2)=cost2a
      pcs(3)=cost3r
 200  continue
      return
      end

c****************************************************
c****************************************************
c****************************************************
c
c     sub  which steps through
c     the n=2 function between c1l and c1h
c
      subroutine nmx2(c1l,c1h,c2,tc4int,tc5int,tc6int)
      real*8 a,b,x1,c2,c1l,c1h
      real*8 tcpre,tcint,delx,x1pre,tc4pre,tc5pre,tc6pre
      real*8 acc2,rej2,tc4int,tc5int,tc6int
      dimension thta(4),tau(4)
      dimension x(2),p(2)
      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u,a,b
      common /ncur/ j
      j=2
c
c     n=1 unknown in current equation, at nmax=2 (c2)
```

```
ccccccccccccccccccc
      nmx=2
      n=1
      istp=0
      delx=(b-a)/100.d0
      tcint=0.d0
      tc4int=0.d0
      tc5int=0.d0
      tc6int=0.d0
c
      x1=c11
20    continue
      call post(x1,sme,thta(1),tau(1),thta(2),tau(2))
c
80    continue
cccccccccccccccccccccccccccc
      if(2.d0*c2-x1.le.a)then
        acc2=0.d0
          else
        call qgaus(2,2.,a,2.d0*c2-x1,acc2)
      endif
      if(b.le.2.d0*c2-x1)then
         rej2=0.d0
           else
         call qgaus(2,1.,2.d0*c2-x1,b,rej2)
      endif
      cost5a=acc2*a2
      cost6r=rej2*r2
      cost4=s1
      ynewlo=cost4+cost5a+cost6r
cvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
      if (istp.eq.0)then
         tc4pre=cost4
         tc5pre=cost5a
         tc6pre=cost6r
         tcpre=ynewlo
         x1pre=x1
         istp=1
           else
         x(1)=(x1pre-thta(1))/sqrt(tau(1)**2.+sme**2.)
         x(2)=(x1-thta(1))/sqrt(tau(1)**2.+sme**2.)
         call normal(x,p)
         prob=p(2)-p(1)
         tc4int=tc4int+prob*((cost4+tc4pre)/2.d0)
         tc5int=tc5int+prob*((cost5a+tc5pre)/2.d0)
         tc6int=tc6int+prob*((cost6r+tc6pre)/2.d0)
         tcint=tcint+prob*((ynewlo+tcpre)/2.d0)
         tcpre=ynewlo
         tc4pre=cost4
         tc5pre=cost5a
         tc6pre=cost6r
         x1pre=x1
      endif
      x1=x1+delx
```

```
c    this next chunk of code  accounts for the
c    slack between the last x1 and c1h
     if (x1.gt.c1h)then
      tcint=tcint+((c1h-x1pre)/delx)*prob*((ynewlo+tcpre)/2.d0)
      tc4int=tc4int+((c1h-x1pre)/delx)*prob*((cost4+tc4pre)/2.d0)
      tc5int=tc5int+((c1h-x1pre)/delx)*prob*((cost5a+tc5pre)/2.d0)
      tc6int=tc6int+((c1h-x1pre)/delx)*prob*((cost6r+tc6pre)/2.d0)
        goto 200
     endif
     goto 20
200  continue
2000 format(a1)
     return
     end
c    most of this taken directly  from  HO.  Not sure why
c    he's passing the sub in the call args (fn)-removed.
     subroutine nelmin1(n,start,xmin,xsec,ynewlo,
    &ysec,reqmin,step,icount,pclo)
c*********************************************************
c    modified from:
c    Olsson, D. M., "A Sequential Simplex Program for
c    Solving Minimization Problems," JQT, V. 6, No. 1,
c    pp. 53-57, Jan. 1974.
c    and from
c    Ho, C., "The Economic Design and Evaluation of Three
c    Variables Control Charts", Ph.D. Dissertation, O.S.U
c    July, 1992.
c*********************************************************
     real*8 start(n),step(n),xmin(n),xsec(n),ynewlo,
    &ysec,reqmin,p(20,21),pstar(20),p2star(20),
    &pbar(20),y(20),z,ylo,rcoeff,ystar,ecoeff,
    &y2star,ccoeff,f,dabit,dchk,coord1,coord2,pclo(9),pcs(9)
     data rcoeff/1.0d0/,ecoeff/2.0d0/,ccoeff/0.5d0/
     kcount=icount
     icount=0
c*********************************************************
c    initialization
c*********************************************************
     do 60 i=1,n
       xmin(i)=0.0d0
       xsec(i)=0.0d0
  60 continue
     ynewlo=0.0d0
     ysec=0.0d0
     if (reqmin.le.0.0d0) icount=icount-1
     if (n.le.0) icount=icount-10
     if (n.gt.20) icount=icount-10
     if (icount.lt.0)then
        print*,'iterations expired'
        return
     endif
     dabit=2.04607d-35
     bignum=1.0d30
     konvge=5
```

```
      xn=float(n)
      nn=n+1
c***********************************************************
c     construction of simplex
c***********************************************************
 1001 do 1 i=1,n
    1 p(i,nn)=start(i)
c
      call clg2(start,f,pcs)
c
      y(nn)=f
      icount=icount+1
      do 2 j=1,n
        dchk=start(j)
        start(j)=dchk+step(j)
        do 3 i=1,n
    3   p(i,j)=start(i)
c
      call clg2(start,f,pcs)
c
      y(j)=f
      icount=icount+1
    2 start(j)=dchk
c***********************************************************
c     simplex construction complete
c***********************************************************
c     find highest and lowest y value
c     ynewlo indicates the vertex of the
c     simplex to be replaced
c***********************************************************
 1000 ylo=y(1)
      ynewlo=ylo
      ilo=1
      ihi=1
      do 5 i=2,nn
        if (y(i).ge.ylo) go to 4
        ylo=y(i)
        ilo=i
    4   if (y(i).le.ynewlo) go to 5
        ynewlo=y(i)
        ihi=i
    5 continue
c***********************************************************
c     perform convergence checks on function
c***********************************************************
      dchk=(ynewlo+dabit)/(ylo+dabit)-1.0d0
      if (dabs(dchk).lt.reqmin) go to 900
      konvge=konvge-1
      if (konvge.ne.0) go to 2020
      konvge=5
c***********************************************************
c     check convergence of coordinate
c     only  every 5 simplex
c***********************************************************
```

```
    do 2015 i=1,n
      coord1=p(i,1)
      coord2=coord1
      do 2010 j=2,nn
        if (p(i,j).ge.coord1) go to 2005
        coord1=p(i,j)
 2005   if (p(i,j).le.coord2) go to 2010
        coord2=p(i,j)
 2010 continue
      dchk=(coord2+dabit)/(coord1+dabit)-1.0d0
      if (dabs(dchk).gt.reqmin) go to 2020
 2015 continue
    go to 900
 2020 if (icount.ge.kcount) go to 900
c**********************************************************
c    calculate pbar, the centroid of the simplex
c    vertices except thjat with y value ynewlo
c**********************************************************
    do 7 i=1,n
      z=0.0d0
      do 6 j=1,nn
        z=z+p(i,j)
   6  continue
      z=z-p(i,ihi)
   7 pbar(i)=z/float(n)
c**********************************************************
c    reflection through the centroid
c**********************************************************
    do 8 i=1,n
   8 pstar(i)=(1.0d0+rcoeff)*pbar(i)-rcoeff*p(i,ihi)
c
    call clg2(pstar,f,pcs)
c     print*,'nm f= ',f
    ystar=f
    icount=icount+1
    if (ystar.ge.ylo) go to 12
    if (icount.ge.kcount) go to 19
c**********************************************************
c    successful reflection, so extension
c**********************************************************
    do 9 i=1,n
   9 p2star(i)=ecoeff*pstar(i)+(1.0d0-ecoeff)*pbar(i)
c
    call clg2(p2star,f,pcs)
    y2star=f
    icount=icount+1
c**********************************************************
c    retain extension or contraction
c**********************************************************
    if (y2star.ge.ystar) go to 19
  10 do 11, i=1,n
  11 p(i,ihi)=p2star(i)
    y(ihi)=y2star
    go to 1000
```

```
c***********************************************************
c    no extension
c***********************************************************
  12 l=0
    do 13 i=1,nn
      if (y(i).gt.ystar) l=l+1
  13 continue
    if (l.gt.1) go to 19
    if (l.eq.0) go to 15
c***********************************************************
c    contraction on the reflection side of the centroid
c***********************************************************
    do 14 i=1,n
  14 p(i,ihi)=pstar(i)
    y(ihi)=ystar
c***********************************************************
c    contraction on the y(ihi) side of the centroid
c***********************************************************
  15 if (icount.ge.kcount) go to 900
    do 16 i=1,n
  16 p2star(i)=ccoeff*p(i,ihi)+(1.0d0-ccoeff)*pbar(i)
c
    call clg2(p2star,f,pcs)
c
    y2star=f
    icount=icount+1
    if (y2star.lt.y(ihi)) go to 10
c***********************************************************
c    contract the whole simplex
c***********************************************************
    do 18 j=1,nn
      do 17 i=1,n
        p(i,j)=(p(i,j)+p(i,ilo))*0.5d0
  17   xmin(i)=p(i,j)
c
      call clg2(xmin,f,pcs)
c
      Y(j)=f
  18 continue
    icount=icount+nn
    if (icount.lt.kcount) go to 1000
    go to 900
c**********************************************************
c    retain reflection
c**********************************************************
  19 continue
    do 20 i=1,n
  20 p(i,ihi)=pstar(i)
    y(ihi)=ystar
    go to 1000
 900 do 23 j=1,nn
      do 22 i=1,n
  22   xmin(i)=p(i,j)
c
```

