A COMPUTER PROGRAM TO DEVELOP AND EVALUATE A 2-SPRT FOR THE NEGATIVE BINOMIAL, BINOMIAL, AND POISSON DISTRIBUTION

By

SIEW JOO LIM

Bachelor of Science in Business Administration

Oklahoma State University

Stillwater, Oklahoma

1987

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfilment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1989

A COMPUTER PROGRAM TO DEVELOP

AND EVALUATE A 2-SPRT FOR THE

NEGATIVE BINOMIAL, BINOMIAL,

AND POISSON DISTRIBUTION

Thesis Approved:

_____

Thesis Adviser

_____

_____

_____

Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my report adviser, Dr. Linda J. Young, for her encouragement and guidance throughout the entire stage for the completion of this report. Many thanks go to my adviser, Dr. Larry Claypool, for his advice and support and to Dr. Ronald W. McNew, for serving on my graduate committee.

In addition, I would like to thank each member of the faculty and to all my friends in the Department of Statistics for their warmth and encouragement. Much appreciation goes to all the brothers and sisters in the Chinese Bible Studies Group and church members for their prayer and loving care.

To my beloved family, I owe a great deal to them for their moral support and love and for believing in my abilities. A special thanks goes to my husband, Ung Hieng, for his love and understanding. He has made the rough time seems easier and he is always there for me.

Last but not least, I would like to thank GOD who makes all things possible. I give all the praises and honors to HIM.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Much work has been directed toward improving the performance of the Sequential Probability Ratio Test (SPRT) by reducing the expected sample size for paremeter values between the two hypothesized ones. Lorden (1976) proposed a procedure for determining a simple combination of two one-sided SPRT's known as the 2-SPRT. This procedure approximately minimizes the expected sample size at a given point θ_0 among all tests with error probabilities controlled at two other points, θ_1 and θ_2 $(\theta_1 < \theta_2)$.

This report investigates the use of a computer program for applying the 2-SPRT procedure to three discrete distributions : binomial, Poisson, and negative binomial. The user specifies the hypothesized values for θ_1 and θ_2 and the respective error probabilities, α and β. Then a 2-SPRT test for the specified distribution is developed using a θ_0 which, at least asymptotically, has the maximum expected sample size.

Chapter II provides a literature review of the 2-SPRT sampling procedure including its introduction and recent development. The 2-SPRT as a solution to the Kiefer-Weiss and the modified Kiefer-Weiss problems is discussed. The 2-SPRT for the binomial distribution, the Poisson distribution, and the negative binomial distribution are also derived.

Chapter III describes the use of the program. Chapter IV analyzes the factors that influence the maximum sample size (M), the average sample number (ASN), the 95^{th} percentile of the sample size ($N_{.95}$) and the operating characteristic function

(OC). A summary and possible avenues of future research are in Chapter V.

CHAPTER II

LITERATURE REVIEW

The sequential probability ratio test (SPRT) was developed by Wald, 1947. Its purpose is to decide on the basis of a sequence of independent observations $X_1, X_2, \ldots$ whether θ_1 or θ_2 is the true value of the parameter θ where the error probabilities are at most α and β. Wald's SPRT has a remarkable optimality property; that is, it minimizes both $E_{\theta_1}(N)$ and $E_{\theta_2}(N)$ among all tests with equal or smaller error probabilities. However, unsatisfactorily large sample sizes may be required for values of θ between θ_1 and θ_2 (Wetherill, 1975). Thus substantial effort has been devoted toward improving the performance of the SPRT by reducing the expected sample size for parameter values between the hypotheses. The problem of finding a procedure which minimizes the maximum expected sample size over all possible θ subject to the error probability constraints α and β is known as the Kiefer-Weiss problem. No optimal results have been found for this problem.

Kiefer and Weiss (1957) proved structure theorem about tests which minimize the expected sample size, $E_{\theta_0}(N)$, for a fixed $\theta = \theta_0$ which is called the modified Kiefer-Weiss problem. Bechhofer (1960) pointed out the desirability of solving the Kiefer-Weiss problem; that is, minimizing the maximum expected sample size over all possible θ. Weiss (1962) showed that the Kiefer-Weiss problem reduces to the modified problem in symmetric cases involving the normal and binomial distributions. Lai (1973) investigated the Weiner process case. Lorden (1976) developed the 2-SPRT as an asymptotic solution to the modified Kiefer-Weiss problem. In 1980, he characterized the basic structure of optimal tests for the modified problem,

with particularly informative results for the Koopman-Darmois family of distributions (Lorden, 1980). In 1983, Huffman extended Lorden's work by showing how to choose θ_0 so that an asymptotic solution of the Kiefer-Weiss problem is obtained for the Koopman-Darmois family. For the case of testing the mean of an exponential density, extensive computer calculations comparing the proposed 2-SPRT with an optimal procedure showed that the 2-SPRT comes within 2% of minimizing the maximum expected sample size over a broad range of error probabilities and parameter values (Huffman, 1983). Nagardeolekar (1988) applied the 2-SPRT to obtain an approximate solution to the Kiefer-Weiss problem for the negative binomial distribution and studied some of its properties through Monte Carlo simulation. We will first review the formulation of the 2-SPRT for a general distribution from the Koopman-Darmois family. Then we will develop the test for each of the three discrete distributions considered in this report.

Review of the Formulation of the 2-SPRT

Assume $X_1, X_2, \cdots$ to be independent and identically distributed random variables from one of the Koopman-Darmois densities $f_\theta(x) = \exp\{\theta x - b(\theta)\}$ $(\theta_i < \theta < \theta_j)$ with respect to a non- degenerate σ - finite measure μ. The function $b(\theta)$ is differentiable on (θ_i, θ_j) and its first two derivatives need to satisfy $b'(\theta) = E_\theta(X)$ and $b''(\theta) = Var_\theta(X)$ (Koopman, 1936). Furthermore, the Kullback Leibler information numbers:

$$I(\theta, \phi) = E_\theta \log\{f_\theta(x)/f_\phi(x)\}$$

are given by

$$I(\theta, \phi) = (\theta - \phi)b'(\theta) - (b(\theta) - b(\phi))$$

Consider testing $H_1: \theta = \theta_1$ versus $H_2: \theta = \theta_2$ $(\theta_1 < \theta_2)$ with error probabilities at most α and β. A third hypothesis $H_0: \theta = \theta_0$ is needed in a 2-SPRT procedure

where θ_0 is the function of θ_1 and θ_2 for which the expected sample size is minimized. If θ_0 is the value of θ in the parameter space that has maximum expected sample size, then a solution to the Kiefer-Weiss problem is obtained. The value of θ_0 which will be found will provide an asymptotic solution to this problem.

Consider the usual likehood ratios :

$$\frac{f_{1n}}{f_{0n}} = \frac{f_1(X_1)...f_1(X_n)}{f_0(X_1)...f_0(X_n)} \quad and \quad \frac{f_{2n}}{f_{0n}} = \frac{f_2(X_1)...f_2(X_n)}{f_0(X_1)...f_0(X_n)}$$

The stopping rule is based on the following conditions :

(1) Reject H_1 if $(f_{1n}/f_{0n}) \leq A$

(2) Reject H_2 if $(f_{2n}/f_{0n}) \leq B$

(3) Continue sampling if neither (1) nor (2) is satisfied,

where $0 \leq A, B \leq 1$, are not both zero and $\alpha + \beta \leq max(A, B)$. The sample size $N(A, B)$ is the smallest $N \geq 0$ such that the sampling is stopped by reaching decision (1) or (2). Choose $\hat{N} = f(x, \theta_1)$ if inequality (1) is not satisfied and $\hat{N} = f(x, \theta_2)$ if inequality (2) is not satisfied. If both (1) and (2) are satisfied simultaneously, then any fixed rule can be used to decide between $f(x, \theta_1)$ and $f(x, \theta_2)$.

Lorden (1976) pointed out that the method which Wald used to derive upper bounds for the error probabilities of an SPRT is applicable to the 2-SPRT and yields $\alpha < A$ (rejecting H_1) and $\beta < B$ (rejecting H_2). Setting $A = \alpha$ in (1) and $B = \beta$ in (2) ensures error probabilities of at most α and β. Furthermore Lorden (1976) developed a theorem which shows that the 2-SPRT provides an asymptotic solution to the modified Kiefer-Weiss problem.

Lorden's Theorem (1976):

Let $\alpha(A, B)$ and $\beta(A, B)$ denote the error probabilities of the 2-SPRT $(N(A, B), \hat{N})$. Let $n(A, B)$ denotes the infimum of $E(n)$ over all tests satisfying $\alpha \leq \alpha\ (A, B)$ and

$\beta \leq \beta\ (A, B)$. Under the assumption that

$$E\left[\log^2\left(\frac{f_{01}}{f_{11}}\right)\right] \quad and \quad E\left[\log^2\left(\frac{f_{01}}{f_{21}}\right)\right]$$

are finite and f_0, f_1, f_2 are distinct, if A, $B > 0$, then

$$E\{N(A,B) - n(A,B)\} \rightarrow 0 \ as \ min(A,B) \rightarrow 0$$

Thus, for any fixed θ, the 2-SPRT provides an asymptotic solution to the modified Kiefer-Weiss problem. Lorden's numerical results indicate that 2-SPRT's have efficiencies greater than 99% regardless of the size of the error probabilities.

Approximate Solution For Kiefer-Weiss Problem

Lorden (1976) presented a theorem which states that the 2-SPRT provides an approximate solution to the Kiefer-Weiss problem if there exist a $\tilde{\theta}$ for which the expected sample size is maximized in the parameter space. Given α and β are the true error probabilities of the 2-SPRT, Huffman's (1983) theorem states that for a $\theta_0 = \tilde{\theta}$, the 2-SPRT procedure minimizes the maximum expected sample size to within $\sigma((\log(\alpha^{-1}))^{1/2})$ under the condition that $0 < C_1 < \log(\alpha)/\log(\beta) < C_2 < \infty$ for fixed but arbitrary constants C_1 and C_2. In other words, the 2-SPRT provides an approximate solution to the Kiefer-Weiss problem for $\theta_0 = \tilde{\theta}$.

To determine how to choose $\tilde{\theta}$, first define θ^* so that

$$\frac{\log(1/A)}{I(\theta^*, \theta_1)} = \frac{\log(1/B)}{I(\theta^*, \theta_2)}$$

Once θ^* is found $\tilde{\theta}$ can be determined using the following relationship between $\tilde{\theta}$ and θ^* (Huffman, 1983).

$$\tilde{\theta} = \theta^* + \frac{r^*}{\sigma^* \sqrt{n^*}}$$

Based on the general formulation of the 2-SPRT, the 2-SPRT for the binomial, Poisson, and negative binomial distribution are derived.

DERIVATION OF THE 2-SPRT FOR THE BINOMIAL DISTRIBUTION

The probability mass function for the binomial distribution is

$$f_X(x;n,p) = \begin{cases} \binom{n}{x} p^x q^{n-x}, & x = 0,1,2,\cdots,n; \\ \\ 0, & \text{elsewhere,} \end{cases}$$

where $0 < p < 1$ and $q = 1 - p$. Since the binomial distribution belongs to the Koopman-Darmois family of distributions, the probability mass function can be written in the following form :

$$f(x) = exp\{\theta x - b(\theta)\},\ \theta_i < \theta < \theta_j$$

where $\theta = \log(p/q)$ and $b(\theta) = -n * \log(1-p) - \log\binom{n}{x}$. The $b(\theta)$ satisfies the following conditions:

$$b'(\theta) = ne^{\theta}/(1+e^{\theta}) = np = E_{\theta}(x)$$

$$b''(\theta) = n * (e^{\theta}/(1+e^{\theta})) * (1/(1+e^{\theta})) = npq = Var_{\theta}(x)$$

The Kullback-Leibler information numbers for this family of densities may be written as

$$I(\theta, \theta_i) = (\theta - \theta_i)b'(\theta) - \{b(\theta) - b(\theta_i)\}$$

Thus for the binomial distribution, we have

$$I(p,p_i) = (\log(p/q) - \log(p/q)) * np + \{n * \log(q) + \log\binom{n}{x} - n\log(q_i) - \log\binom{n}{x}\}$$

which may be written as

$$I(p,p_i) = np\log(pq_i/qp_i) + n\log(q/q_i) \qquad (2.1)$$

for $i = 1,2$. Both $I(p,p_1)$ and $I(p,p_2)$ are positive on (p_1,p_2).

Consider testing a null hypothesis $H_1: p = p_1$ against an alternative hypothesis $H_2: p = p_2$ $(p_1 < p_2)$. We are interested in minimizing the expected sample size for $p = p_0$, $(p_1 < p_0 < p_2)$. Therefore, we define a third hypothesis $H_0: p = p_0$. A one-sided SPRT of H_0 against H_1 is conducted for possible rejection of H_1. At the same time, another one-sided SPRT of H_0 against H_2 is conducted for possible rejection of H_2. The test is conducted by taking one observation at a time from the population and computing $T_n = \sum_{i=1}^{n} x_i$, the sum of the number of observations obtained up to that point. The 2-SPRT stopping rule may now be stated as follows

(1) Reject H_1 in favor of $H_2 : p = p_2$ if

$$\frac{p_1^{T_n} q_1^{n-T_n}}{p_0^{T_n} q_0^{n-T_n}} \leq A$$

That is, if

$$T_n \geq \frac{\log(1/A) + n\log(q_1/q_0)}{\log(p_0 q_1/p_1 q_0)} \tag{2.2}$$

(2) Reject H_2 in favor of $H_1 : p = p_1$ if

$$\frac{p_2^{T_n} q_2^{n-T_n}}{p_0^{T_n} q_0^{n-T_n}} \leq B$$

That is, if

$$T_n \leq \frac{\log(B) + n\log(q_0/q_2)}{\log(p_2 q_0/p_0 q_2)} \tag{2.3}$$

(3) Continue sampling if neither (2.2) nor (2.3) is satisfied and $n < M$ where

$M(p_0)$ = the smallest integer

$$\geq \frac{\log(1/A)\log(p_2 q_0/p_0 q_2) - \log(B)\log(p_0 q_1/p_1 q_0)}{\log(q_0/q_2)\log(p_0 q_1/p_1 q_0) - \log(q_1/q_0)\log(p_2 q_0/p_0 q_2)} \tag{2.4}$$

At $n = M$ a decision is made based on the following condition :

(1) Reject H_2 if $T_n < Mp_0$

(2) Reject H_1 if $T_n > Mp_0$

(3) Randomize with equal probability if $T_n = Mp_0$

Since two one-sided SPRT's are being conducted simultaneously, the error rates must be adjusted if the desired Type I and Type II error rates of α and β, respectively, are to be achieved. Huffman (1983) developed the following formu las for determining the constants A and B for desired error probabilities α and β.