```
      call clg2(xmin,f,pcs)
c
      y(j)=f
   23 continue
     ynewlo=bignum
     do 24 j=1,nn
      if (y(j).ge.ynewlo) go to 24
      ynewlo=y(j)
      pclo(1)=pcs(1)
      pclo(2)=pcs(2)
      pclo(3)=pcs(3)
      pclo(4)=pcs(4)
      pclo(5)=pcs(5)
      pclo(6)=pcs(6)
      pclo(7)=pcs(7)
      pclo(8)=pcs(8)
      pclo(9)=pcs(9)
      ibest=j
   24 continue
     y(ibest)=bignum
     ysec=bignum
     do 25 j=1,nn
      if (y(j).ge.ysec) go to 25
      ysec=y(j)
      isec=j
   25 continue
     do 26 i=1,n
      xmin(i)=p(i,ibest)
      xsec(i)=p(i,isec)
   26 continue
c
     return
     end
c*****************************************************
c*****************************************************
c*****************************************************
c    sub to find posterior distribution parameters
c
     subroutine post(x1,sme,thtaa,taua,thtab,taub)
     real*8 x1
     thtab=(x1/sme**2.+thtaa/taua**2.)/
    &(1./sme**2.+1./taua**2.)
     taub=sqrt(1./(1./sme**2.+1./taua**2.))
     return
     end
c*****************************************************
c*****************************************************
c*****************************************************
c    sub for input of all sampling paramters
c
      subroutine vardef(u,thta,tau,bias,sme,s1,a2,r2,nspc)
ccccccccccccccccccc
     character iopt
     character*5 spec
```

```
      spec='Upper'
      iflg=0
10    print*
      print*,'Enter the Specification Limit.'
      print*
      read(*,*,err=10)u
      if(iflg.eq.1)goto 90
15    print*
      print*,'Is this an Upper (1) or Lower (2) Spec?'
      print*,'Enter 1 or 2.'
      print*
      read(*,*,err=15)nspc
      if(nspc.ne.1.and.nspc.ne.2)then
        write(*,*)'Invalid Entry.  Try again.'
        goto 15
      endif
      if(nspc.eq.2)spec='Lower'
20    print*
      print*,'Enter the value of the prior distribution mean.'
      print*
      read(*,*,err=20)thta
      if(iflg.eq.1)goto 90
30    print*
      print*,'Enter the value of the prior standard deviation.'
      print*
      read(*,*,err=30)tau
      if(tau.lt.0)then
        print*
        print*,'**** Standard Deviation cannot be negative ****'
        goto 30
      endif
      if(iflg.eq.1)goto 90
40    print*
      print*,'Enter the value of the measurement error'
      print*,'distribution mean (bias).'
      print*,'Sign Convention: If the instrument reads higher '
      print*,'than the true value, this bias should be positive.'
      print*
      read(*,*,err=40)bias
      if(iflg.eq.1)goto 90
50    print*
      print*,'Enter the value of the measurement error'
      print*,'distribution standard deviation.'
      print*
      read(*,*,err=50)sme
      if(sme.lt.0)then
        print*
        print*,'**** Standard Deviation must be positive ****'
        goto 50
      endif
      if(iflg.eq.1)goto 90
60    print*
      print*,'Enter the cost associated with a single measurement'
      print*,'iteration (S).'
```

```
      print*
      read(*,*,err=60)s1
      if(s1.lt.0)then
        print*
        print*,'**** Cost must be positive ****'
        goto 60
      endif
      if(iflg.eq.1)goto 90
70    print*
      print*,'Enter the cost associated with a false acceptance of'
      print*,'a batch of product (A).'
      print*
      read(*,*,err=70)a2
      if(a2.lt.0)then
        print*
        print*,'**** Cost must be positive ****'
        goto 70
      endif
      if(iflg.eq.1)goto 90
80    print*
      print*,'Enter the cost associated with a false rejection of'
      print*,'a batch of product (R).'
      print*
      read(*,*,err=80)r2
      if(r2.lt.0)then
        print*
        print*,'**** Cost must be positive ****'
        goto 80
      endif
90    print*
      write(*,300)spec,u
      write(*,310)thta
      write(*,320)tau
      write(*,330)bias
      write(*,340)sme
      write(*,350)s1
      write(*,360)a2
      write(*,370)r2
      print*
      print*,'Is the above information correct?'
      print*,'Enter to accept or parameter # to reenter.'
      print*
      read(*,380)iopt
      print*
      if(iopt.eq.' ')goto 1000
      if(iopt.lt.'1'.or.iopt.gt.'8')then
        print*,'Invalid Entry.  Please Reenter.'
        goto 90
      endif
      iflg=1
      if(iopt.eq.'1')then
        goto 10
      endif
      if(iopt.eq.'2')then
```

```
        goto 20
      endif
      if(iopt.eq.'3')then
        goto 30
      endif
      if(iopt.eq.'4')then
        goto 40
      endif
      if(iopt.eq.'5')then
        goto 50
      endif
      if(iopt.eq.'6')then
        goto 60
      endif
      if(iopt.eq.'7')then
        goto 70
      endif
      if(iopt.eq.'8')then
        goto 80
      endif
300   format(1x,'1  ',a5' Specification Limit=      ',f10.4)
310   format(' 2  Prior Distribution Mean=         ',f10.4)
320   format(' 3  Prior Standard Deviation=        ',f10.4)
330   format(' 4  Error Distribution Mean (Bias)= ',f10.4)
340   format(' 5  Error Distribution Std. Dev.=  ',f10.4)
350   format(' 6  Iteration Cost (S)=            ',f8.2)
360   format(' 7  False Acceptance Cost (A)=     ',f8.2)
370   format(' 8  False Rejection Cost (R)=      ',f8.2)
380   format(a1)
1000  continue
      return
      end
c     ten pt gaussian quadrature
c*****************************************************
c*****************************************************
c*****************************************************
c     taken from Press, Flannery, Teukolsky and Vetterling
c     (1986), Numerical Recipes, Cambridge Univ. Press, NY.
c     This function is
c     called from fn, which serves as an intermediate
c     sub between nelmin and qgaus.
c
c
      subroutine qgaus(j,flg,a,b,ss)
      real*8 x(5),w(5),ss,xm,xr,dx,t1,t2,a,b
      data x/.1488743389d0,.4333953941d0,.6794095682d0,
     &.8650633666d0,.9739065285d0/
      data w/.2955242247d0,.2692667193d0,.2190863625d0,
     &.1494513491d0,.0666713443d0/
      xm=0.5d0*(b+a)
      xr=0.5d0*(b-a)
      ss=0.d0
      do 11 k=1,5
       dx=xr*x(k)
```

```
c
      call func(j,flg,xm+dx,t1)
      call func(j,flg,xm-dx,t2)
c
      ss=ss+w(k)*(t1+t2)
c
 11   continue
     ss=xr*ss
     return
     end
c*******************************************************
c*******************************************************
c*******************************************************
cccccccccccccccccccccccccccccccccccccccccccccccccccccc
c    functions obtained from mcad
c    calls sub errint from stegun and zucker
c    must bring in a x to this sub, which is obtained
c    from gaussian quadrature
c
c
     subroutine func(j,flg,x,ffunc)
c
     real*8 x,ffunc,erfarg,erfnum,erfden,ffjnk,erf,erfc
     dimension tau(4),thta(4)
     common /parms/ tau,sme,thta,u,a,b
c
c
c    next eqns common to both errors (type I and II)
c
c
     erfnum=-u*tau(j)**2.-u*sme**2.+sme**2.*
    &thta(j)+tau(j)**2.*x
     erfden=sme*(tau(j)*sqrt(tau(j)**2.+sme**2.))
     erfarg=.7071067811865475244d0*(erfnum/erfden)
     call errint(erfarg,erf,erfc)
c
     ffjnk=exp(-.5*(x-thta(j))**2./(tau(j)**2.+sme**2.))/
    &sqrt(tau(j)**2.+sme**2.)
c
c    this ffjnk for x from u to infinity (out of up spec)
c    error is in accepting batch type ii
c
     if (flg.eq.1.) go to 21
     ffunc=.19947114020071633897d0*(1.d0+erf)*ffjnk
     go to 25
c
c    this ffjnk for x from -infinity to u (in upr spec)
c    error is in rejecting batch type i
c
  21 ffunc=-.19947114020071633897d0*(erf-1.d0)*ffjnk
c
  25 continue
     return
     end
```

```
c****************************************************
c****************************************************
c****************************************************
c     subroutine to set a (-infinity) and b (infinity)
c     practical limits for the function used in order
c     to utilize gaussian quadrature
c     add arg j to define which n (not nmax) is current
c     j passes from management program
      subroutine setabx(a,b,j)
      real*8 x,ffunc,pk,trof,a,b
      dimension thta(4),tau(4)
c      common /costs/ s1,a2,r2
      common /parms/ tau,sme,thta,u
c     type i error
c     must set b, unknown involving c is lower limit
c     start at u, work up
c
      flg=1.
      x=u
      call func(j,flg,x,ffunc)
      pk=ffunc
      trof=ffunc
      if(ffunc.eq.0.)then
         x=x-sme
         goto 12
      endif
      x=x+sme
 10   continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.pk)then
         pk=ffunc
      endif
      if(ffunc.lt.trof)then
         trof=ffunc
         b=x
         if(trof.le.(.000001d0*pk))then
           goto 20
         endif
      endif
      x=x+sme
      goto 10
 12   continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.0.)then
         b=x
         goto 20
        else
         x=x-sme
         goto 12
      endif
 20   continue
c     type ii
c     need a, unknown involving c is upper limit
c     start at u, work way down
```

```
      flg=2.
      x=u
27    continue
      call func(j,flg,x,ffunc)
      pk=ffunc
      trof=ffunc
      if(ffunc.eq.0.)then
        x=x+sme
        goto 32
      endif
      x=x-sme
30    continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.pk)then
        pk=ffunc
      endif
      if(ffunc.lt.trof)then
        trof=ffunc
        a=x
        if(trof.le.(.000001d0*pk))then
          goto 40
        endif
      endif
      x=x-sme
      goto 30
32    continue
      call func(j,flg,x,ffunc)
      if(ffunc.gt.0.)then
        a=x
        goto 40
       else
        x=x+sme
        goto 32
      endif
40    continue
      return
      end
c******************************************************
c******************************************************
c******************************************************
c     sub to evaluate normal curve probabilites
c
      subroutine normal(x,p)
c
c     algorithm  as 2 j.r.statist.soc. c,(1968) v.17,no.2
c     by B. E. Cooper
c
c     computes normal areas and ordinates for an array of x values
c
      dimension x(2),p(2),q(2),z(2)
      dimension a(5)
c
      dimension connor(17)
      data connor
```

```
     1/ 8.0327350124e-17,  1.4483264644e-15,  2.4558270103e-14,
     2  3.9554295164e-13,  5.9477940136e-12,  8.3507027951e-11,
     3  1.0892221037e-9,   1.3122532964e-8,   1.4503852223e-7,
     4  1.4589169001e-6,   1.3227513228e-5,   1.0683760684e-4,
     5  7.5757575758e-4,   4.6296296296e-3,   2.380952381e-2,  0.1,
     6  3.3333333333e-1/
c
      data rrt2pi /0.3989422804/
      n=2
c
      ifault=0
      if(n) 1,1,2
    1 ifault=1
      go to 100
    2 do 31 i=1,n
      s=x(i)
      y=s*s
      if (s) 10,11,12
   11 z(i)=rrt2pi
      p(i)=0.5
      q(I)=0.5
      goto 31
c
   10 s=-s
   12 z(i)=rrt2pi*exp(-.5*y)
      if (s-2.5)13,14,14
   13 y=-.5*y
      p(i)=connor(1)
      do 15 l=2,17
   15 p(i)=p(i)*y+connor(l)
      p(i)=(p(i)*y+1.0)*x(i)*rrt2pi+0.5
      q(i)=1.0-p(i)
      goto 31
c
   14 continue
      a(2)=1.0
      a(5)=1.0
      a(3)=1.0
      y=1.0/y
      a(4)=1.0+y
      r=2.0
   19 do 17 l=1,3,2
      do 18 j=1,2
      k=l+j
      ka=7-k
      a(k)=a(ka)+a(k)*r*y
 18   continue
      r=r+1.0
   17 continue
      atst=(a(2)/a(3))-(a(5)/a(4))
      if(abs(atst).gt.(.000001))goto 19
   20 p(i)=(a(5)/a(4))*z(i)/x(i)
      if(x(i))21,11,22
   21 p(i)=-p(i)
```

```
      q(i)=1.0-p(i)
      goto 31
   22 q(i)=p(i)
      p(i)=1.0-p(i)
   31 continue
  100 continue
      return
      end

c*******************************************************
c*******************************************************
c*******************************************************
c     error function evaluation
c     taken from Stegun and Zucker
c     called from func.for
c
      subroutine errint (x,erf,erfc)
      real*8 an,bn,cons,c1,dn,erf,erfc,f,fn,fnm1,
     1  fnm2,four,gn,gnm1,gnm2,one,prev,pwr,rnbc,scf,sum,
     2  tn,toler,trrtpi,two,ulcf,ulps,wn,x,y,ysq
c
      data nbc,nbm/11,60/
      data one,two,four,ulps,cons/1.d0,2.d0,4.d0,1.d0,.83d0/
      data trrtpi/1.1283791670955125 74d0/
c
c
      rnbc=nbc
      toler=two**(-nbm)
c
c  test on zero
c
      if(x) 2,1,2
    1  erf=0.d0
      erfc=one
c
      return
c
    2  y=dabs(x)
      ysq=y**2.d0
      if(y-ulps) 3,3,4
c
c  maximum argument
c
    4  c1=two**((rnbc-one)/two)
      ulcf=cons*c1
c
c  scale factor
c
      scf=two**(c1**2.d0-rnbc)
c
c  limiting value
c
      if(y-ulcf) 10,10,11
c
```

```
 11  erf=one
     erfc=0.d0
     go to 7
c
c  method -- power series
c
 3   sum=0.d0
     dn=one
     tn=one
     pwr=two*ysq
 6   dn=dn+two
     tn=pwr*tn/dn
     sum=tn+sum
c
c  tolerance check
c
     if(tn-toler)  5,6,6
c
 5   erf=(sum+one)*trrtpi*y*dexp(-ysq)
     erfc=one-erf
c
c  negative argument
c
 7   if(x) 8,9,9
 8   erf=-erf
     erfc=two-erfc
 9   return
c
c  method-- continued fraction
c
 10  fnm2=0.d0
     gnm2=one
     fnm1=two*y
     gnm1=two*ysq+one
c
     prev=fnm1/gnm1
     wn=one
     bn=gnm1+four
 14  an=-wn*(wn+one)
     fn=bn*fnm1+an*fnm2
     gn=bn*gnm1+an*gnm2
     f=fn/gn
c
c  tolerance check
c
     if(dabs(one-(f/prev))-toler) 12,12,13
 13  if(prev-f) 17,17,18
c  both fn and gn must be tested if abs(x) .lt. .61
 17  if(gn.lt.scf) go to 16
c
c  scaling
c
 15  fn=fn/scf
     gn=gn/scf
```