Let

$$A(p) = \frac{a_2(p) - a_1(p)}{a_2(p)}\alpha \tag{2.5}$$

and

$$B(p) = \frac{a_1(p) - a_2(p)}{a_1(p)}\beta \tag{2.6}$$

where the constants $a_i(p)$ $i = 1, 2$ are defined by

$$a_i(p) = \frac{\log(pq_i/qp_i)}{I(p, p_i)}$$

and $I(p, p_i)$ is defined in (2.1).

According to Huffman's theorem, the 2-SPRT provides an approximate solution to the Kiefer-Weiss problem for $p_0 = \tilde{p}$. The problem now reduces to how to determine $\tilde{p}$. First determine p^* such that

$$\frac{\log(1/A)}{I(p^*, p_1)} = \frac{\log(1/B)}{I(p^*, p_2)} \tag{2.7}$$

where $A = A(p^*)$, $B = B(p^*)$, and $I(p^*, p_i)$ can be obtained using (2.1)..

Once p^* is found, $\tilde{p}$ can be determined using the relationship

$$\tilde{\theta} = \theta^* + \frac{r^*}{\sigma^*\sqrt{n^*}} \tag{2.8}$$

This may be written in terms of $\tilde{p}$ and p^* as

$$\log\left(\frac{\tilde{p}}{1-\tilde{p}}\right) = \log\left(\frac{p^*}{q^*}\right)\left(\frac{r^*}{\sigma^*\sqrt{n^*}}\right)$$

Solving for $\tilde{p}$, we obtain

$$\tilde{p} = \frac{\left(\frac{p^*}{q^*}\right) exp\left(\frac{r^*}{\sigma^*\sqrt{n^*}}\right)}{1 + \left(\frac{p^*}{q^*}\right) exp\left(\frac{r^*}{\sigma^*\sqrt{n^*}}\right)}$$

where

$$r^* = \Phi^{-1}\left[\frac{a_2(p^*)}{a_1(p^*) - a_2(p^*)}\right] \tag{2.9}$$

$\Phi(.)$ denotes the cumulative distribution function of the standard normal variate and

$$\sigma^* = \sqrt{n^* p^* q^*}$$

and

$$n^* = \frac{\left[\frac{\log(1/B)}{I(p^*,p_2)} a_1(p^*) \; - \; \frac{\log(1/A)}{I(p^*,p_1)} a_2(p^*)\right]}{a_1(p^*) \; - \; a_2(p^*)} \tag{2.10}$$

DERIVATION OF THE 2-SPRT FOR THE POISSON DISTRIBUTION

The probability mass function for the Poisson distributon is :

$$f_X(x;\lambda) \; = \begin{cases} \frac{\lambda}{x!}\, e^{-\lambda x}, & x = 0, 1, 2, \cdots \\ \\ 0, & \text{elsewhere.} \end{cases}$$

Since the Poisson distribution also belongs to the Koopman-Darmois family of distributions, the probability mass function can be written in the following form:

$$f(x) \; = \; exp\{x \log(\lambda) \; - \; \lambda \; - \; \log(x!)\}$$

where $\theta = \log(\lambda)$, $b(\theta) = \log(x!) \; + \; \lambda$, $b'(\theta) = e^{\theta} = \lambda = \mathrm{E}_\theta(x)$, and $b''(\theta) = e^{\theta} = \lambda = \mathrm{Var}_\theta(x)$. The Kullback-Leibler information number for the Poisson distribution is

$$I(\lambda, \lambda_i) \; = \; \{\log(\lambda) \; - \; \log(\lambda_i)\}\, \lambda \; - \; \lambda \; + \; \lambda_i$$

and hence

$$I(\lambda, \lambda_i) = \lambda\{\log(\lambda/\lambda_i) - 1\} + \lambda_i \tag{2.11}$$

for $i = 1,2$. Both $I(\lambda, \lambda_1)$ and $I(\lambda, \lambda_2)$ are positive on (λ_1, λ_2).

Consider testing $H_1: \lambda = \lambda_1$ against $H_2: \lambda = \lambda_2$ $(\lambda_1 < \lambda_2)$. Construct a third hypothesis $H_0: \lambda = \lambda_0$ where $(\lambda_1 < \lambda_0 < \lambda_2)$. The 2-SPRT testing procedure for the Poisson distribution is similar to the 2-SPRT for the binomial distribution. The 2-SPRT stopping rule is based on the following conditions :

(1) Reject H_1 in favor of $H_2: \lambda = \lambda_2$ if

$$\frac{\lambda_1^{T_n} e^{-n\lambda_1}}{\lambda_0^{T_n} e^{-n\lambda_0}} \leq A$$

That is, if

$$T_n \geq \frac{\log(1/A) + n(\lambda_0 - \lambda_1)}{\log(\lambda_0/\lambda_1)} \tag{2.12}$$

(2) Reject H_2 in favor of $H_1: \lambda = \lambda_1$ if

$$\frac{\lambda_2^{T_n} e^{-n\lambda_2}}{\lambda_0^{T_n} e^{-n\lambda_0}} \leq B$$

That is, if

$$T_n \leq \frac{\log(B) - n(\lambda_0 - \lambda_2)}{\log(\lambda_2/\lambda_0)} \tag{2.13}$$

(3) Continue sampling if neither (2.12) nor (2.13) is satisfied and $n < M$ where

$M(p_0)$ = the smallest interger

$$\geq \frac{\log(1/A)\log(\lambda_2/\lambda_0) - \log(B)\log(\lambda_0/\lambda_1)}{(\lambda_2 - \lambda_0)\log(\lambda_0/\lambda_1) - (\lambda_0 - \lambda_1)\log(\lambda_2/\lambda_0)} \tag{2.14}$$

At $n = M$ a decision is made based on the following conditions :

(1) Reject H_2 if $T_n < M\lambda_0$

(2) Reject H_1 if $T_n > M\lambda_0$

(3) Randomize with equal probability if $T_n = M\lambda_0$

The constants A and B for desired error probabilities α and β are defined in (2.5) and (2.6) by subtituting λ for p.

For the Poisson distribution, $\tilde{\lambda}$ for the approximate Kiefer-Weiss solution can be found as follows. Find λ^* to satisfy

$$\frac{\log(1/A)}{I(\lambda^*, \lambda_1)} = \frac{\log(1/B)}{I(\lambda^*, \lambda_2)}$$

where $I(\lambda, \lambda_i)$ is defined in (2.11), $A(\lambda^*)$ and $B(\lambda^*)$ can be obtained using (2.5) and (2.6), and

$$a_i(\lambda) = \frac{\log(\lambda/\lambda_i)}{I(\lambda, \lambda_i)} \quad , \quad i = 1, 2.$$

Once λ^* is defined, $\tilde{\lambda}$ can be determined from (2.10). Thus, we solve

$$\log(\tilde{\lambda}) \;=\; \log(\lambda^*) \;+\; \frac{r^*}{\sigma^*\sqrt{n^*}}$$

for $\tilde{\lambda}$ and obtain the solution

$$\tilde{\lambda} \;=\; \lambda^* exp\left(\frac{r^*}{\sigma^*\sqrt{n^*}}\right)$$

where r^* is defined using (2.11) and $\sigma^* = \sqrt{\lambda^*}$.

The 2-SPRT provides an approximate solution to the Kiefer-Weiss problem for $\lambda_0 = \tilde{\lambda}$.

DERIVATION OF THE 2-SPRT FOR THE NEGATIVE BINOMIAL DISTRIBUTION

The probability mass function for the negative binomial distribution is

$$f_X(x; n, p) \;=\; \begin{cases} \binom{r+x-1}{r-1} p^r q^x, & x = 0, 1, 2, \cdots \\ \\ 0, & \text{elsewhere} \end{cases}$$

The probability mass function for the negative binomial distribution can be written as

$$f(x) = exp\{x\log(q) + r\log(p) + \log\binom{r+x-1}{r-1}\}$$

where $\theta = \log(1-p)$ and $b(\theta) = -r\log(p) - \log\binom{r+x-1}{r-1}$. The first two derivatives of $b(\theta)$ are

$$b'(\theta) = r(1-p)/p = E_\theta(x)$$

$$b''(\theta) = r(1-p)/p^2 = Var_\theta(x)$$

The Kullback-Leibler information number for this family of distributions is

$$I(p,p_i) = (rq/p)\log(q/q_i) + r\log(p/p_i) \tag{2.15}$$

for i = 1,2. Both $I(p,p_1)$ and $I(p,p_2)$ are positive on (p_1,p_2)

Consider testing H_1: $p = p_1$ against H_2: $p = p_2$ $(p_1 < p_2)$. The third hypothesis would be H_0: $p = p_0$ where $(p_1 < p_0 < p_2)$. The 2-SPRT stopping rule is based on the following conditions:

(1) Reject H_1 in favor of H_2: $p = p_2$ if

$$T_n \leq \frac{\log(A) + nr\log(p_0/p_1)}{\log(q_1/q_0)} \tag{2.16}$$

(2) Reject H_2 in favor of H_1: $p = p_1$ if

$$T_n \geq \frac{\log(1/B) + nr\log(p_2/p_0)}{\log(q_0/q_2)} \tag{2.17}$$

(3) Continue sampling if neither (2.16) nor (2.17) is satisfied and $n < M$ where

$M(p_0)$ = the smallest integer

$$\geq \frac{\log(B)\log(q_1/q_0) + \log(A)\log(q_0/q_2)}{r\{\log(p_2/p_0)\log(q_1/q_0) - \log(p_0/p_1)\log(q_0/q_2)\}} \tag{2.18}$$

At $n = M$ a decision is made based on the following

(1) Reject H_2 if $T_n > Mrq_0/p_0$

(2) Reject H_1 if $T_n < Mrq_0/p_0$

(3) Randomize with equal probability if $T_n = Mrq_0/p_0$

The constants A and B for desired error probabilities α and β are defined in (2.5) and (2.6).

As with the other distributions, we begin to find p_0 by first determining the p^* which will satisfy (2.7) where $I(p^*, p_i)$ is defined in (2.15) and

$$a_i(p) = \frac{\log(q/q_i)}{I(p, p_i)} \quad , \; i = 1, 2.$$

Once p^* is defined, $\tilde{p}$ can be determined from (2.8). Thus we have

$$\tilde{p} = 1 - q^* exp\left(\frac{r^*}{\sigma^* \sqrt{n^*}}\right)$$

where r^* and n^* are defined in (2.9) and (2.10) respectively, and

$$\sigma^* = \sqrt{rq^*/p^*}$$

The approximate Kiefer-Weiss solution is obtained by using $p_0 = \tilde{p}$ in the test given in (2.16), (2.17), and (2.18).

CHAPTER III

THE PROGRAM

A computer program has been written in Quick Basic which allows a researcher to develop a sampling plan for the 2-SPRT. It is designed in such a way that any user can run the program easily. A portion of this program is modified from Nagardeolekar's (1988) FORTRAN program which computed the 2-SPRT stopping boundaries for the negative binomial distribution. The 2-SPRT may be developed using this program when sampling from any one of three discrete population distributions: binomial, Poisson, or negative binomial. In addition to the distribution, a user must specify the hypothesized values for both the null and alternative, and the type I and type II error rates. The user may then obtain the 2-SPRT stopping boundaries, the OC and ASN functions, and the output data sheets. A detailed description of the program, from the user's viewpoint, will now be given.

First, the user is asked if he would like to do the sampling from the negative binomial, the binomial or the Poisson population distribution. Then he is asked to specify the parameter values for the null and alternative hypotheses, the type I error rate, and the type II error rate. For the negative binomial distribution the parameters values are the mean (μ) rather than p where $\mu = k(1-p)/p$. In all cases, the parameter value for the null hypothesis must be less than the parameter value for the alternative hypothesis. For the negative binomial distribution, the parameter value k (number of success before k failures) must also be specified.

Once this information is received, the user is able to choose from the following options:

(1) Compute values for the 2-SPRT boundaries

(2) Compute the exact properties (OC and ASN functions)

(3) Compute the output data sheet

(4) Exit

If the user selects the option to compute the values for the 2-SPRT boundaries, the following information is displayed:

(a) The parameter value of the third hypothesis for which the expected sample size is minimized

(b) The upper and lower intercepts

(c) The upper and lower slopes for the decision boundaries

(d) The maximum sample size

Unlike the SPRT which has parallel boundaries, the 2-SPRT has convergent decision boundaries. Figure 1 shows the continuation region and the terminating decision regions for the 2-SPRT testing procedure. The convergent nature of the decison boundaries assures a termination with a definite decisio n. Table I provides formulas for the slopes and intercepts of the 2-SPRT decision boundaries. Table II gives an example of the output under option (1) for the binomial distribution with $p_1 = 0.4$, $p_2 = 0.7$, $\alpha = 0.05$, $\beta = 0.1$.