```
      fnm1=fnm1/scf
      gnm1=gnm1/scf
   16 fnm2=fnm1
      gnm2=gnm1
      fnm1=fn
      gnm1=gn
      wn=wn+two
      bn=bn+four
      prev=f
      go to 14
   18 f=prev
   12 erfc=f*dexp(-ysq)*trrtpi/two
      erf=one-erfc
c
      go to 7
      end
```

APPENDIX C

TABLES OF RESULTS

Economic Runs	Cost Set 1	Nmax=1						
Common Parameters:								
	Prior Std Dev.=	1.5						
	Bias=	0.0						
	Err. Std. Dev.=	0.5						
	Upper Spec.=	102.0						
					Prior Mean			
		96.0	98.0	100.0	102.0	104.0	106.0	108.0
Cutoff(s)	C1	102.1568	101.9346	101.7123	101.4901	101.2679	101.0457	100.8235
Observation 1	Sampling Cost	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
	False Accept Cost	0.0012	0.0803	0.7623	1.0463	0.2057	0.0057	0.0000
	False Reject Cost	0.0006	0.0678	1.1164	2.7384	1.0254	0.0605	0.0006
E(Total Cost)		0.2518	0.3981	2.1287	4.0347	1.4810	0.3162	0.2506
Economic Runs	Cost Set 2	Nmax=1						
					Prior Mean			
		96.0	98.0	100.0	102.0	104.0	106.0	108.0
Cutoff(s)	C1	102.6667	102.4445	102.2222	102.0000	101.7778	101.5556	101.3333
Observation 1	Sampling Cost	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
	False Accept Cost	0.0023	0.1976	2.6032	5.1208	1.4763	0.0615	0.0004
	False Reject Cost	0.0004	0.0615	1.4764	5.1208	2.6033	0.1976	0.0022
E(Total Cost)		0.2527	0.5091	4.3296	10.4916	4.3296	0.5091	0.2527
Economic Runs	Cost Set 3	Nmax=1						
					Prior Mean			
		96.0	98.0	100.0	102.0	104.0	106.0	108.0
Cutoff(s)	C1	103.1765	102.9543	102.7321	102.5099	102.2877	102.0655	101.8432
Observation 1	Sampling Cost	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
	False Accept Cost	0.0006	0.0605	1.0253	2.7383	1.1164	0.0678	0.0006
	False Reject Cost	0.0000	0.0057	0.2057	1.0464	0.7623	0.0803	0.0012
E(Total Cost)		0.2506	0.3162	1.4810	4.0347	2.1287	0.3981	0.2518

Economic Runs	Cost Set 1	Nmax=2						
Common Parameters:								
	Prior Std Dev.=	1.5						
	Bias=	0.0						
	Err. Std. Dev.=	0.5						
	Upper Spec.=	102.0						
					Prior Mean			
		96.0	98.0	100.0	102.0	104.0	106.0	108.0
Cutoff(s)	C1,L	101.5402	101.3218	101.0993	100.8670	100.6604	100.4261	100.2493
	C1,H	103.6939	102.8218	102.5495	102.3170	102.0776	101.8760	101.6663
	C2	101.9819	101.8708	101.7597	101.6486	101.5375	101.4264	101.3152
Observation 1	Sampling Cost	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
	False Accept Cost	0.0003	0.0111	0.0663	0.0532	0.0070	0.0010	0.0000
	False Reject Cost	0.0000	0.0023	0.0906	0.4141	0.2736	0.0240	0.0003
Observation 2	Sampling Cost	0.0001	0.0042	0.0475	0.0857	0.0237	0.0011	0.0000
	False Accept Cost	0.0005	0.0384	0.4328	0.7320	0.1844	0.0071	0.0000
	False Reject Cost	0.0005	0.0496	0.7144	1.5571	0.4988	0.0247	0.0002
E(Total Cost)		0.2513	0.3556	1.6016	3.0921	1.2375	0.3071	0.2506
Economic Runs	Cost Set 2	Nmax=2						
					Prior Mean			
		96.0	98.0	100.0	102.0	104.0	106.0	108.0
Cutoff(s)	C1,L	101.8608	101.6111	101.3852	101.1781	100.8422	100.7043	100.5244
	C1,H	103.5842	103.3114	103.0854	102.8286	102.6454	102.3771	102.1714
	C2	102.3333	102.2222	102.1111	102.0000	101.8889	101.7778	101.6667
Observation 1	Sampling Cost	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
	False Accept Cost	0.0007	0.0324	0.2392	0.2746	0.0219	0.0008	0.0000
	False Reject Cost	0.0000	0.0007	0.0334	0.2660	0.2111	0.0336	0.0006
Observation 2	Sampling Cost	0.0000	0.0027	0.0412	0.0996	0.0432	0.0026	0.0000
	False Accept Cost	0.0010	0.1060	1.5560	3.4158	1.1699	0.0591	0.0005
	False Reject Cost	0.0005	0.0593	1.1576	3.4227	1.5818	0.1048	0.0011
E(Total Cost)		0.2522	0.4510	3.2774	7.7288	3.2778	0.4509	0.2522
Economic Runs	Cost Set 3	Nmax=2						
					Prior Mean			
		96.0	98.0	100.0	102.0	104.0	106.0	108.0
Cutoff(s)	C1,L	102.4125	102.1334	101.9236	101.6827	101.4867	101.2555	100.3069
	C1,H	104.2243	103.5845	103.3242	103.1328	102.8918	102.6737	102.4489
	C2	102.6847	102.5736	102.4625	102.3514	102.2403	102.1292	102.0181
Observation 1	Sampling Cost	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
	False Accept Cost	0.0004	0.0245	0.2744	0.4138	0.1047	0.0033	0.0000
	False Reject Cost	0.0000	0.0001	0.0078	0.0533	0.0693	0.0113	0.0003
Observation 2	Sampling Cost	0.0000	0.0011	0.0235	0.0857	0.0464	0.0041	0.0001
	False Accept Cost	0.0002	0.0244	0.4980	1.5574	0.7011	0.0487	0.0005
	False Reject Cost	0.0000	0.0071	0.1838	0.7319	0.4300	0.0382	0.0005
E(Total Cost)		0.2506	0.3071	1.2376	3.0921	1.6016	0.3556	0.2513

Economic Runs	Cost Set 1	Nmax=3						
Common Parameters:								
	Prior Std Dev.=	1.5						
	Bias=	0.0						
	Err. Std. Dev.=	0.5						
	Upper Spec.=	102.0						
					Prior Mean			
		96.0	98.0	100.0	102.0	104.0	106.0	108.0
Cutoff(s)	C1,L	101.4470	101.2393	100.9831	100.7645	100.5467	100.4284	100.4594
	C1,H	103.7258	103.0602	102.8136	102.4760	102.2645	102.0404	102.1076
	C2,L	101.6380	101.5229	101.3949	101.3123	101.1893	101.0690	101.1737
	C2,H	102.4185	103.0580	102.1425	102.0601	101.9678	101.8484	101.5743
	C3	101.9378	101.8638	101.7897	101.7156	101.6415	101.5675	101.4934
Observation 1	Sampling Cost	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
	False Accept Cost	0.0002	0.0078	0.0365	0.0289	0.0033	0.0001	0.0000
	False Reject Cost	0.0000	0.0006	0.0280	0.2377	0.1636	0.0170	0.0001
Observation 2	Sampling Cost	0.0001	0.0049	0.0574	0.1003	0.0304	0.0015	0.0000
	False Accept Cost	0.0001	0.0056	0.0475	0.0803	0.0147	0.0004	0.0000
	False Reject Cost	0.0001	0.0000	0.1865	0.4168	0.1597	0.0099	0.0002
Observation 3	Sampling Cost	0.0000	0.0024	0.0242	0.0451	0.0139	0.0006	0.0000
	False Accept Cost	0.0003	0.0252	0.3060	0.5445	0.1556	0.0067	0.0000
	False Reject Cost	0.0004	0.0424	0.4513	0.9781	0.3276	0.0157	0.0001
E(Total Cost)		0.2511	0.3389	1.3874	2.6817	1.1188	0.3019	0.2505
Economic Runs	Cost Set 2	Nmax=3						
					Prior Mean			
		96.0	98.0	100.0	102.0	104.0	106.0	108.0
Cutoff(s)	C1,L	101.6884	101.4419	101.1947	100.9869	100.7962	100.7360	100.6183
	C1,H	103.7187	103.4071	103.3480	103.0216	102.8088	102.5637	102.3553
	C2,L	101.8720	101.7575	101.6657	101.5739	101.3399	101.1520	100.7061
	C2,H	102.8246	102.7353	102.6148	102.4666	102.3471	102.2192	102.1004
	C3	102.2222	102.1481	102.0741	102.0000	101.9259	101.8519	101.7778
Obsevation 1	Sampling Cost	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
	False Accept Cost	0.0004	0.0178	0.1047	0.1041	0.0166	0.0090	0.0000
	False Reject Cost	0.0000	0.0004	0.0066	0.0993	0.1030	0.0175	0.0004
Observation 2	Sampling Cost	0.0000	0.0036	0.0520	0.1200	0.0511	0.0036	0.0000
	False Accept Cost	0.0002	0.0189	0.2440	0.4530	0.0452	0.0007	0.0000
	False Reject Cost	0.0000	0.0021	0.0681	0.3508	0.2268	0.0217	0.0003
Observation 3	Sampling Cost	0.0000	0.0015	0.0232	0.0560	0.0242	0.0015	0.0000
	False Accept Cost	0.0008	0.0750	1.0993	2.4866	0.9622	0.0535	0.0005
	False Reject Cost	0.0005	0.0527	0.9472	2.5777	1.1161	0.0727	0.0007
E(Total Cost)		0.2519	0.0422	2.7951	6.4975	2.7952	0.4220	0.2519

Statistical Runs		Cost Set 1	Nmax=1		
Common Parameters:					
	Prior Mean-		102.00		
	Prior Std Dev.=		1.50		
	Bias=		0.00		
	Err. Std. Dev.=		0.50		
	Upper Spec.=		102.00		
	S=		0.25		
	A=		100.00		
	R=		20.00		
All Risk Pairs					
			E(Total Cost)	P(False Accept)	P(False Reject)
		Economic	4.0624	0.0108	0.1364
		101.0/102.0	4.5452	0.0254	0.0877
Lower/Upper		101.0/102.5	5.6544	0.0418	0.0614
Indifference Limits		101.0/103.0	7.8412	0.0687	0.0360
		101.5/102.0	9.6556	0.0891	0.0248
		101.5/102.5	9.6556	0.0891	0.0248
		101.5/103.0	12.4716	0.1194	0.0139
		102.0/102.5	19.1360	0.1881	0.0036
		102.0/103.0	19.1360	0.1881	0.0036

Statistical Runs		Cost Set 1	Nmax=2								
Common Parameters:											
	Prior Mean-		102.00								
	Prior Std Dev.=		1.50								
	Bias=		0.00								
	Err. Std. Dev.=		0.50								
	Upper Spec.=		102.00								
	S=		0.25								
	A=		100.00								
	R=		20.00								
				E(Total Cost)			P(False Accept)			P(False Reject)	
		alpha/beta	0.05/0.05	0.05/0.01	0.1/0.01	0.05/0.05	0.05/0.01	0.1/0.01	0.05/0.05	0.05/0.01	0.1/0.01
		Economic	3.1200	3.1200	3.1200	0.0075	0.0075	0.0075	0.1015	0.1015	0.1015
		101.0/102.0	3.3157	3.3315	3.3447	0.0031	0.0030	0.0030	0.1334	0.1337	0.1351
Lower/Upper		101.0/102.5	3.2887	3.2112	3.2044	0.0138	0.0125	0.0120	0.0799	0.0818	0.0844
Indifference Limits		101.0/103.0	4.8845	4.6246	4.5446	0.0380	0.0349	0.0336	0.0395	0.0412	0.0442
		101.5/102.0	3.2791	3.3012	3.2845	0.0129	0.0129	0.0129	0.0791	0.0791	0.0792
		101.5/102.5	4.7373	4.7186	4.6875	0.0365	0.0361	0.0357	0.0373	0.0374	0.0384
		101.5/103.0	8.499	8.2628	8.1206	0.0791	0.0765	0.0748	0.0140	0.0144	0.0158
		102.0/102.5	8.4675	8.4987	8.4846	0.0779	0.0779	0.0779	0.0779	0.0779	0.0779
		102.0/103.0	13.3176	13.2928	13.2018	0.1292	0.1287	0.1279	0.0030	0.0030	0.0032

Statistical Runs		Cost Set 1	Nmax=3								
Common Parameters:											
	Prior Mean-		102.00								
	Prior Std Dev.=		1.50								
	Bias=		0.00								
	Err. Std. Dev.=		0.50								
	Upper Spec.=		102.00								
	S=		0.25								
	A=		100.00								
	R=		20.00								
				E(Total Cost)			P(False Accept)			P(False Reject)	
		alpha/beta	0.05/0.05	0.05/0.01	0.1/0.01	0.05/0.05	0.05/0.01	0.1/0.01	0.05/0.05	0.05/0.01	0.1/0.01
		Economic	2.6567	2.6567	2.6567	0.0061	0.0061	0.0061	0.0826	0.0826	0.0826
		101.0/102.0	3.1654	3.2049	3.2419	0.0015	0.0013	0.0013	0.1318	0.1333	0.1362
Lower/Upper		101.0/102.5	2.8437	2.6822	2.7191	0.01	0.0075	0.007	0.0751	0.0786	0.0837
Indifference Limits		101.0/103.0	4.6095	4.0905	4.0012	0.0361	0.0301	0.0286	0.0343	0.0372	0.0409
		101.5/102.0	2.7632	2.8061	2.7729	0.0079	0.0079	0.0078	0.0737	0.0737	0.0737
		101.5/102.5	4.1893	4.0809	3.9869	0.0317	0.0302	0.029	0.0315	0.0322	0.0341
		101.5/103.0	8.1901	7.7248	7.4275	0.0765	0.0715	0.0682	0.0103	0.0107	0.0129
		102.0/102.5	8.0922	8.1473	8.1074	0.0743	0.0743	0.0741	0.0081	0.0081	0.0082
		102.0/103.0	13.3454	13.2093	12.9252	0.1293	0.1276	0.1248	0.0017	0.0018	0.0022

APPENDIX D

COMPUTER SIMULATION PROGRAMS FOR TESTING OF THE MODELS

(FORTRAN Code Listings)