On the other hand, if the user choose option (2), the exact OC, POWER, and ASN function and the 95^{th} percentile of the sample size are computed. These exact properties are computed in the following manner. For each observation, there are three possible outcomes. The observation may fall in the acceptance region, the continuation region, or the rejection region. After each observation is taken, the probability of each possible point in the continuation region is computed based on the condition that no boundary has been crossed at an ealier time. All possible values for the n^{th} observation which would result in crossing into the acceptance region for the first time are considered and the associated probabilities are computed given

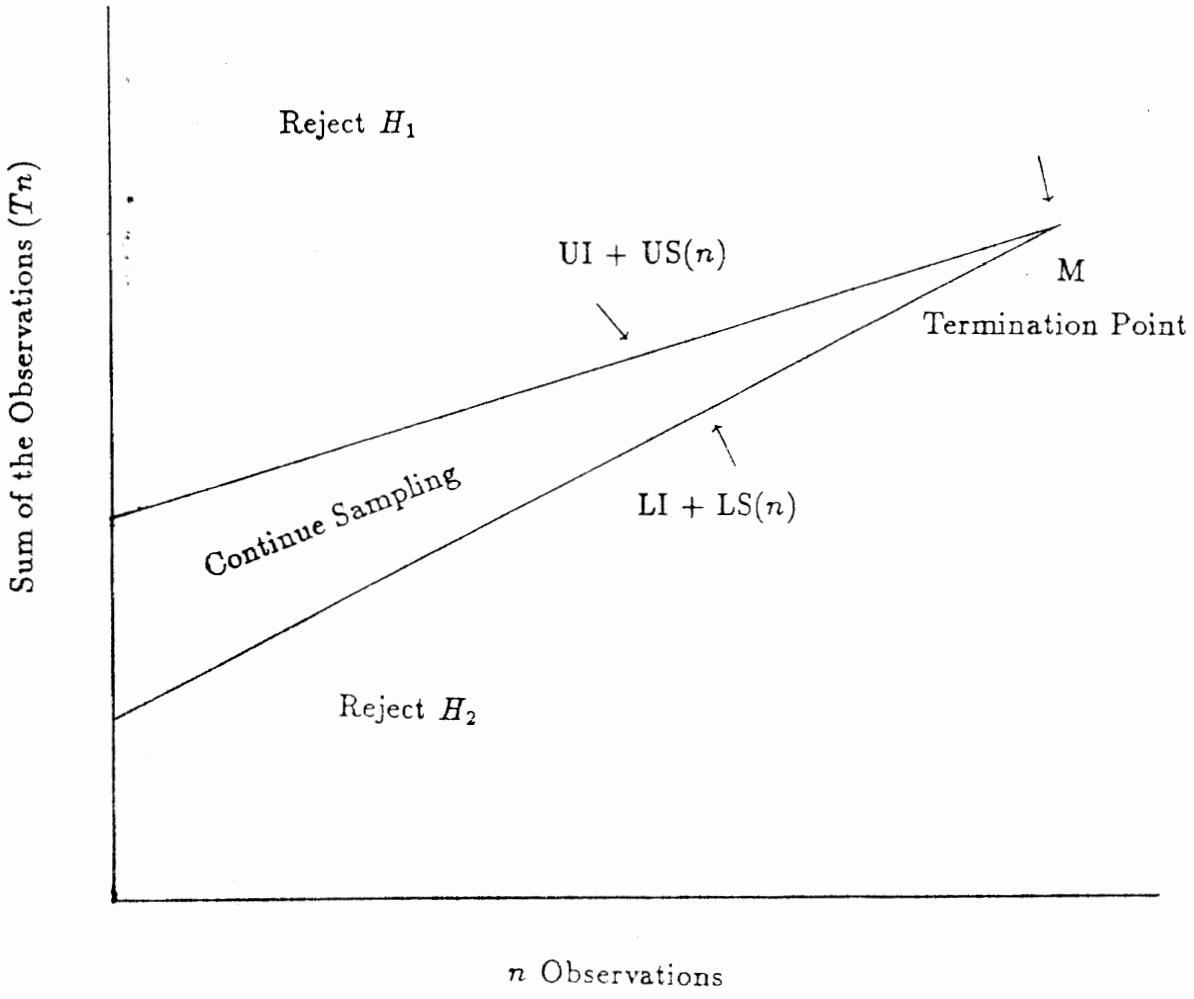


Figure 1. Decision Boundaries for 2-SPRT

TABLE I

SLOPE AND INTERCEPT FOR THE 2-SPRT DECISION BOUNDARIES FOR THE BINOMIAL, POISSON AND NEGATIVE BINOMIAL DISTRIBUTIONS

	BINOMIAL	POISSON	N.BINOMIAL
LI	$\frac{\log(B)}{\log(p_2 q_0 / p_0 q_2)}$	$\frac{\log(B)}{\log(\lambda_2/\lambda_0)}$	$\frac{\log(A)}{\log(q_1/q_0)}$
UI	$\frac{\log(1/A)}{\log(p_0 q_1 / p_1 q_0)}$	$\frac{\log(1/A)}{\log(\lambda_0/\lambda_1)}$	$\frac{\log(1/B)}{\log(q_0/q_2)}$
LS	$\frac{\log(q_0/q_2)}{\log(p_2 q_0 / p_0 q_2)}$	$\frac{\lambda_2 - \lambda_0}{\log(\lambda_2/\lambda_0)}$	$\frac{r \log(p_0/p_1)}{\log(q_1/q_0)}$
US	$\frac{\log(q_1/q_0)}{\log(p_0 q_1 / p_1 q_0)}$	$\frac{\lambda_0 - \lambda_1}{\log(\lambda_0/\lambda_1)}$	$\frac{r \log(p_2/p_0)}{\log(q_0/q_2)}$

TABLE II

A COMPUTER PRINTOUT OF THE
DECISION BOUNDARIES FOR A
BINOMIAL DISTRIBUTION

```
NULL HYPOTHESIS : p1 =    0.400
ALTERNATIVE HYPOTHESIS : p2 =    0.700
ALPHA = 0.10
BETA  = 0.10
THIRD HYPOTHESIS : p0 =    0.552
LOWER INTERCEPT FOR THE BOUNDARY : lint =   -2.524
UPPER INTERCEPT FOR THE BOUNDARY : uint =    2.616
LOWER SLOPE FOR THE BOUNDARY : lslope =    0.628
UPPER SLOPE FOR THE BOUNDARY : uslope =    0.476
MAXIMUM SAMPLE SIZE FOR A DECISION : M =   33.734
```

the sum after the $(n-1)^{th}$ observations. Then the probability of the observation falling into the rejection region is one minus the sum of the probability of an observation falling into either the acceptance region or the continuation region. The OC function is the sum of the probabilities of accepting H_0 after each observation. The cumulative probabilty of stopping (accepting H_1 or H_2) is used to determine the 95^{th} percentile of the sample size. Computation ceases when the probability of termination exceeds 0.9999. Table III shows an example of the output under option (2) for the binomial distribution with $p_1 = 0.4$, $p_2 = 0.7$, $\alpha = \beta = 0.1$.

After the user is satisfied with the properties of a proposed sampling plan, a data sheet containing the upper and lower boundaries can be printed using option (3). This output data sheet is computed using the slopes and intercepts of the decision boundaries. If the number of samples taken exceeds the maximum sample size, computation is stopped. Table IV is an example of the resulting data sheet for a binomial distribution with $p_1 = 0.4$, $p_2 = 0.7$, $\alpha = \beta = 0.1$.

The design of this program will allow the user to go from one part of the computation to another without exiting the program. After completing the desired computations, the user can exit from the program by choosing option (4).

TABLE III

A COMOPUTER PRINTOUT FOR THE EXACT PROPERTIES FOR A BINOMIAL DISTRIBUTION WHEN $p_1 = 0.4$, $p_2 = 0.7$, AND $\alpha = \beta = 0.10$

EXACT VALUES FOR THE BINOMIAL 2-SPRT

p1 = 0.400
p2 = 0.700
alpha = 0.10
beta = 0.10

MU	OC	POWER	ASN	N95
0.080	1.000	0.000	5.523	8.000
0.160	1.000	0.000	6.303	9.000
0.240	0.997	0.003	7.417	12.000
0.320	0.979	0.021	8.971	17.000
0.400	0.911	0.089	10.908	20.000
0.430	0.861	0.139	11.643	22.000
0.460	0.795	0.205	12.313	23.000
0.490	0.712	0.288	12.863	24.000
0.520	0.616	0.384	13.240	25.000
0.550	0.511	0.489	13.399	25.000
0.580	0.404	0.596	13.318	25.000
0.610	0.303	0.697	12.996	25.000
0.640	0.214	0.786	12.462	23.000
0.670	0.141	0.859	11.761	23.000
0.700	0.086	0.914	10.954	21.000
0.760	0.025	0.975	9.263	17.000
0.820	0.005	0.995	7.758	15.000
0.880	0.000	1.000	6.574	11.000
0.940	0.000	1.000	5.682	9.000

TABLE IV

A COMOPUTER PRINTOUT OF THE DATA SHEET FOR A BINOMIAL DISTRIBUTION WHEN $p_1 = 0.4$, $p_2 = 0.7$, AND $\alpha = \beta = 0.10$

SAMPLE NUMBER	LOWER BOUND	RUNNING TOTAL	UPPER BOUND	SAMPLE NUMBER	LOWER BOUND	RUNNING TOTAL	UPPER BOUND
1	-2		3	51	0		0
2	-1		4	52	0		0
3	-1		4	53	0		0
4	0		5	54	0		0
5	1		5	55	0		0
6	1		5	56	0		0
7	2		6	57	0		0
8	2		6	58	0		0
9	3		7	59	0		0
10	4		7	60	0		0
11	4		8	61	0		0
12	5		8	62	0		0
13	6		9	63	0		0
14	6		9	64	0		0
15	7		10	65	0		0
16	8		10	66	0		0
17	8		11	67	0		0
18	9		11	68	0		0
19	9		12	69	0		0
20	10		12	70	0		0
21	11		13	71	0		0
22	11		13	72	0		0
23	12		14	73	0		0
24	13		14	74	0		0
25	13		15	75	0		0
26	14		15	76	0		0
27	14		15	77	0		0
28	15		16	78	0		0
29	16		16	79	0		0
30	16		17	80	0		0
31	17		17	81	0		0
32	18		18	82	0		0
33	18		18	83	0		0
34	0		0	84	0		0
35	0		0	85	0		0
36	0		0	86	0		0
37	0		0	87	0		0
38	0		0	88	0		0
39	0		0	89	0		0
40	0		0	90	0		0
41	0		0	91	0		0
42	0		0	92	0		0
43	0		0	93	0		0
44	0		0	94	0		0
45	0		0	95	0		0
46	0		0	96	0		0
47	0		0	97	0		0
48	0		0	98	0		0
49	0		0	99	0		0
50	0		0	100	0		0

CHAPTER IV

RESULTS

Factors That Influence The Maximum Sample Size

The error rates influence the maximum sample size (M). As α and β increase, the value of M decreases for all three distributions. The value of M also decreases when α is held constant while β increases and when β is held constant while α increases. This is what we would expect to happen since if we are willing to accept a larger error rate, a smaller maximum sample size will be required to make a decision. Tables V, VI, and VII show the behavior of M as we change the error rates α and β for the binomial, Poisson, and negative binomial distributions. Figure 2 shows this behavior graphically for a negative binomail distribution when $\mu_1 = 0.2$, $\mu_2 = 0.7$, and $k = 1$. For all three distributions, the rate M decreases is about the same as α and β increase from one level to another. On the other hand, when β is held constant and α increases, the value of M decreases at a faster rate than when α is held constant and β increases.

Another factor that influences the value of M is the distance between the two hypothesized parameter values. Table VIII shows the changes in M as the distance between the two hypothesized parameters increases when sampling from the binomial distribution with $\alpha = \beta = 0.05$. As the distance between the two hypothesized parameters increases, M decreases. This is true for all three distributions. The value of M drops rapidly at first and then decreases slowly as the distance between the parameters increases. Also larger M values are needed to make a decision for a

TABLE V

THE INFLUENCE OF ERROR PROBABILITIES ON THE MAXIMUM SAMPLE SIZE (M) FOR A NEGATIVE BINOMIALN DISTRIBUTION WHEN μ_1=0.2, μ_2=0.7, AND k=1

$\alpha = \beta$	M	$\alpha(0.05)^1$	M	$\beta(0.05)^2$	M
0.01	72.49	0.01	55.01	0.01	58.70
0.03	51.98	0.03	46.57	0.03	47.71
0.05	42.44	0.05	42.44	0.05	42.44
0.07	36.15	0.07	39.62	0.07	38.91
0.09	31.46	0.09	37.47	0.09	36.24
0.10	29.49	0.10	36.55	0.10	35.11
0.13	24.59	0.13	34.23	0.13	32.29
0.15	21.92	0.15	32.95	0.15	30.75
0.17	19.58	0.17	31.82	0.17	29.40
0.20	16.55	0.20	30.35	0.20	27.68

1 - α is held constant (0.05) while β varies
2 - β is held constant (0.05) while α varies

TABLE VI

THE INFLUENCE OF ERROR PROBABILITIES ON THE MAXIMUM SAMPLE SIZE (M) FOR A BINOMIAL DISTRIBUTION WHEN $p_1 = 0.4$, AND $p_2 = 0.7$

$\alpha = \beta$	M	$\alpha = 0.05^{(1)}$	M	$\beta = 0.05^{(2)}$	M
0.01	81.99	0.01	64.22	0.01	64.67
0.03	58.97	0.03	53.46	0.03	53.62
0.05	48.26	0.05	48.26	0.05	48.26
0.07	41.21	0.07	44.75	0.07	44.65
0.09	35.94	0.09	42.09	0.09	41.90
0.10	33.73	0.10	40.96	0.10	40.74
0.13	28.24	0.13	38.13	0.13	37.82
0.15	25.24	0.15	36.57	0.15	36.20
0.17	22.61	0.17	35.20	0.17	34.79
0.20	19.21	0.20	33.42	0.20	32.94

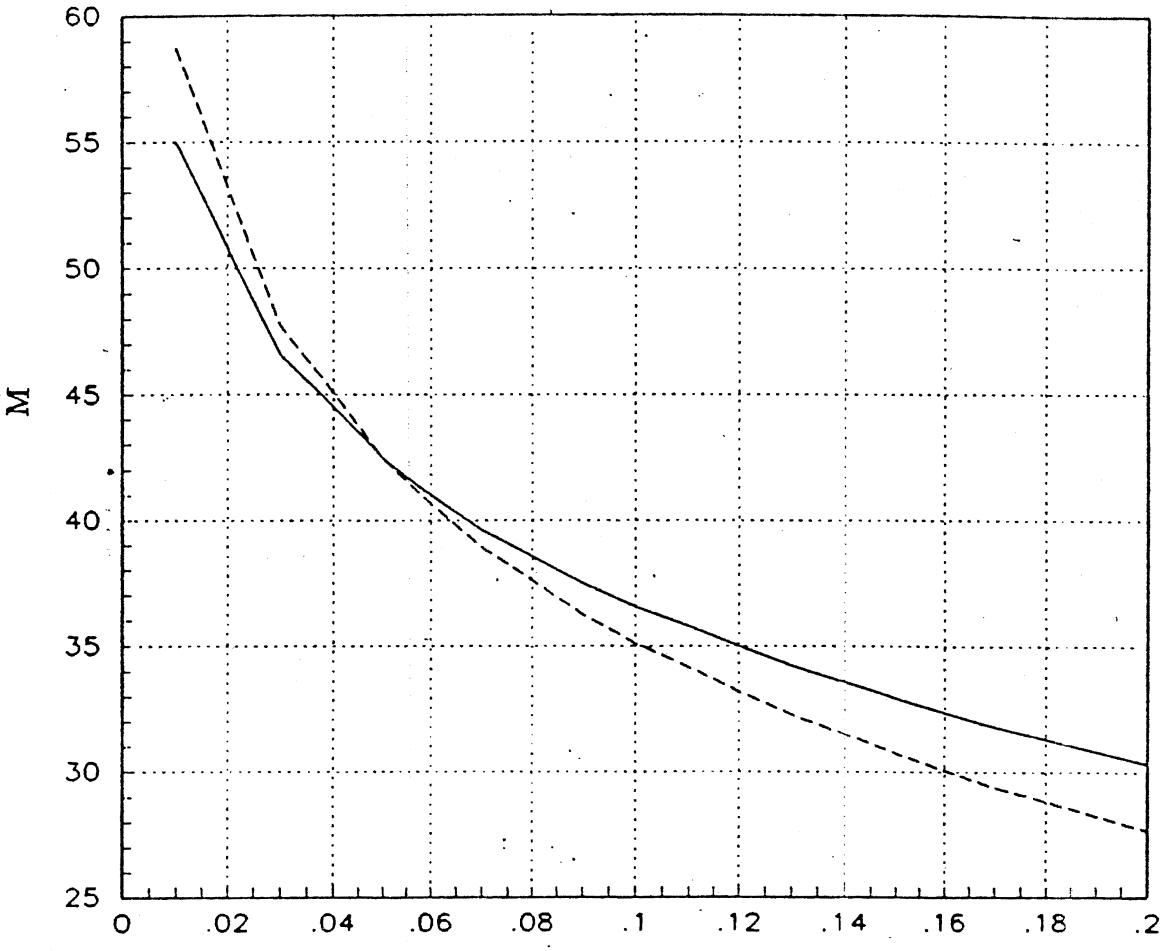
1 - α is held constant (0.05) while β varies
2 - β is held constant (0.05) while α varies

TABLE VII

THE INFLUENCE OF ERROR PROBABILITIES ON THE MAXIMUM SAMPLE SIZE (M) FOR A POISSON DISTRIBUTION WHEN $\lambda_1 = 0.4$, AND $\lambda_2 = 0.7$

$\alpha = \beta$	M	$\alpha(0.05)^1$	M	$\beta(0.05)^2$	M
0.01	187.33	0.01	150.42	0.01	145.08
0.03	134.77	0.03	123.34	0.03	121.55
0.05	110.33	0.05	110.33	0.05	110.33
0.07	94.24	0.07	101.61	0.07	102.88
0.09	82.21	0.09	95.03	0.09	97.32
0.10	77.17	0.10	92.26	0.10	94.99
0.13	64.62	0.13	85.33	0.13	89.27
0.15	57.78	0.15	81.55	0.15	86.22
0.17	51.79	0.17	78.25	0.17	83.61
0.20	44.02	0.20	73.99	0.20	80.36

1 - α is held constant (0.05) while β varies
2 - β is held constant (0.05) while α varies



—— α is constant (0.05) while α increase

---β is constant (0.05) while β increase

Figure 2. Maximum Sample Size for a Negative Binomial Distribution using 2-SPRT when μ_1=0.2, μ_2=0.7, and k=1

TABLE VIII

THE EFFECT OF THE DISTANCE BETWEEN THE TWO HYPOTHESIZED PARAMETERS ON (M) FOR A BINOMIAL DISTRIBUTION WHEN $\alpha = \beta = 0.05$

Null	Alternative	M
0.10	0.20	227.51
0.10	0.30	68.19
0.10	0.40	33.99
0.10	0.50	20.52
0.10	0.60	13.59
0.10	0.70	9.43
0.10	0.80	6.63
0.10	0.90	4.51
0.30	0.40	416.26
0.30	0.60	48.26
0.30	0.80	15.75
0.50	0.60	453.55
0.50	0.70	107.99
0.50	0.80	43.66
0.50	0.90	20.52

set of hypotheses that have larger hypothesized parameters than one with smaller hypothesized parameters.

For the negative binomial distribution, the value of M is also influenced by the parameter k. M decreases as k increases. The rate at which M decreases is greater for a set of small k values than for a set of larger k values. Also, M decreases more rapidly for a set of larger hypothesized parameter values than for a set of smaller hypothesized parameter values. The last two features are illustrated in Table IX.

Factors That Influence The ASN AND $N_{.95}$

Factors that influence the maximum sample size (M) tend to influence the average sample number (ASN) in the same manner. In all the cases, the ASN is less than half the value of M. As expected, the ASN is influenced by the error rates and also the distance between the two hypothesized parameter values for all three distributions. As both α and β increase, the ASN decreases. Also the ASN decreases when α is constant while β increases and when β is constant while α increases. Figure 3 and Figure 4 show this behavior for the binomial distribution when $p_1 = 0.4$ and $p_2 = 0.7$. In addition, as the distance between the two hypothesized parameter values increases, the ASN decreases significantly. Figure 5 illustrates the decrease of the ASN when the distrance between the two hypothesized parameters increases for a binomial distribution when $\alpha = \beta = 0.1$. For the negative binomial distribution, the ASN is also influenced by the parameter k. The ASN decreases as k increases and the rate of decrease depends on the chosen hypothesized parameter values. This feature is showed in Figure 6 for the negative binomial distribution when $\mu_1 = 0.25$, $\mu_2 = 0.4$, $\alpha = \beta = 0.1$.

The same factors that influence the ASN also influence the 95^{th} percentile of the sample size, $N_{.95}$. However, $N_{.95}$ is more sensitive than the ASN. With the same changes in the error rates, $N_{.95}$ is affected more than the ASN. This is true

TABLE IX

THE INFLUENCE ON (M) AS k INCREASED FOR A NEGATIVE BINOMIAL DISTRIBUTION WITH TWO SET OF HYPOTHESIZED PARAMETER VALUES

	$\mu_1 = 0.2$ $\mu_2 = 0.7$ $\alpha = \beta = 0.05$	$\mu_1 = 0.4$ $\mu_2 = 0.9$ $\alpha = \beta = 0.05$
k	M	M
1	42.44	74.47
2	36.21	60.19
3	34.10	55.39
4	33.04	52.98
5	32.41	51.53
6	31.98	50.56
7	31.67	49.87
8	31.44	49.35
9	31.26	48.95
10	31.12	48.62

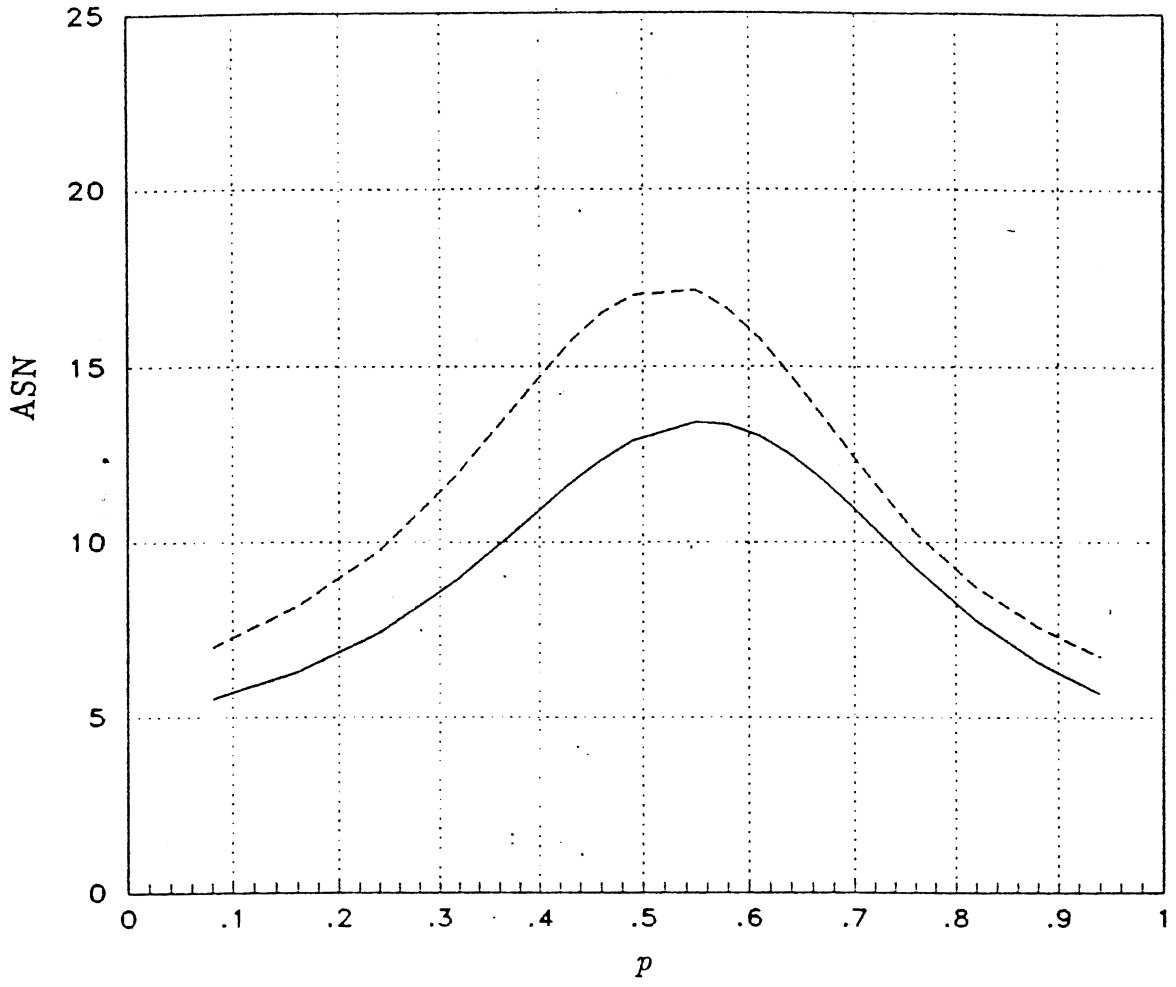
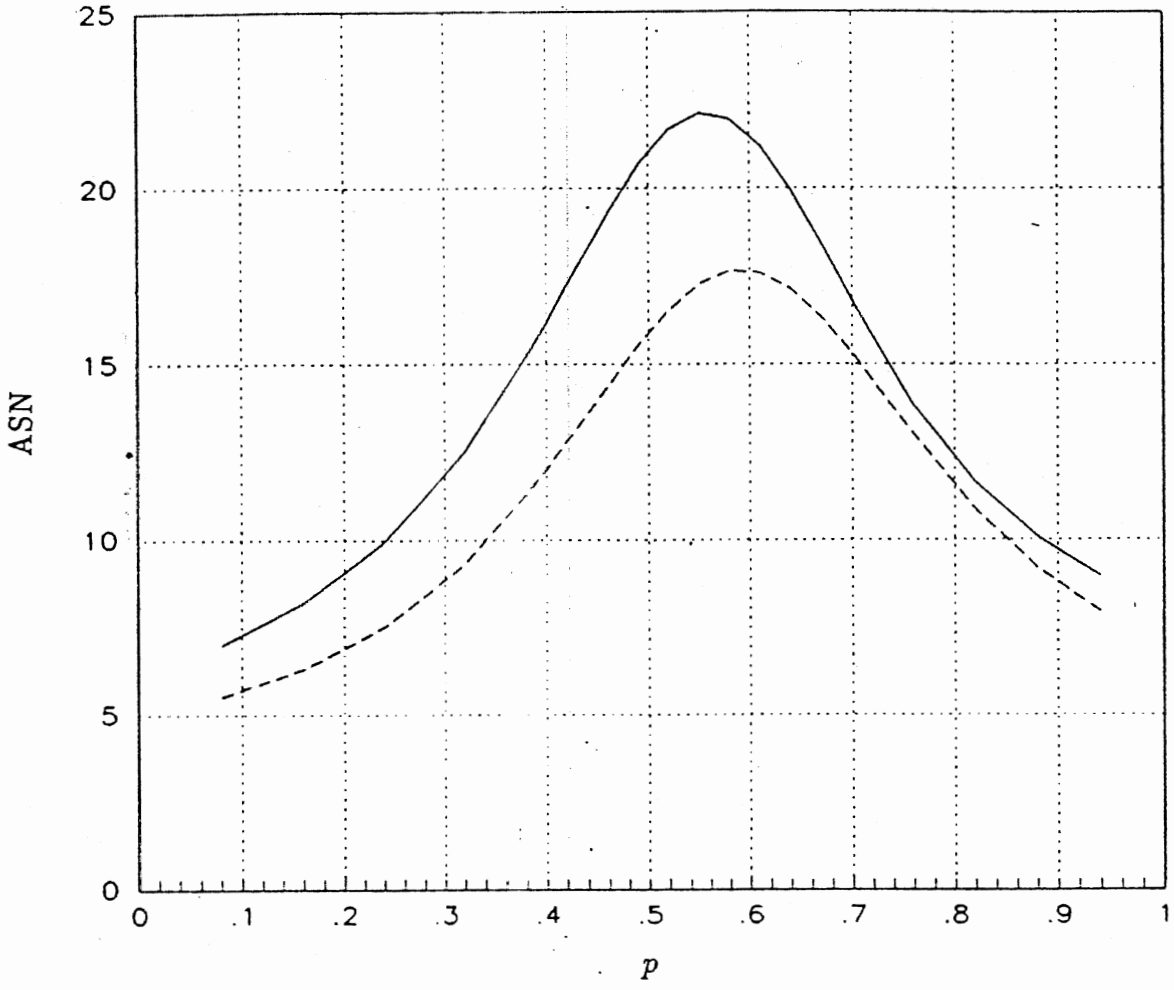