```
c     ****************************************************
c     ****************************************************
c     ****************************************************
c     simulation pgm for nmax=1, disposition by stat
c     and by an economic plan (with cutoffs provided
c     by user).
c
c     mu0=prior mean
c     sd0=prior std dev
c     sdme=measurement error std dev
c     u=upper spec
c     s1=iteration cost
c     a2=false acceptance cost
c     r2=false rejectio cost
c     c1=economic cutoff for first observation
c     n=number of trials (batches)
c     alpha=type I error risk (stat)
c     beta=type II error risk (stat)
c     the0=lower indifference limit (stat)
c     the1=upper indifference limit (stat)
c
      integer rej1,acc1,stacc1,strej1,stacc2,strej2
      real mu0
      n=50000
ccxxxxxxxxxxxxxxxx
      mu0=102.
      sd0=1.5
      sdme=.5
      u=102.
      s1=.25
      a2=100.
      r2=20.
      c1=101.49010705
cxxxxxxxxxxxxxxxxxx
      alpha=.1
      beta=.01
      the0=102.
      the1=103.
      dlna=log((1.0-beta)/alpha)
      dlnb=log(beta/(1.0-alpha))
c      sumx=0.

ccxxxxxxxxxxxxxxxxx
      write(4,*)'Trials= ',n
      write(4,*)'Upper Specification= ',u
      write(4,*)'theta0= ',mu0
      write(4,*)'prior s= ',sd0,' me s= ',sdme
      write(4,*)'c1= ',c1
      write(4,*)'s1= ',s1,' a2= ',a2,' r2= ',r2
      write(4,*)'alpha= ',alpha,' beta= ',beta
      write(4,*)'lwr indif= ',the0,' upr indif= ',the1
      write(3,305)
23    continue
      print*,'seed? 999 to quit'
```

```
      read(*,300)isd
      if(isd.eq.999)goto 400
c      isd=10
      igd=0
      ingd=0
      call seed(isd)
      tctot=0.
      sttot=0.
      rej1=0
      acc1=0
      strej1=0
      strej2=0
      stacc1=0
      stacc2=0
      jr1=0
      ja1=0
      jast1=0
      jast2=0
      jrst1=0
      jrst2=0
      nost=0
      match=0
c
c
c
c
      do 200 k=1,n
      jdec=0
      jstat=0
      tcrun=s1
      strun=s1
      cst=0.
      st=0.
40    call random(rv1)
      if(rv1.eq.0.)goto 40
50    call random(rv2)
      if(rv2.eq.0.)goto 50
c      print*,'rv1= ',rv1,'rv2= ',rv2
      z=sqrt(-2.*log(rv1))*cos(2.*3.1415927*rv2)
      zme=sqrt(-2.*log(rv1))*sin(2.*3.1415927*rv2)
      yact=z*sd0+mu0
      y1obs=yact+(zme*sdme)
cxxxxxxxxxxxxxxxxxx
c      sumx=sumx+y1obs
      dllr=((the1-the0)/sdme**2.0)*y1obs+(1/(2.0*sdme**2.0))*
     \(the0**2.0-the1**2.0)
      if(dllr.lt.dlnb)then
c        print*
c        write(*,248)dllr
c        write(*,249)dlnb
c        write(*,*)'********  Accept batch  ********'
c        print*
        stacc1=stacc1+1
        jstat=1
```

```
      goto 80
      elseif(dllr.ge.dlna)then
c        print*
c        write(*,250)dllr
c        write(*,251)dlna
c        write(*,*)'******** Reject batch ********'
c        print*
        strej1=strej1+1
        jstat=2
        goto 80
     endif
    st=0.
    nost=nost+1
    if(dllr.gt.1.)then
       strej2=strej2+1
        if(yact.le.u)then
           jrst2=jrst2+1
        endif
     else
       stacc2=stacc2+1
        if(yact.gt.u)then
           jast2=jast2+1
        endif
    endif
c    write(*,*)'no statistical decision on 1'
cxxxxxxxxxxxxxxxxxxx
80   continue
    if(y1obs.le.c1)then
       acc1=acc1+1
       jdec=1
       goto 100
    endif
    if(y1obs.gt.c1)then
       rej1=rej1+1
       jdec=2
       goto 100
    endif
 100  continue
    if(yact.le.u)then
        igd=igd+1
        if(jdec.eq.2)then
           jr1=jr1+1
           cst=r2
c            write(*,*)'false econ reject'
        endif
        if(jstat.eq.2)then
           jrst1=jrst1+1
           st=r2
        endif
       else
        ingd=ingd+1
        if(jdec.eq.1)then
           ja1=ja1+1
           cst=a2
```

```
c          write(*,*)'false econ accept'
        endif
        if(jstat.eq.1)then
          jast1=jast1+1
           st=a2
        endif
      endif
      if(jdec.eq.jstat)match=match+1
      tcrun=tcrun+cst
      strun=strun+st
      tctot=tctot+tcrun
      sttot=sttot+strun
200   continue
c
c
      tcave=tctot/n
      stave=sttot/(n-nost)
cc     write(*,*)'RUN #',m
c     write(*,*)'x1= ',y1obs
      write(4,*)
      write(4,*)'*****'
      write(4,*)'match= ',match,' %= ',real(match)/real(n)
      print*,'*****'
      write(4,320)
      write(4,325)stacc1,acc1
      write(4,330)real(stacc1)/real(n),real(acc1)/real(n)
      write(4,335)strej1,rej1
      write(4,340)real(strej1)/real(n),real(rej1)/real(n)
c     write(*,345)nost,real(nost)/real(n))
c     write(*,385)nost,real(nost)/real(n),real(nost*s1)/real(n)
      write(4,385)nost,real(nost)/real(n)
      write(4,388)stacc2,strej2
      write(4,389)jast2,jrst2,real(jast2*a2)/real(n),
     &real(jrst2*r2)/real(n)
      write(4,350)jast1,ja1
      write(4,355)real(jast1)/real(n),real(ja1)/real(n)
      write(4,357)real(jast1+jast2)/real(n)
      write(4,360)jrst1,jr1
      write(4,365)real(jrst1)/real(n),real(jr1)/real(n)
      write(4,367)real(jrst1+jrst2)/real(n)
      print*,'*****'
      write(4,370)stave,tcave
c     this next stat ave cost includes all decisions on 1 and
c     truncated decisions,too
      stall=(sttot+real(jast2*a2)+real(jrst2*r2))/real(n)
      write(4,374)stall
      write(4,*)'tot act above u= ',ingd,' ',real(ingd)/real(n)
      write(4,*)'tot act below u= ',igd,' ',real(igd)/real(n)
      write(4,*)
      cr1=(real(jr1)/real(n))*r2
      ca1=(real(ja1)/(n))*a2
      write(4,*)'false econ rej on 1= ',real(jr1)/real(n),
     &' cost= ',cr1
      write(4,*)'false econ acc on 1= ',real(ja1)/real(n),
```

```
   &' cost= ',ca1
    write(4,*)
    write(3,310)s1,ca1,cr1,tcave
248   format(' Ln of likelihood ratio, ',f14.4,', less than')
249   format(' ln of B, ',f8.4,'.')
250   format(' Ln of likelihood ratio, ',f14.4,', greater than')
251   format(' ln of A, ',f8.4,'.')
300  format(i3)
305  format(5x,'tc(1)',7x,'tc(a2)',7x,'tc(r3)',7x,'tcost')
310  format(1x,f4.2,5x,f11.8,5x,f11.8,5x,f11.8)
    goto 23
320  format(25x,'Stat',10x,'Econ')
325  format(' Count Accept',11x,i5,8x,i5)
330  format(' Percent Accept',8x,f6.4,7x,f6.4)
335  format(' Count Reject',11x,i5,8x,i5)
340  format(' Percent Accept',8x,f6.4,7x,f6.4)
c345  format(' No Decision',9x,i5,'/',f6.4)
350  format(' Count False Acc(1)',5x,i5,8x,i5)
355  format(' Perc False Acc(1)',5x,f6.4,7x,f6.4)
357  format(' Perc False Acc(all)',3x,f6.4)
360  format(' Count False Rej(1)',5x,i5,8x,i5)
365  format(' Perc False Rej(1)',5x,f6.4,7x,f6.4)
367  format(' Perc False Rej(all)',3x,f6.4)
370  format(' Ave Tot Cost (in 1)',3x,f8.4,5x,f8.4)
374  format(' Ave Stat Cost (w/Trunc)',f8.4)
c385  format(' No Decision',16x,i5,5x,f6.4,3x,'It cost',4x,f6.4)
385  format(' No Decision',16x,i5,5x,f6.4)
388  format('      acc/rej            ',i5,'/',i5)
389  format('      false acc/false rej  ',i5,'/',i5,5x,'Cost',
   &2x,f8.4,'/',f8.4)
400      return
    end
```