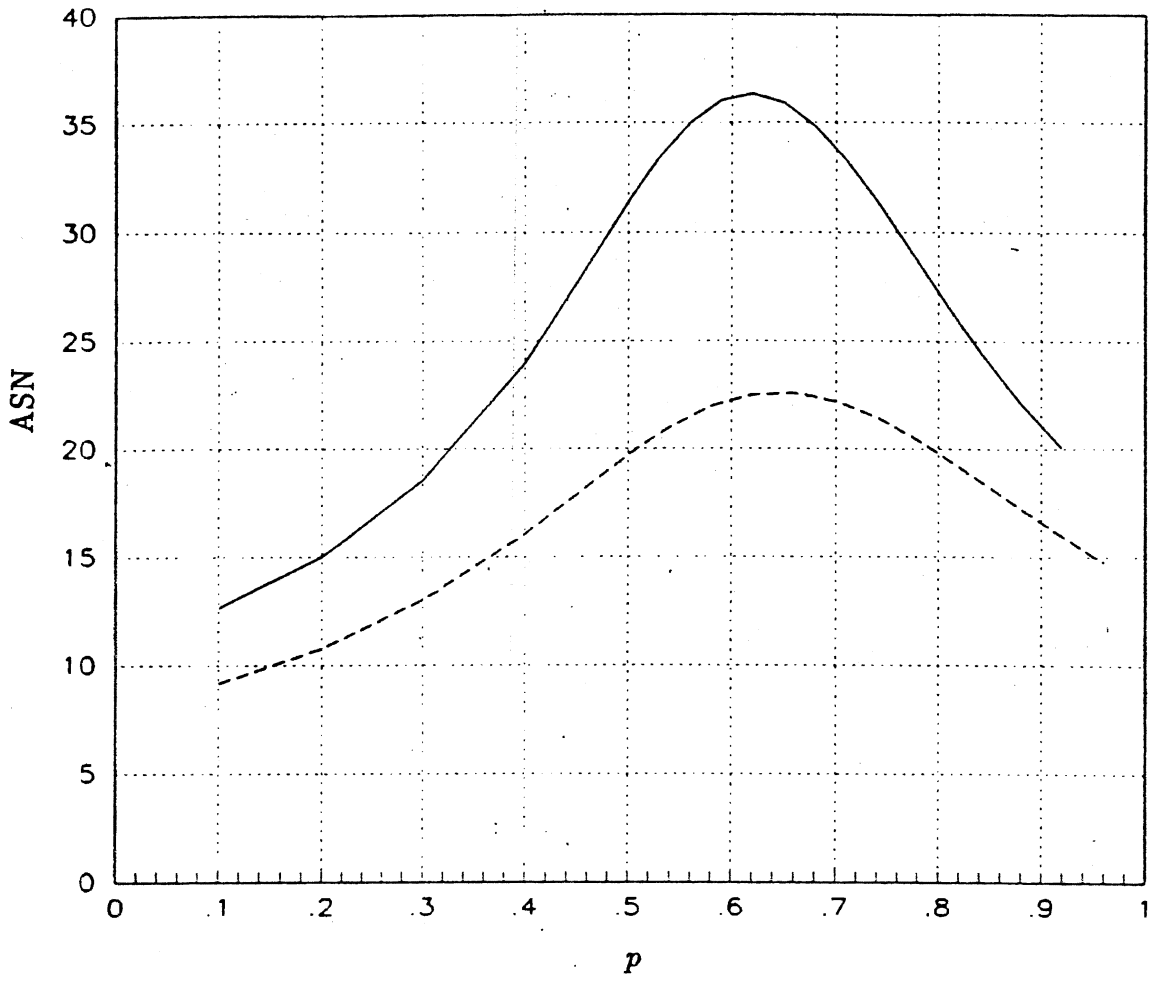
Figure 3. ASN for a Binomial Distribution with two set of error rates when $p_1 = 0.4$, $p_2 = 0.7$



—— $\alpha = \beta = 0.05$

---$\alpha = 0.05, \beta = 0.10$

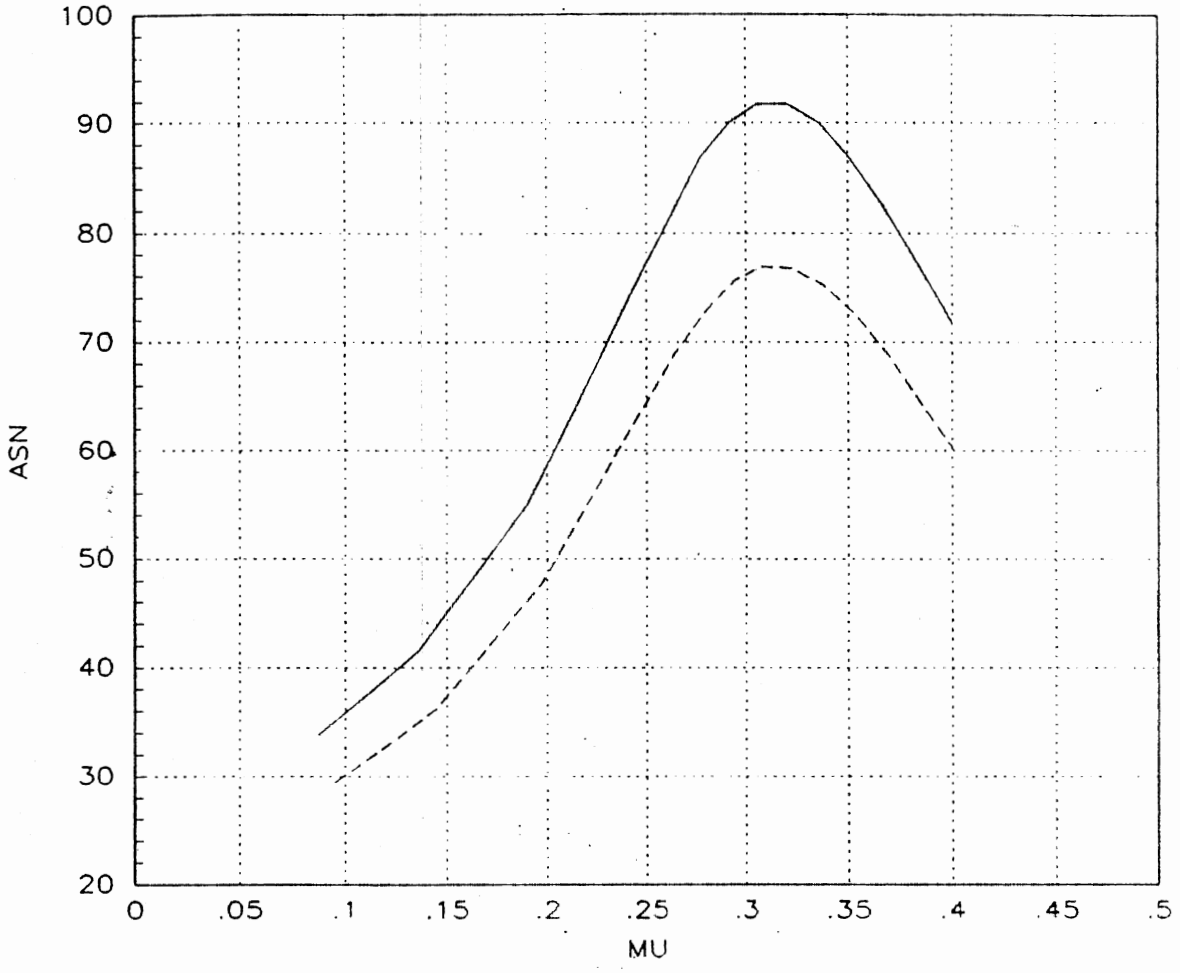
Figure 4. ASN for a Binomial Distribution with two set of error rates when $p_1 = 0.4$, $p_2 = 0.7$



—— $\lambda_1 = 0.5, \lambda_2 = 0.8$

--- $\lambda_1 = 0.5, \lambda_2 = 0.9$

Figure 5. ASN for a Poisson Distribution with two set of hypothesized values when $\alpha = \beta = 0.10$



—— $k = 1$

----$k = 2$

Figure 6. ASN for a Negative Binomial Distribution with two k values when $\mu_1 = 0.25$, $\mu_2 = 0.40$ $\alpha = \beta = 0.10$

also for the changes in the distance between the two hypothesized parameter values. In addition the sensitivity to the error rates and the distance between parameters will increase as the percentile level increases; that is, $N_{.99}$ is more sensitive than $N_{.95}$.

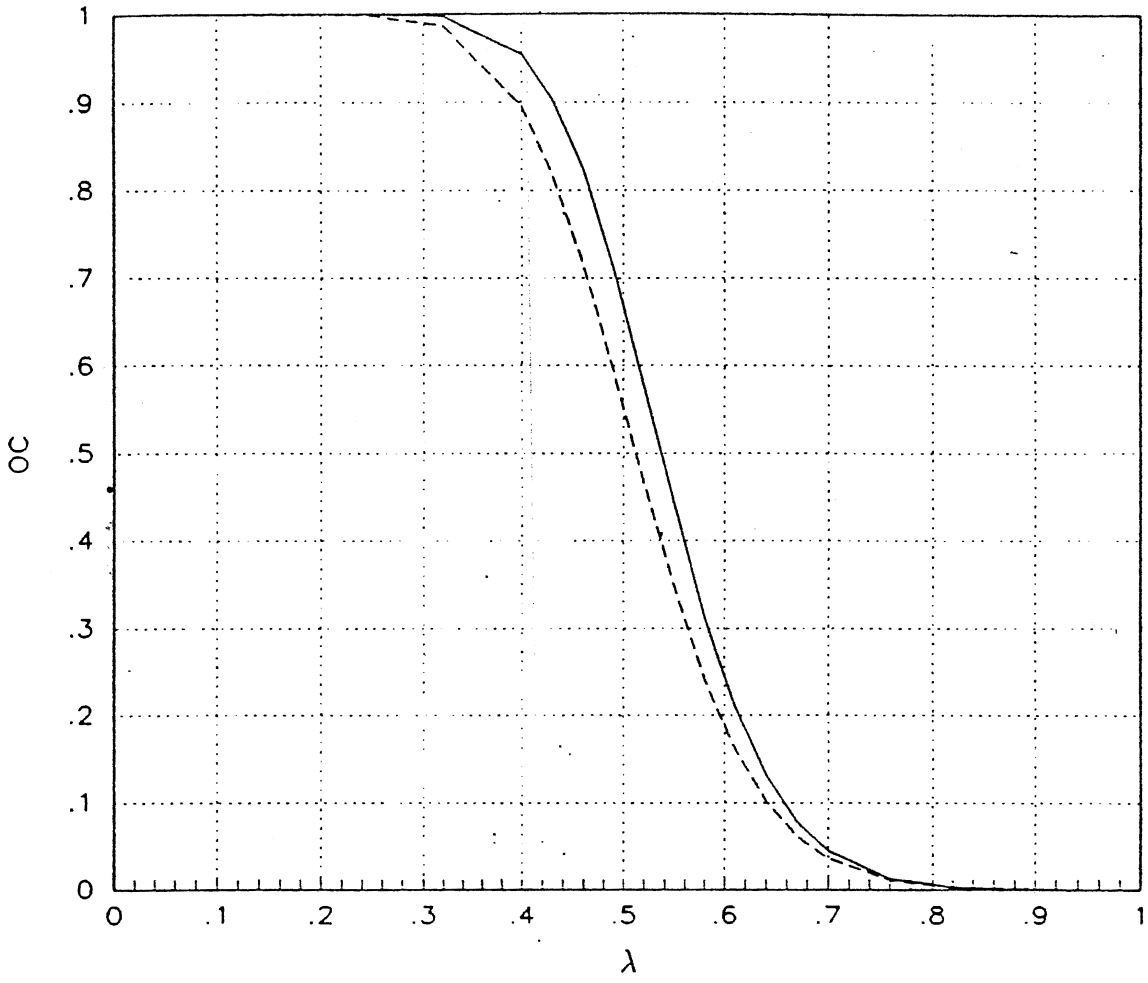
Practically, a researcher can always adjust the error rates in order to obtain an ASN that is economically feasible. A researcher needs to be aware of other factors that will influence the ASN besides the error rates.

Factors That Influence The OC Function

There are two factors that influence the OC function. The first factor is the error rates. For both the binomial and Poisson distribution, the OC value decreases as the probability of type I error (α) increases. On the other hand, the OC value increases as the probability of type II error (β) increases. Figure 7 illustrates the decrease in the OC value as α increases when sampling from the Poisson distribution with $\lambda_1 = 0.5$ and $\lambda_2 = 0.8$. Figure 8 shows the increase in the OC value as β increases when sampling from the same distribution.

For the negative binomial distribution, ther error rates influence the OC function in a different manner. As either α or β increases, the OC value is decreased to a point in a parameter space and then it is increased. The turning point where the OC value starts to decrease is inconsistent. Figures 9 and 10 show the influence of error rates on the OC value for the negative binomial distribution when $\mu_1 = 0.4$, and $\mu_2 = 0.7$.

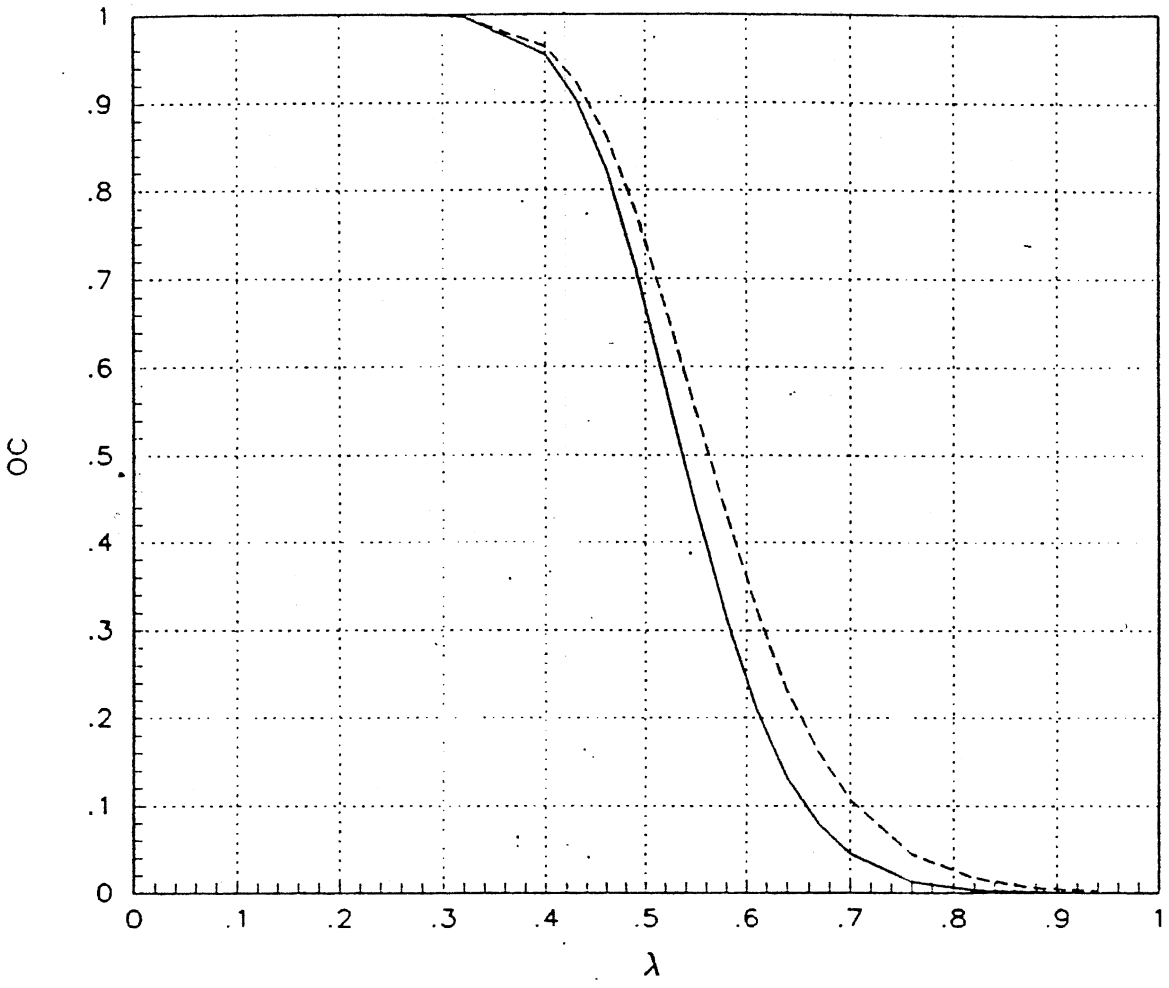
For all three distributions, the OC function is decreased up to a point in the parameter space. After that point, it is increased when both α and β increase. The turning point where the OC value starts to increase is when the true parameter equals the parameter value of the third hypothesis for which the expected sample size is minimized. This feature is illustrated in Figure 11 for the binomial



—— $\alpha = \beta = 0.05$

--- $\alpha = 0.10, \beta = 0.05$

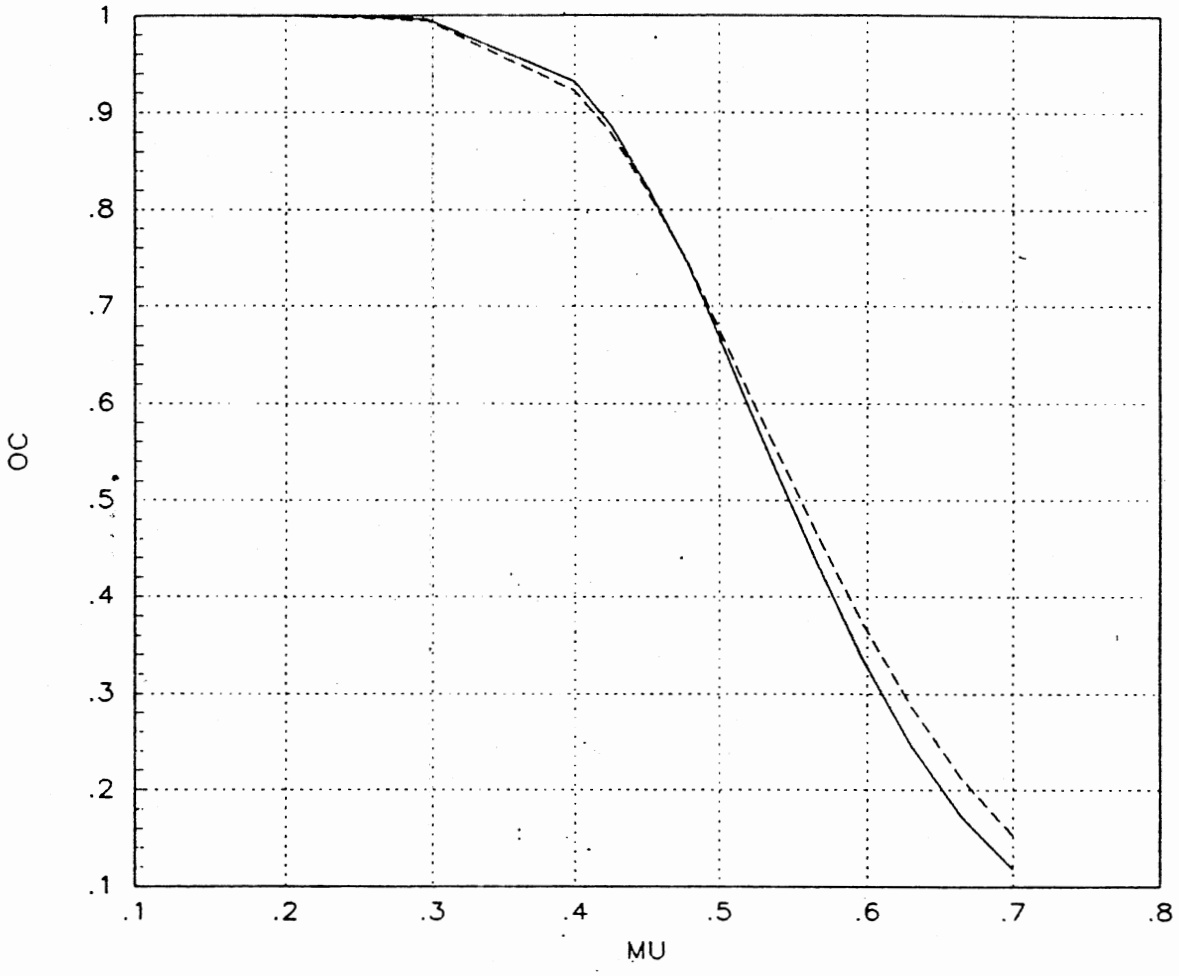
Figure 7. OC corve for a Poisson Distribution with two set of error rates when $\lambda_1 = 0.4$, $\lambda_2 = 0.7$



—— $\alpha = \beta = 0.05$

--- $\alpha = 0.05, \beta = 0.10$

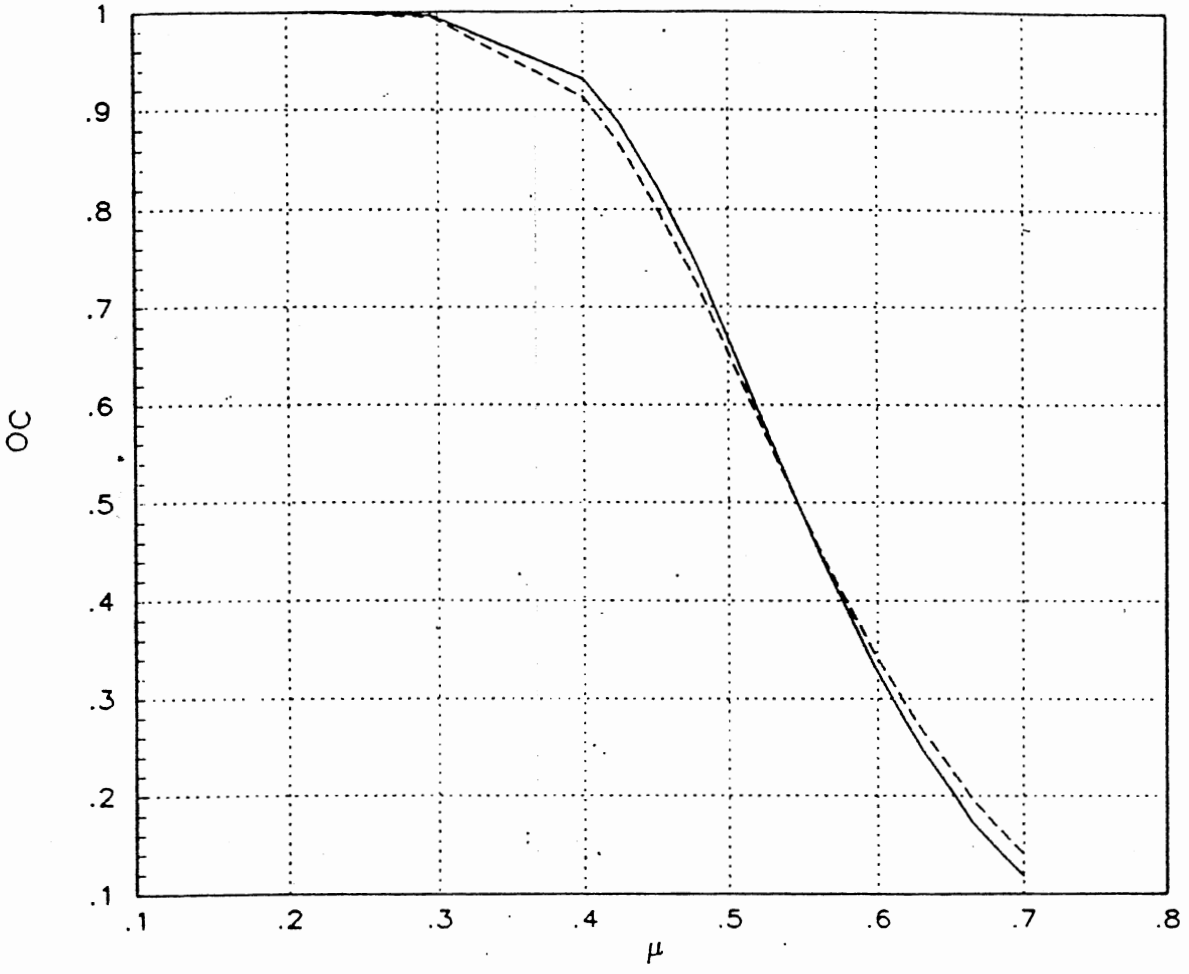
Figure 8. OC corve for a Poisson Distribution with two set of error rates when $\lambda_1 = 0.4$, $\lambda_2 = 0.7$



—— $\alpha = \beta = 0.10$

--- $\alpha = 0.15, \beta = 0.10$

Figure 9. OC curve for a Negative Binomial Distribution with two set of error rates when $\mu_1 = 0.4$, $\mu_2 = 0.7$



—— $\alpha = \beta = 0.10$

---$\alpha = 0.10, \beta = 0.15$

Figure 10. OC curve for a Negative Binomial Distribution with two set of error rates when $\mu_1 = 0.4$, $\mu_2 = 0.7$