```
c     *********************************************
c     *********************************************
c     *********************************************
c     simulation pgm for nmax=2, disposition by stat
c     and by an economic plan (with cutoffs provided
c     by user).
c
c     mu0=prior mean
c     sd0=prior std dev
c     sdme=measurement error std dev
c     u=upper spec
c     s1=iteration cost
c     a2=false acceptance cost
c     r2=false rejectio cost
c     c1l,c1h,c2=economic cutoffs
c     n=number of trials (batches)
c     alpha=type I error risk (stat)
c     beta=type II error risk (stat)
c     the0=lower indifference limit (stat)
c     the1=upper indifference limit (stat)
c
      integer rej1,acc1,rej2,acc2,rej3,acc3,stflga,stflgr
      integer stacc(4),strej(4)
      real mu0
      dimension yobs(3)
c     50000 trials
      n=50000
ccxxxxxxxxxxxxxxxx
      mu0=102.
      sd0=1.5
      sdme=.5
      u=102.
      s1=.25
      a2=100.
      r2=20.
      c1l=100.86696886
      c1h=102.31702894
      c2=101.64859854
c     c2l=101.3026704895847
c     c2h=102.0779310721692
c     c3 =101.7155998779199
cxxxxxxxxxxxxxxxxxx
c     statistical parameters
c     dlna and dlnb are the decision limits for SPRT
c
      alpha=.1
      beta=.01
      the0=102.
      the1=103.
      dlna=log((1.0-beta)/alpha)
      dlnb=log(beta/(1.0-alpha))
c     print*,'dlna= ',dlna,' dlnb= ',dlnb
ccxxxxxxxxxxxxxxxxxx
      write(4,*)'Trials= ',n
```

```
      write(4,*)'Upper Specification= ',u
      write(4,*)'theta0= ',mu0
      write(4,*)'prior s= ',sd0,' me s= ',sdme
      write(4,*)'c2= ',c2
      write(4,*)'c1l= ',c1l,' c1h= ',c1h
      write(4,*)'s1= ',s1,' a2= ',a2,' r2= ',r2
      write(4,*)'alpha= ',alpha,' beta= ',beta
      write(4,*)'lwr indif= ',the0,' upr indif= ',the1
      write(3,305)
23    continue
c     accept the seed from the user. run 5000 trials for each
c     seed
c     initiate all counters
c
      print*,'seed? 999 to quit'
      read(*,300)isd
      if(isd.eq.999)goto 600
      igd=0
      ingd=0
      call seed(isd)
      tctot=0.
      sttot=0.
      rej1=0
      acc1=0
      rej2=0
      acc2=0
      rej3=0
      acc3=0
      jr1=0
      jr2=0
      jr3=0
      ja1=0
      ja2=0
      ja3=0
      jrst1=0
      jrst2=0
      jrst3=0
      jrst4=0
      jast1=0
      jast2=0
      jast3=0
      jast4=0
      nost=0
      do 30 j=1,4
         stacc(j)=0
         strej(j)=0
30    continue
      matcha=0
      matchr=0
c     matcha1=0
c     matcha2=0
c     matcha3=0
c     matchr1=0
c     matchr2=0
```

```
c     matchr3=0
c
c
c
c
      do 200 k=1,n
      strun=0.
      sumx=0.
      jdec=0
      cst=0.
      st=0.
      stflgr=0
      stflga=0
c     generate random variables from the prior distribution
c     this is the actual batch value
40      call random(rv1)
      if(rv1.eq.0.)goto 40
50      call random(rv2)
      if(rv2.eq.0.)goto 50
c      print*,'rv1= ',rv1,'rv2= ',rv2
      z=sqrt(-2.*log(rv1))*cos(2.*3.1415927*rv2)
      zme=sqrt(-2.*log(rv1))*sin(2.*3.1415927*rv2)
      yact=z*sd0+mu0
c     generate measurement error components from the me dist
c     use with actual value to generate observation values 1-3
c
      do 60 j=1,2
53      call random(rv1)
      if(rv1.eq.0.)goto 53
56      call random(rv2)
      if(rv1.eq.0.)goto 56
      zme=sqrt(-2.*log(rv1))*sin(2.*3.1415927*rv2)
      yobs(j)=yact+(zme*sdme)
60    continue
cxxxxxxxxxxxxxxxxxx
c     statistical case:  find the log of likelihood ratio at
c     each observation
c     determine at each stage if disposition can be made
c
      do 70 i=1,2
      strun=strun+s1
      sumx=sumx+yobs(i)
      dllr=((the1-the0)/sdme**2.0)*sumx+(i/(2.0*sdme**2.0))*
     &(the0**2.0-the1**2.0)
      if(dllr.lt.dlnb)then
c         print*
c         write(*,248)dllr
c         write(*,249)dlnb
c         write(*,*)'********  Accept batch  ********'
c         print*
        stflga=i
        stacc(i)=stacc(i)+1
c        jstat=i
        goto 80
```

```
        elseif(dllr.ge.dlna)then
c          print*
c          write(*,250)dllr
c          write(*,251)dlna
c          write(*,*)'********  Reject batch  ********'
c          print*
         stflgr=i
         strej(i)=strej(i)+1
c          jstat=i
         goto 80
      endif
c     next code to use statistical (truncated sprt) rule of thumb
c     in the case that no decision is reached in nmax
70    continue
      nost=nost+1
      if(dllr.gt.0.)then
        strej(4)=strej(4)+1
         if(yact.le.u)then
           jrst4=jrst4+1
         endif
       else
        stacc(4)=stacc(4)+1
         if(yact.gt.u)then
           jast4=jast4+1
         endif
      endif
cxxxxxxxxxxxxxxxxxxx
c     econ case:  compare observation mean to cutoff values
c     at each stage
80    continue
      if(yobs(1).le.c1l)then
        acc1=acc1+1
        jdec=1
        tcrun=s1
        goto 100
      endif
      if(yobs(1).gt.c1h)then
        rej1=rej1+1
        jdec=2
        tcrun=s1
        goto 100
      endif
c      if((yobs(2)+yobs(1))/2.le.c2l)then
      if((yobs(2)+yobs(1))/2.le.c2)then
        acc2=acc2+1
        jdec=3
        tcrun=2.*s1
        goto 100
      endif
c      if((yobs(2)+yobs(1))/2.gt.c2h)then
      if((yobs(2)+yobs(1))/2.gt.c2)then
        rej2=rej2+1
        jdec=4
        tcrun=2.*s1
```

```
      goto 100
     endif
c    determine if the correct disposition was made
c    assess false disposition costs
100  continue
     if(yact.le.u)then
       igd=igd+1
       if(jdec.eq.2)then
         jr1=jr1+1
         cst=r2
c          write(*,*)'false econ reject'
       endif
       if(stflgr.eq.1)then
         jrst1=jrst1+1
         st=r2
       endif
       if(jdec.eq.4)then
         jr2=jr2+1
         cst=r2
       endif
       if(stflgr.eq.2)then
        jrst2=jrst2+1
         st=r2
       endif
       if(jdec.eq.6)then
         jr3=jr3+1
         cst=r2
       endif
       if(stflgr.eq.3)then
        jrst3=jrst3+1
         st=r2
       endif
      else
       ingd=ingd+1
       if(jdec.eq.1)then
         ja1=ja1+1
         cst=a2
c          write(*,*)'false econ accept'
       endif
c    assign economic costs to the statistical case -
c    determine if correct disposition, if not add cost
       if(stflga.eq.1)then
        jast1=jast1+1
         st=a2
       endif
       if(jdec.eq.3)then
        ja2=ja2+1
        cst=a2
       endif
       if(stflga.eq.2)then
        jast2=jast2+1
        st=a2
       endif
       if(jdec.eq.5)then
```

```
          ja3=ja3+1
          cst=a2
        endif
        if(stflga.eq.3)then
          jast3=jast3+1
          st=a2
        endif
      endif
c     find if the econ and stat cases match
      if(jdec.eq.1.and.stflga.gt.0)matcha=matcha+1
      if(jdec.eq.3.and.stflga.gt.0)matcha=matcha+1
      if(jdec.eq.5.and.stflga.gt.0)matcha=matcha+1
      if(jdec.eq.2.and.stflgr.gt.0)matchr=matchr+1
      if(jdec.eq.4.and.stflgr.gt.0)matchr=matchr+1
      if(jdec.eq.6.and.stflgr.gt.0)matchr=matchr+1
      if(stflga.eq.0.and.stflgr.eq.0)then
c         nost=nost+1
        strun=2*s1
      endif
      tcrun=tcrun+cst
c      print*,'tcrun= ',tcrun
      strun=strun+st
      tctot=tctot+tcrun
      sttot=sttot+strun
200   continue
c
c
      tcave=tctot/n
      stave=sttot/(n-nost)
cc      write(*,*)'RUN #',m
c      write(*,*)'x1= ',y1obs
c      print*,('                 1           2           3')
c      print*,'Match Acc = ',matcha1,'  ',matcha2,'  ',matcha3
c      print*,'Match Rej = ',matchr1,'  ',matchr2,'  ',matchr3
      write(4,*)
      write(4,*)'*****'
      write(4,*)'Match Acc = ',matcha,'     ',real(matcha)/real(n)
      write(4,*)'Match Rej = ',matchr,'     ',real(matchr)/real(n)
c      write(4,*)'*****'
      write(4,315)
      write(4,320)
      write(4,325)stacc(1),real(stacc(1))/real(n),acc1,
     &real(acc1)/real(n)
      write(4,345)stacc(2),real(stacc(2))/real(n),acc2,
     &real(acc2)/real(n)
c      write(4,365)stacc(3),real(stacc(3))/real(n),acc3,
c     &real(acc3)/real(n)
      write(4,335)strej(1),real(strej(1))/real(n),rej1,
     &real(acc1)/real(n)
      write(4,355)strej(2),real(strej(2))/real(n),rej2,
     &real(acc2)/real(n)
c      write(4,375)strej(3),real(strej(3))/real(n),rej3,
c     &real(acc3)/real(n)
c      write(4,385)nost,real(nost)/real(n),real(nost*s1)/real(n)
```