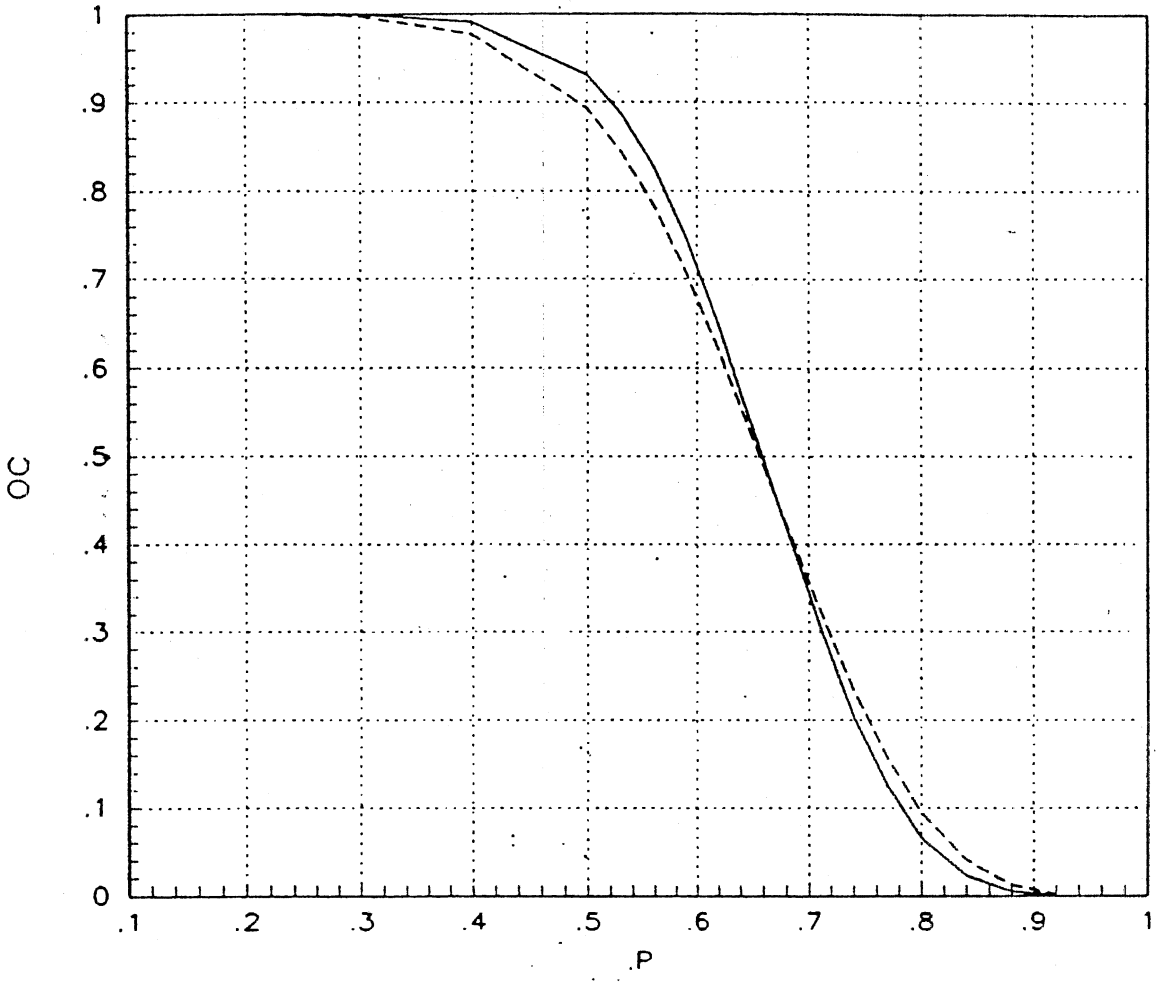
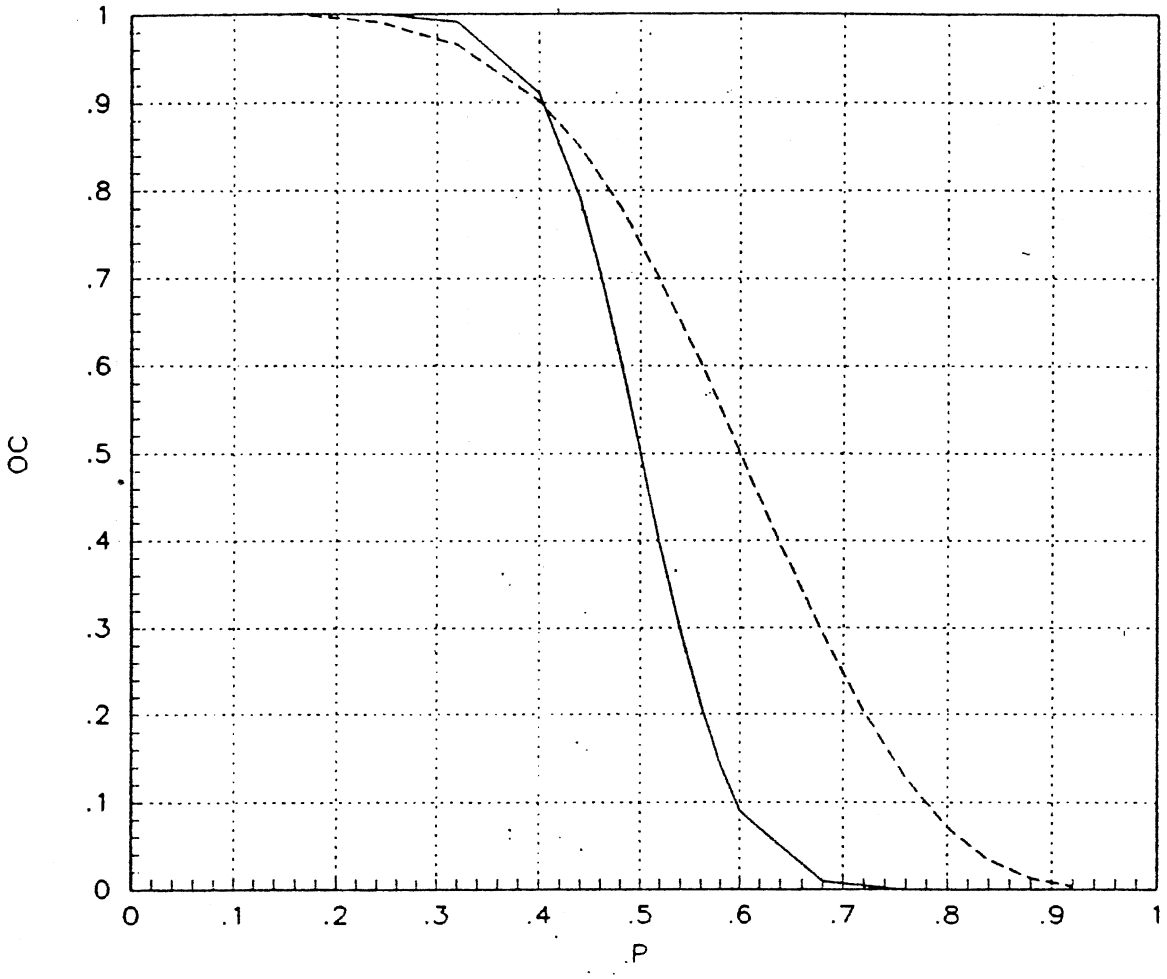


Figure 11. OC corve for a Binomial Distribution with two set of error rates when $p_1 = 0.5$, $p_2 = 0.7$

distribution with $p_1 = 0.5$, and $p_2 = 0.8$ ($p_0 = 0.66$).

The second factor that influences the OC function is the distance between the two hypothesized parameter values. As the distance between the two hypothesized parameter values increases, the OC value decreases up to the point where the true parameters equals the parameter value in the null hypothesis and then the OC value starts to increase. This is true for all three distributions and Figure 12 illustrates this result for the binomial distribution with $\alpha = \beta = 0.1$.



—— $p_1 = 0.4, p_2 = 0.6$

----$p_1 = 0.4, p_2 = 0.8$

Figure 12. OC curve for a Binomial Distribution with two set of hypothesized values when $\alpha = \beta = 0.10$

CHAPTER V

SUMMARY

A computer program to develop and evaluate a sequential testing procedure which consists of two simultaneously conducted sequential probability ratio tests (2-SPRT) for the negative binomial, binomial, and Poisson population distributi ons, is discussed in this report. A researcher can obtain the decision boundaries for the 2-SPRT testing procedure easily using this computer program. Another part of the program computes the exact properties of a proposed sampling plan. Here the researcher determines the average sample number (ASN), the operating characteristic function (OC), and the 95^{th} percentile of the sample size ($N_{.95}$). Parameter values and error rates may be altered until a sampling plan is feasible. Then the user may choose to output a data sheet for immediate field implementation.

Unlike the SPRT, the 2-SPRT sequential testing procedure allows a researcher to have a knowledge of the maximum number of observations he might have to take before he starts the experiment. For all three populations, the value of M decreases as the error rates and the distance between the two hypothesized parameter values increase. The result for both the ASN and the OC function depends on the chosen hypothesized parameter values and the error rates.

This program could be expanded to included other discrete and continuous distributions, such as the normal distribution.

BIBLIOGRAPHY

[1] Bechhoffer, R. E. (1960). A note on the limiting relative efficiency of the Wald sequential probability ratio test. J.Amer.Statist.Assoc. 55, 660-663

[2] Hoeffding, W. (1960). Lower bounds for the expected sample size and the average risk of a sequential procedure. Annals of Mathmatical Statistics. 31, 352-368

[3] Huffman, M. D. (1983). An efficient approximate solution to the Kiefer-Weiss problem. Annals of Mathematical Statistics. 11, 306-316.

[4] Kiefer, J. and Weiss, L. (1957). Some properties of generalized sequential probability ratio tests. Annals of Mathematical Statistics. 28, 57-75.

[5] Koopman, B. O. (1936). On distributions admitting a sufficient statistic. Trans.Amer.Math.Soc.. 39, 399-409.

[6] Lai, T. L. (1973). Optimal stopping and sequential tests which minimize the maximum expected sample size. Ann.Statist.. 1, 659-673

[7] Lorden, G. (1976). 2-SPRT's and the modified Kiefer-Weiss problem of minimizing an expected sample size. Ann.Statist.. 4, 281-291

[8] Lorden, G. (1980). Structure of sequential tests minimizing an expected sample size. Z.Wahrsch.veru.Gebiete 51, 291-302

[9] Mood, A. M., Graybill, F. A., Boes, D. C. (1974). Introduction to the Theory of Statistics . McGraw-Hill Book Company. New York.

[10] Nagardeoleke, M. S. (1988). An approximate solution to the Kiefer- Weiss problem for the negative binomial. Doctor of Philosophy Dissertation. 3, 23-37

[11] Wald, Abraham. (1947). Sequential Analysis. John Wiley and Sons, Inc. New York.

[12] Weiss, L. (1962). On sequential tests which minimize the expected sample size. J.Amer.Statist.Assoc.. 57, 551-566

[13] Wetherill, G. B. (1975). Sequential Methods in Statistics. London, Chapman and Hall. New York.