```
      write(4,385)nost,real(nost)/real(n)
      write(4,388)stacc(4),strej(4)
      write(4,389)jast4,jrst4,real(jast4*a2)/real(n),
     &real(jrst4*r2)/real(n)
      write(4,390)jast1,real(jast1)/real(n),ja1,
     &real(ja1)/real(n)
      write(4,410)jast2,real(jast2)/real(n),ja2,
     &real(ja2)/real(n)
c      write(4,430)jast3,real(jast3)/real(n),ja3,
c     &real(ja3)/real(n)
      write(4,442)jast1+jast2,real(jast1+jast2)/
     &real(n),ja1+ja2,real(ja1+ja2)/real(n)
      write(4,443)jast1+jast2+jast4,real(jast1+jast2+
     &jast4)/real(n),ja1+ja2,real(ja1+ja2)/real(n)
      write(4,400)jrst1,real(jrst1)/real(n),jr1,
     &real(jr1)/real(n)
      write(4,420)jrst2,real(jrst2)/real(n),jr2,
     &real(jr2)/real(n)
c      write(4,440)jrst3,real(jrst3)/real(n),jr3,
c     &real(jr3)/real(n)
      write(4,444)jrst1+jrst2,real(jrst1+jrst2)/
     &real(n),jr1+jr2,real(jr1+jr2)/real(n)
      write(4,445)jrst1+jrst2+jrst4,real(jrst1+jrst2+
     &jrst4)/real(n),jr1+jr2,real(jr1+jr2)/real(n)
      print*,'*****'
      write(4,450)stave,tcave
      stall=(sttot+real(jast4*a2)+real(jrst4*r2))/real(n)
      write(4,455)stall
      write(4,*)'tot act above u= ',ingd,' ',real(ingd)/real(n)
      write(4,*)'tot act below u= ',igd,' ',real(igd)/real(n)
c      write(4,*)
      cr1=(real(jr1)/real(n))*r2
      ca1=(real(ja1)/(n))*a2
c      write(4,*)'false econ rej on 1= ',real(jr1)/real(n),
c     &' cost= ',cr1
c      write(4,*)'false econ acc on 1= ',real(ja1)/real(n),
c     &' cost= ',ca1
c      write(4,*)
      write(3,310)s1,ca1,cr1,tcave
248   format(' Ln of likelihood ratio, ',f14.4,', less than')
249   format(' ln of B, ',f8.4,'.')
250   format(' Ln of likelihood ratio, ',f14.4,', greater than')
251   format(' ln of A, ',f8.4,'.')
300   format(i3)
305   format(5x,'tc(1)',7x,'tc(a2)',7x,'tc(r3)',7x,'tcost')
310   format(1x,f4.2,5x,f11.8,5x,f11.8,5x,f11.8)
      goto 23
315   format(35x,'Stat',15x,'Econ')
320   format(30x,'Count',5x,'%',10x,'Count',5x,'%')
325   format(' Count/% Accept on 1',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
335   format(' Count/% Reject on 1',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
345   format(' Count/% Accept on 2',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
355   format(' Count/% Reject on 2',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
365   format(' Count/% Accept on 3',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
```

```
375  format(' Count/% Reject on 3',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
c385  format(' No Decision',16x,i5,5x,f6.4,3x,'It cost',4x,f6.4)
385  format(' No Decision',16x,i5,5x,f6.4)
388  format('      acc/rej             ',i5,'/',i5)
389  format('      false acc/false rej  ',i5,'/',i5,5x,'Cost',
   &2x,f8.4,'/',f8.4)
390  format(' Count/% False Accept on 1',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
400  format(' Count/% False Reject on 1',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
410  format(' Count/% False Accept on 2',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
420  format(' Count/% False Reject on 2',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
430  format(' Count/% False Accept on 3',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
440  format(' Count/% False Reject on 3',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
442  format(' Count/% False Accept (2) ',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
443  format(' Count/% False Accept tot ',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
444  format(' Count/% False Reject (2) ',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
445  format(' Count/% False Reject tot ',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
450  format(' Ave Tot Cost (in 2)',10x,f8.4,10x,f8.4)
455  format(' Ave Stat Cost (w/ Trunc)     ',f8.4)
600    return
     end
```

```
c     ******************************************
c     ******************************************
c     ******************************************
c     simulation pgm for nmax=3, disposition by stat
c     and by an economic plan (with cutoffs provided
c     by user).
c
c     mu0=prior mean
c     sd0=prior std dev
c     sdme=measurement error std dev
c     u=upper spec
c     s1=iteration cost
c     a2=false acceptance cost
c     r2=false rejectio cost
c     c1l,c1h,c2l,c2h,c3=economic cutoffs
c     n=number of trials (batches)
c     alpha=type I error risk (stat)
c     beta=type II error risk (stat)
c     the0=lower indifference limit (stat)
c     the1=upper indifference limit (stat)
      integer rej1,acc1,rej2,acc2,rej3,acc3,stflga,stflgr
      integer stacc(4),strej(4)
      real mu0
      dimension yobs(3)
c     50000 trials
      n=50000
ccxxxxxxxxxxxxxxxx
      mu0=102.
      sd0=1.5
      sdme=.5
      u=102.
      s1=.25
      a2=100.
      r2=20.
      c1l=100.76454886
      c1h=102.47599537
      c2l=101.31232719
      c2h=102.06013412
      c3 =101.71559988
cxxxxxxxxxxxxxxxxx
c     statistical parameters
c     dlna and dlnb are the decision limits for SPRT
c
      alpha=.1
      beta=.01
      the0=102.
      the1=103.
      dlna=log((1.0-beta)/alpha)
      dlnb=log(beta/(1.0-alpha))
c      print*,'dlna= ',dlna,' dlnb= ',dlnb
ccxxxxxxxxxxxxxxxx
      write(4,*)'Trials= ',n
      write(4,*)'Upper Specification= ',u
      write(4,*)'theta0= ',mu0
```

```
      write(4,*)'prior s= ',sd0,' me s= ',sdme
      write(4,*)'c3= ',c3
      write(4,*)'c2l= ',c2l,' c2h= ',c2h
      write(4,*)'c1l= ',c1l,' c1h= ',c1h
      write(4,*)'s1= ',s1,' a2= ',a2,' r2= ',r2
      write(4,*)'alpha= ',alpha,' beta= ',beta
      write(4,*)'lwr indif= ',the0,' upr indif= ',the1
      write(3,305)
23    continue
c     accept the seed from the user. run 5000 trials for each
c     seed
c     initiate all counters
c
      print*,'seed? 999 to quit'
      read(*,300)isd
      if(isd.eq.999)goto 600
      igd=0
      ingd=0
      call seed(isd)
      tctot=0.
      sttot=0.
      rej1=0
      acc1=0
      rej2=0
      acc2=0
      rej3=0
      acc3=0
      jr1=0
      jr2=0
      jr3=0
      ja1=0
      ja2=0
      ja3=0
      jrst1=0
      jrst2=0
      jrst3=0
      jrst4=0
      jast1=0
      jast2=0
      jast3=0
      jast4=0
      nost=0
      do 30 j=1,4
         stacc(j)=0
         strej(j)=0
30    continue
      matcha=0
      matchr=0
c      matcha1=0
c      matcha2=0
c      matcha3=0
c      matchr1=0
c      matchr2=0
c      matchr3=0
```

```
c
c
c
c
      do 200 k=1,n
      strun=0.
      sumx=0.
      jdec=0
      cst=0.
      st=0.
      stflgr=0
      stflga=0
c     generate random variables from the prior distribution
c     this is the actual batch value
40      call random(rv1)
      if(rv1.eq.0.)goto 40
50      call random(rv2)
      if(rv2.eq.0.)goto 50
c      print*,'rv1= ',rv1,'rv2= ',rv2
      z=sqrt(-2.*log(rv1))*cos(2.*3.1415927*rv2)
      zme=sqrt(-2.*log(rv1))*sin(2.*3.1415927*rv2)
      yact=z*sd0+mu0
c     generate measurement error components from the me dist
c     use with actual value to generate observation values 1-3
c
      do 60 j=1,3
53      call random(rv1)
      if(rv1.eq.0.)goto 53
56      call random(rv2)
      if(rv1.eq.0.)goto 56
      zme=sqrt(-2.*log(rv1))*sin(2.*3.1415927*rv2)
      yobs(j)=yact+(zme*sdme)
60    continue
cxxxxxxxxxxxxxxxxxx
c     statistical case:  find the log of likelihood ratio at
c     each observation
c     determine at each stage if disposition can be made
c
      do 70 i=1,3
      strun=strun+s1
      sumx=sumx+yobs(i)
      dllr=((the1-the0)/sdme**2.0)*sumx+(i/(2.0*sdme**2.0))*
     \(the0**2.0-the1**2.0)
      if(dllr.lt.dlnb)then
c        print*
c        write(*,248)dllr
c        write(*,249)dlnb
c        write(*,*)'********  Accept batch  ********'
c        print*
        stflga=i
        stacc(i)=stacc(i)+1
c        jstat=i
        goto 80
        elseif(dllr.ge.dlna)then
```

```
c         print*
c         write(*,250)dllr
c         write(*,251)dlna
c         write(*,*)'******** Reject batch ********'
c         print*
         stflgr=i
         strej(i)=strej(i)+1
c         jstat=i
         goto 80
      endif
c     next code to use statistical (truncated sprt) rule of thumb
c     in the case that no decision is reached in nmax
70    continue
      nost=nost+1
      if(dllr.gt.0.)then
        strej(4)=strej(4)+1
         if(yact.le.u)then
            jrst4=jrst4+1
         endif
       else
        stacc(4)=stacc(4)+1
         if(yact.gt.u)then
            jast4=jast4+1
         endif
      endif
cxxxxxxxxxxxxxxxxxx
c     econ case:  compare observation mean to cutoff values
c     at each stage
80    continue
      if(yobs(1).le.c1l)then
        acc1=acc1+1
        jdec=1
        tcrun=s1
        goto 100
      endif
      if(yobs(1).gt.c1h)then
        rej1=rej1+1
        jdec=2
        tcrun=s1
        goto 100
      endif
      if((yobs(2)+yobs(1))/2.le.c2l)then
        acc2=acc2+1
        jdec=3
        tcrun=2.*s1
        goto 100
      endif
      if((yobs(2)+yobs(1))/2.gt.c2h)then
        rej2=rej2+1
        jdec=4
        tcrun=2.*s1
        goto 100
      endif
      if((yobs(3)+yobs(2)+yobs(1))/3.le.c3)then
```

```
        acc3=acc3+1
        jdec=5
        tcrun=3.*s1
        goto 100
      endif
      if((yobs(3)+yobs(2)+yobs(1))/3.gt.c3)then
        rej3=rej3+1
        jdec=6
        tcrun=3.*s1
        goto 100
      endif
c     determine if the correct disposition was made
c     assess false disposition costs
100   continue
      if(yact.le.u)then
         igd=igd+1
         if(jdec.eq.2)then
            jr1=jr1+1
            cst=r2
c             write(*,*)'false econ reject'
         endif
         if(stflgr.eq.1)then
            jrst1=jrst1+1
            st=r2
         endif
         if(jdec.eq.4)then
            jr2=jr2+1
            cst=r2
         endif
         if(stflgr.eq.2)then
           jrst2=jrst2+1
            st=r2
         endif
         if(jdec.eq.6)then
            jr3=jr3+1
            cst=r2
         endif
         if(stflgr.eq.3)then
           jrst3=jrst3+1
            st=r2
         endif
        else
         ingd=ingd+1
         if(jdec.eq.1)then
            ja1=ja1+1
            cst=a2
c             write(*,*)'false econ accept'
         endif
c     assign economic costs to the statistical case -
c     determine if correct disposition, if not add cost
         if(stflga.eq.1)then
           jast1=jast1+1
            st=a2
         endif
```

```
      if(jdec.eq.3)then
        ja2=ja2+1
        cst=a2
      endif
      if(stflga.eq.2)then
        jast2=jast2+1
        st=a2
      endif
      if(jdec.eq.5)then
        ja3=ja3+1
        cst=a2
      endif
      if(stflga.eq.3)then
        jast3=jast3+1
        st=a2
      endif
    endif
c   find if the econ and stat cases match
    if(jdec.eq.1.and.stflga.gt.0)matcha=matcha+1
    if(jdec.eq.3.and.stflga.gt.0)matcha=matcha+1
    if(jdec.eq.5.and.stflga.gt.0)matcha=matcha+1
    if(jdec.eq.2.and.stflgr.gt.0)matchr=matchr+1
    if(jdec.eq.4.and.stflgr.gt.0)matchr=matchr+1
    if(jdec.eq.6.and.stflgr.gt.0)matchr=matchr+1
    if(stflga.eq.0.and.stflgr.eq.0)then
c       nost=nost+1
      strun=3*s1
    endif
    tcrun=tcrun+cst
c    print*,'tcrun= ',tcrun
    strun=strun+st
    tctot=tctot+tcrun
    sttot=sttot+strun
200 continue
c
c
    tcave=tctot/n
    stave=sttot/(n-nost)
cc    write(*,*)'RUN #',m
c    write(*,*)'x1= ',y1obs
c    print*,('                  1           2           3')
c    print*,'Match Acc = ',matcha1,' ',matcha2,' ',matcha3
c    print*,'Match Rej = ',matchr1,' ',matchr2,' ',matchr3
    write(4,*)
    write(4,*)'*****'
    write(4,*)'Match Acc = ',matcha,'     ',real(matcha)/real(n)
    write(4,*)'Match Rej = ',matchr,'     ',real(matchr)/real(n)
    print*,'*****'
    write(4,315)
    write(4,320)
    write(4,325)stacc(1),real(stacc(1))/real(n),acc1,
   &real(acc1)/real(n)
    write(4,345)stacc(2),real(stacc(2))/real(n),acc2,
   &real(acc2)/real(n)
```

```
      write(4,365)stacc(3),real(stacc(3))/real(n),acc3,
     &real(acc3)/real(n)
      write(4,335)strej(1),real(strej(1))/real(n),rej1,
     &real(acc1)/real(n)
      write(4,355)strej(2),real(strej(2))/real(n),rej2,
     &real(acc2)/real(n)
      write(4,375)strej(3),real(strej(3))/real(n),rej3,
     &real(acc3)/real(n)
c     write(4,385)nost,real(nost)/real(n),real(nost*s1)/real(n)
      write(4,385)nost,real(nost)/real(n)
      write(4,388)stacc(4),strej(4)
      write(4,389)jast4,jrst4,real(jast4*a2)/real(n),
     &real(jrst4*r2)/real(n)
      write(4,390)jast1,real(jast1)/real(n),ja1,
     &real(ja1)/real(n)
      write(4,410)jast2,real(jast2)/real(n),ja2,
     &real(ja2)/real(n)
      write(4,430)jast3,real(jast3)/real(n),ja3,
     &real(ja3)/real(n)
      write(4,442)jast1+jast2+jast3,real(jast1+jast2+jast3)/
     &real(n),ja1+ja2+ja3,real(ja1+ja2+ja3)/real(n)
      write(4,443)jast1+jast2+jast3+jast4,real(jast1+jast2+jast3+
     &jast4)/real(n),ja1+ja2+ja3,real(ja1+ja2+ja3)/real(n)
      write(4,400)jrst1,real(jrst1)/real(n),jr1,
     &real(jr1)/real(n)
      write(4,420)jrst2,real(jrst2)/real(n),jr2,
     &real(jr2)/real(n)
      write(4,440)jrst3,real(jrst3)/real(n),jr3,
     &real(jr3)/real(n)
      write(4,444)jrst1+jrst2+jrst3,real(jrst1+jrst2+jrst3)/
     &real(n),jr1+jr2+jr3,real(jr1+jr2+jr3)/real(n)
      write(4,445)jrst1+jrst2+jrst3+jrst4,real(jrst1+jrst2+jrst3+
     &jrst4)/real(n),jr1+jr2+jr3,real(jr1+jr2+jr3)/real(n)
      print*,'*****'
      write(4,450)stave,tcave
      stall=(sttot+real(jast4*a2)+real(jrst4*r2))/real(n)
      write(4,455)stall
      write(4,*)'tot act above u= ',ingd,' ',real(ingd)/real(n)
      write(4,*)'tot act below u= ',igd,' ',real(igd)/real(n)
c     write(4,*)
      cr1=(real(jr1)/real(n))*r2
      ca1=(real(ja1)/(n))*a2
c     write(4,*)'false econ rej on 1= ',real(jr1)/real(n),
c    &' cost= ',cr1
c     write(4,*)'false econ acc on 1= ',real(ja1)/real(n),
c    &' cost= ',ca1
c     write(4,*)
      write(3,310)s1,ca1,cr1,tcave
248   format(' Ln of likelihood ratio, ',f14.4,', less than')
249   format(' ln of B, ',f8.4,'.')
250   format(' Ln of likelihood ratio, ',f14.4,', greater than')
251   format(' ln of A, ',f8.4,'.')
300   format(i3)
305   format(5x,'tc(1)',7x,'tc(a2)',7x,'tc(r3)',7x,'tcost')
```

```
310  format(1x,f4.2,5x,f11.8,5x,f11.8,5x,f11.8)
   goto 23
315  format(35x,'Stat',15x,'Econ')
320  format(30x,'Count',5x,'%',10x,'Count',5x,'%')
325  format(' Count/% Accept on 1',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
335  format(' Count/% Reject on 1',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
345  format(' Count/% Accept on 2',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
355  format(' Count/% Reject on 2',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
365  format(' Count/% Accept on 3',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
375  format(' Count/% Reject on 3',8x,i5,5x,f6.4,5x,i5,5x,f6.4)
c385  format(' No Decision',16x,i5,5x,f6.4,3x,'It cost',4x,f6.4)
385  format(' No Decision',16x,i5,5x,f6.4)
388  format('      acc/rej             ',i5,'/',i5)
389  format('      false acc/false rej  ',i5,'/',i5,5x,'Cost',
   &2x,f8.4,'/',f8.4)
390  format(' Count/% False Accept on 1',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
400  format(' Count/% False Reject on 1',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
410  format(' Count/% False Accept on 2',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
420  format(' Count/% False Reject on 2',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
430  format(' Count/% False Accept on 3',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
440  format(' Count/% False Reject on 3',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
442  format(' Count/% False Accept (3) ',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
443  format(' Count/% False Accept tot ',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
444  format(' Count/% False Reject (3) ',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
445  format(' Count/% False Reject tot ',2x,i5,5x,f6.4,5x,i5,
   &5x,f6.4)
450  format(' Ave Tot Cost (in 3)',10x,f8.4,10x,f8.4)
455  format(' Ave Stat Cost (w/Trunc)',5x,f8.4)
600    return
   end
```

VITA

Raeanne Lambert

Candidate for the Degree of

Doctor of Philosophy

Thesis: A SEQUENTIAL DECISION METHOD FOR HOMOGENEOUS BATCH DISPOSITION IN THE PRESENCE OF MEASUREMENT ERROR

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Ada, Oklahoma, July 11, 1960, the daughter of Ray and Joanne Lambert.

Education: Graduated from Ada High School, Ada, Oklahoma in May 1978; received Bachelor of Science degree in Petroleum Engineering from The University of Tulsa, Tulsa, Oklahoma in December, 1982; received Master of Business Administration degree from Eastern New Mexico University, Portales, New Mexico in December, 1985. Completed the requirements for the Doctor of Philosophy degree at Oklahoma State University in July, 1995.

Professional Experience: Engineering Instructor, Tulsa Junior College, 1994- ; Quality Engineer, Baxter Healthcare Corporation, 1990-1994; Systems Engineer, Integrated Support Systems, 1987-1988; Petroleum Engineer, Chevron, U.S.A., 1983-1986.