APPENDIX

COMPUTER PROGRAM

```
DECLARE SUB NEGATIVE (MU1#, MU2#, R#, alpha#, beta#)
DECLARE SUB NB2SPRT (MU1#, MU2#, R#, alpha#, beta#)
DECLARE FUNCTION NBFUNC# (p#)
DECLARE SUB ZERO (b#, C#, abser#, reler#, iflag)
DECLARE SUB NBEXACT (MU1#, MU2#, R#, alpha#, beta#, lint,
                     uint, lslope, uslope, pp(), maxx)
DECLARE SUB NBPROB (p0, R#, pp(), maxx)
DECLARE SUB NBSHEET (MU1#, MU2#, R#, alpha#, beta#)
DECLARE SUB BINOMIAL (p1#, p2#, alpha#, beta#)
DECLARE SUB BI2SPRT (p1#, p2#, alpha#, beta#)
DECLARE FUNCTION BIFUNC# (p#)
DECLARE SUB ZEROBI (b#, C#, abser#, reler#, iflag)
DECLARE SUB BIEXACT (p1#, p2#, alpha#, beta#, lint, uint,
                     lslope, uslope, pp(), maxx)
DECLARE SUB BIPROB (p0, pp(), maxx)
DECLARE SUB BISHEET (p1#, p2#, alpha#, beta#)
DECLARE SUB POISSON (p1#, p2#, alpha#, beta#)
DECLARE SUB PO2SPRT (p1#, p2#, alpha#, beta#)
DECLARE FUNCTION POFUNC# (p#)
DECLARE SUB ZEROPO (b#, C#, abser#, reler#, iflag)
DECLARE SUB POEXACT (p1#, p2#, alpha#, beta#, lint, uint,
                     lslope, uslope, pp(), maxx)
DECLARE SUB POPROB (p0, pp(), maxx)
DECLARE SUB POSHEET (p1#, p2#, alpha#, beta#)
DECLARE SUB SHEET (LI, UI, LS, US, MA)
DECLARE SUB NORTRY (p#, X#, d#, id)
DECLARE SUB POW (p0, mu, lint, uint, lslope, uslope, alpha#,
                 beta#, pp(), maxx)
COMMON SHARED p0, p1#, p2#, MU1#, MU2#, alpha#, beta#, R#,
              p#, lint, uint, lslope, uslope, maxx, n
DIM SHARED pp(1 TO 2, 1 TO 200)


'A PROGRAM TO CALCULATE AN APPROXIMATE SOLUTION TO
'THE KIEFER-WEISS PROBLEM FOR THE NEGAITEVE BINOMIAL.
'BINOMIAL AND POISOON DISTRIBUTION

10    CLS
      LOCATE 6, 17

      PRINT "CHOOSE ONE OF THE FOLLOWING DISTRIBUTION:"
      PRINT
      PRINT "                    (1) NEGATIVE BINOMIAL"
      PRINT "                    (2) BINOMIAL"
      PRINT "                    (3) POISSON"
      PRINT "                    (4) EXIT"
      PRINT
```

```
20    CH$ = INPUT$(1)
      SELECT CASE CH$

      CASE "1"
           CALL NEGATIVE(mu1#, mu2#, R#, alpha#, beta#)
      CASE "2"
           CALL BINOMIAL(p1#, p2#, alpha#, beta#)
      CASE "3"
           CALL POISSON(p1#, p2#, alpha#, beta#)
      CASE "4"
           GOTO 40
      CASE ELSE
           BEEP: GOTO 10
      END SELECT

25    CLS
      LOCATE 6, 17
      PRINT " CHOOSE ONE OF THE FOLLOWING:"
      LOCATE 8, 18
      PRINT "(1) MAKE ANOTHER ANALYSIS"
      LOCATE 9, 18
      PRINT "(2) EXIT"

26    E$ = INPUT$(1)
      SELECT CASE E$
          CASE "1"
                GOTO 10
          CASE "2"
                GOTO 40
          CASE ELSE
               BEEP: GOTO 25
      END SELECT

40    CLS
45    END
```

```
SUB NEGATIVE (mu1#, mu2#, R#, alpha#, beta#)

500 CLS
    LOCATE 5, 15

    PRINT
    PRINT "ENTER VALUES FOR THE FOLLOWING PARAMETERS:"
    PRINT "( mu2 MUST BE GREATER THAN mu1 )"
    PRINT
    INPUT "                    FOR NULL HYPOTHESIS :
                                  mu1 = ", MU1#"
    INPUT "           FOR ALTERNATIVE HYPOTHESIS :
                                  mu2 = ", MU2#"
    INPUT "NUMBER OF SUCCESSES BEFORE X FAILURE :
                                    k = ", R#"
    INPUT "                      ALPHA = ", alpha#
    INPUT "                       BETA = ", beta#
    PRINT

510 CLS
    LOCATE 5, 15
    PRINT
    PRINT "AT THIS TIME YOU MAY WANT TO:"
    PRINT
    PRINT "(1) MAKE CHANGES"
    PRINT "(2) COMPUTE VALUES FOR 2-SPRT DECISION BOUNDARIES"
    PRINT "(3) COMPUTE THE EXACT VALUES (OC AND ASN FUNCTION)"
    PRINT "(4) COMPUTE THE OUTPUT DATA SHEET"
    PRINT "(5) EXIT"

    CH$ = INPUT$(1)
    SELECT CASE CH$
    CASE "1"
       GOTO 500
    CASE "2"
       CALL NB2SPRT(mu1#, mu2#, R#, alpha#, beta#)
    CASE "3"
       CALL NBEXACT(mu1#, mu2#, R#, alpha#, beta#, lint,
                    uint, lslope, uslope, pp(), maxx)
    CASE "4"
       CALL NBSHEET(mu1#, mu2#, R#, alpha#, beta#)
    CASE "5"
       GOTO 560
    CASE ELSE
       BEEP: GOTO 510
    END SELECT
```

```
530 CLS
    LOCATE 6, 15
    PRINT "CHOOSE ONE OF THE FOLLOWING"
    PRINT "(1)   MAKE ANOTHER COMPUTATION"
    PRINT "(2)   DO ANALYSIS FROM ANOTHER DISTRIBUTION"
    PRINT "(3)   EXIT"

540 E$ = INPUT$(1)
    SELECT CASE E$
    CASE "1"
        GOTO 510
    CASE "2"
        GOTO 560
    CASE "3"
        GOTO 560
    CASE ELSE
        BEEP: GOTO 530
    END SELECT

560 END SUB
```

```
SUB NBPROB (p0, R#, pp(), maxx)

q0 = 1 - p0
pp(1, 1) = p0 ^ R#
pp(2, 1) = pp(1, 1)
FOR i = 2 TO 200
pp(1, i) = pp(1, i - 1) * q0 * (R# + i - 2) / (i - 1)
pp(2, i) = pp(2, i - 1) + pp(1, i)
IF pp(2, i) > .9999
   THEN GOTO 130
NEXT i
130 maxx = i - 1

END SUB
```

```
FUNCTION NBFUNC# (p#) STATIC

SHARED MU1#, MU2#, p1#, p2#, alpha#, beta#, R#
p1# = R# / (R# + MU2#)
p2# = R# / (R# + MU1#)
q1# = 1 - p1#
q2# = 1 - p2#
q# = 1 - p#
ss1# = CDBL(LOG(p# / p1#))
ss2# = (q# / p#) * CDBL(LOG(q1# / q#))
inf1# = R# * (ss1# - ss2#)
ss3# = (q# / p#) * CDBL(LOG(q# / q2#))
ss4# = CDBL(LOG(p2# / p#))
inf2# = R# * (ss3# - ss4#)
a1# = (-1) * CDBL(LOG(q1# / q#)) / inf1#
a2# = CDBL(LOG(q# / q2#)) / inf2#
dummy1# = a1# / ((a1# - a2#) * alpha#)
dummy2# = a2# / ((a2# - a1#) * beta#)
ee1# = CDBL(LOG(dummy1#))
ee2# = CDBL(LOG(dummy2#))
NBFUNC# = inf1# * ee2# - inf2# * ee1#

END FUNCTION
```

```
SUB NB2SPRT (MU1#, MU2#, R#, alpha#, beta#)

p1# = R# / (R# + MU2#)
p2# = R# / (R# + MU1#)
pb = R# / (R# + MU1#)
p1 = R# / (R# + MU2#)
q1# = 1# - p1#
q2# = 1# - p2#
'program starts to calculate pterda for keifer-weiss problem
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001

'call subprogram zero to obtain pstar
CALL ZERO(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
   pstar# = b#
ELSE
   ee = 1
END IF
qstar# = 1 - pstar#
zz1# = CDBL(LOG(pstar# / p1#))
zz2# = (qstar# / pstar#) * CDBL(LOG(q1# / qstar#))
inf1s# = R# * (zz1# - zz2#)
zz3# = (qstar# / pstar#) * CDBL(LOG(qstar# / q2#))
zz4# = CDBL(LOG(p2# / pstar#))
inf2s# = R# * (zz3# - zz4#)
a1star# = (-1) * CDBL(LOG(q1# / qstar#)) / inf1s#
a2star# = CDBL(LOG(qstar# / q2#)) / inf2s#
y# = a1star# / (a1star# - a2star#)

'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
rstar# = X#
sigmas# = SQR(R# * qstar# / (pstar# * pstar#))
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
zz5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
zz6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (zz5# - zz6#) / (a2star# - a1star#)
zz7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = 1# - qstar# * CDBL(EXP(zz7#))
qterda# = 1# - pterda#
MUTERDA# = (R# * qterda#) / pterda#
```

```
'find nbar by replacing pstar by pterda
zz8# = CDBL(LOG(pterda# / p1#))
zz9# = (qterda# / pterda#) * CDBL(LOG(q1# / qterda#))
inf1b# = R# * (zz8# - zz9#)
zz10# = (qterda# / pterda#) * CDBL(LOG(qterda# / q2#))
zz11# = CDBL(LOG(p2# / pterda#))
inf2b# = R# * (zz10# - zz11#)
a1bar# = (-1) * CDBL(LOG(q1# / qterda#)) / inf1b#
a2bar# = CDBL(LOG(qterda# / q2#)) / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
zz12# = CDBL(LOG(1# / bb#)) * (a2bar# / inf2b#)
zz13# = CDBL(LOG(1# / aa#)) * (a1bar# / inf1b#)
nbar# = (zz12# - zz13#) / (a2bar# - a1bar#)
y# = a1bar# / (a1bar# - a2bar#)
CALL NORTRY(y#, X#, d#, id)
rbar# = X#
sigmab# = SQR(R# * qterda# / (pterda# * pterda#))
uint = LOG(1 / bb#) / LOG(qterda# / q2#)
uslope = R# * LOG(p2# / pterda#) / LOG(qterda# / q2#)
lint = LOG(aa#) / LOG(q1# / qterda#)
lslope = R# * LOG(pterda# / p1#) / LOG(q1# / qterda#)
c1 = LOG(q1# / qterda#)
c2 = LOG(qterda# / q2#)
c3 = LOG(p2# / pterda#)
c4 = LOG(pterda# / p1#)
capm = (LOG(bb#) * c1 + LOG(aa#) * c2) /
       (R# * (c3 * c1 - c4 * c2))

LPRINT
LPRINT USING "NULL HYPOTHESIS: mu1 =
              ###.###"; MU1#
LPRINT USING "ALTERNATIVE HYPOTHESIS: mu2 =
              ###.###"; MU2#
LPRINT USING "THIRD HYPOTHESIS: mu0 =
              ###.###"; MUTERDA#
LPRINT USING "ALPHA = #.##"; alpha#
LPRINT USING "BETA  = #.##"; beta#
LPRINT USING "NUMBER OF SUCCESSES BEFORE X
              FAILURE : k = #.##"; R#
LPRINT USING "LOWER INTERCEPT FOR THE BOUNDARY :
              lint = ###.###"; lint
LPRINT USING "UPPER INTERCEPT FOR THE BOUNDARY :
              uint = ###.###"; uint
LPRINT USING "LOWER SLOPE FOR THE BOUNDARY :
              lslope = ###.###"; lslope
LPRINT USING "UPPER SLOPE FOR THE BOUNDARY :
              uslope = ###.###"; uslope
LPRINT USING "MAXIMUM SAMPLE SIZE FOR A DECISION :
              M = ###.###"; capm

BEEP
END SUB
```

```
'SUB ZERO (b#, C#, abser#, reler#, iflag) STATIC

maxit = 500
    rzero# = 0
    runit# = 1
    rtwo# = 2
    reight# = 8
    u# = runit#
600 u# = u# / rtwo#
    p# = runit# + u#
    IF (p# - runit#) > 0 THEN
        GOTO 600
    ELSE
        GOTO 601
    END IF
601 u# = u# * rtwo#
    re# = (reler# + u#) / rtwo# + ABS((reler# - u#)
          / rtwo#)
    ic = 0
    acbs# = CDBL(ABS(b# - C#))
    a# = C#
    fa# = NBFUNC#(a#)
    fb# = NBFUNC#(b#)
    fc# = fa#
    count = 2
    k1# = CDBL(ABS(fb#))
    k2# = CDBL(ABS(fc#))
    fx# = CDBL((k1# + k2#) / rtwo# + ABS((k1# - k2#)
          / rtwo#))
602 IF (k2# - k1#) < 0 THEN
        GOTO 603
    ELSE
        GOTO 604
    END IF
603 a# = b#
    fa# = fb#
    b# = C#
    fb# = fc#
    C# = a#
    fc# = fa#
604 cmb# = (C# - b#) / rtwo#
    acmb# = CDBL(ABS(cmb#))
    tol# = re# * CDBL(ABS(b#)) + abser#
    IF (acmb# - tol#) > 0 THEN
        GOTO 605
    ELSE
        GOTO 619
    END IF
605 IF (count - maxit) < 0 THEN
        GOTO 606
    ELSE
        GOTO 625
    END IF
```

```
606 p# = (b# - a#) * fb#
    q# = 1 - p#
    IF p < 0 THEN
        GOTO 607
    ELSE
        GOTO 608
    END IF
607 p# = -1 * p#
    q# = -1 * q#
608 a# = b#
    fa# = fb#
    ic = ic + 1
    IF (ic - 4) < 0 THEN
        GOTO 611
    ELSE
        GOTO 609
    END IF
609 IF (reight# * acmb# - acbs#) < 0 THEN
        GOTO 610
    ELSE
        GOTO 615
    END IF
610 ic = 0
    acbs# = acmb#
611 IF (p# - CDBL(ABS(q#)) * tol#) > 0 THEN
        GOTO 613
    ELSE
        GOTO 612
    END IF
612 IF cmb# < 0 THEN
        b# = b# - CDBL(ABS(tol#))
    ELSE
        b# = b# + CDBL(ABS(tol#))
    END IF
    GOTO 616
613 IF (p# - cmb# * q#) < 0 THEN
        GOTO 614
    ELSE
        GOTO 615
    END IF
614 b# = b# + p# / q#
    GOTO 616
615 b# = (C# + b#) / rtwo#
616 fb# = NBFUNC#(b#)
    IF fb# = 0 THEN
        GOTO 622
    ELSE
        GOTO 617
    END IF
```

```
617 count = count + 1
    IF fb# < 0 THEN
        k3# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k3# = CDBL(ABS(runit#))
    END IF
    IF fc# < 0 THEN
        k4# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k4# = CDBL(ABS(runit#))
    END IF
    IF (k3# - k4#) = 0 THEN
        GOTO 618
    ELSE
        GOTO 602
    END IF
618 C# = a#
    fc# = fa#
    GOTO 602
619 IF fb# < 0 THEN
        k3# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k3# = CDBL(ABS(runit#))
    END IF
    IF fc# < 0 THEN
        k4# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k4# = CDBL(ABS(runit#))
    END IF
    IF (k3# - k4#) = 0 THEN
        GOTO 624
    ELSE
        GOTO 620
    END IF
620 IF (CDBL(ABS(fb#)) - fx#) > 0 THEN
        GOTO 623
    ELSE
        GOTO 621
    END IF
621 iflag = 1
    GOTO 626
622 iflag = 2
    GOTO 626
623 iflag = 3
    GOTO 626
624 iflag = 4
    GOTO 626
625 iflag = 5

626 END SUB
```

```
SUB NBEXACT (MU1#, MU2#, R#, alpha#, beta#, lint,
             uint, lslope, uslope, pp( ), maxx)
p1# = R# / (R# + MU2#)
p2# = R# / (R# + MU1#)
pb = R# / (R# + MU1#)
p1 = R# / (R# + MU2#)
q1# = 1# - p1#
q2# = 1# - p2#
'program starts to calculate pterda for keifer-weiss problem
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001
'call subprogram zero to obtain pstar
CALL ZERO(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
   pstar# = b#
ELSE
   ee = 1
END IF
qstar# = 1 - pstar#
zz1# = CDBL(LOG(pstar# / p1#))
zz2# = (qstar# / pstar#) * CDBL(LOG(q1# / qstar#))
inf1s# = R# * (zz1# - zz2#)
zz3# = (qstar# / pstar#) * CDBL(LOG(qstar# / q2#))
zz4# = CDBL(LOG(p2# / pstar#))
inf2s# = R# * (zz3# - zz4#)
a1star# = (-1) * CDBL(LOG(q1# / qstar#)) / inf1s#
a2star# = CDBL(LOG(qstar# / q2#)) / inf2s#
y# = a1star# / (a1star# - a2star#)
'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
rstar# = X#
sigmas# = SQR(R# * qstar# / (pstar# * pstar#))
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
zz5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
zz6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (zz5# - zz6#) / (a2star# - a1star#)
zz7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = 1# - qstar# * CDBL(EXP(zz7#))
qterda# = 1# - pterda#
'find nbar by replacing pstar by pterda
zz8# = CDBL(LOG(pterda# / p1#))
zz9# = (qterda# / pterda#) * CDBL(LOG(q1# / qterda#))
inf1b# = R# * (zz8# - zz9#)
zz10# = (qterda# / pterda#) * CDBL(LOG(qterda# / q2#))
zz11# = CDBL(LOG(p2# / pterda#))
```

```
inf2b# = R# * (zz10# - zz11#)
a1bar# = (-1) * CDBL(LOG(q1# / qterda#)) / inf1b#
a2bar# = CDBL(LOG(qterda# / q2#)) / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
zz12# = CDBL(LOG(1# / bb#)) * (a2bar# / inf2b#)
zz13# = CDBL(LOG(1# / aa#)) * (a1bar# / inf1b#)
nbar# = (zz12# - zz13#) / (a2bar# - a1bar#)
y# = a1bar# / (a1bar# - a2bar#)
CALL NORTRY(y#, X#, d#, id)
rbar# = X#
sigmab# = SQR(R# * qterda# / (pterda# * pterda#))
uint = LOG(1 / bb#) / LOG(qterda# / q2#)
uslope = R# * LOG(p2# / pterda#) / LOG(qterda# / q2#)
lint = LOG(aa#) / LOG(q1# / qterda#)
lslope = R# * LOG(pterda# / p1#) / LOG(q1# / qterda#)
c1 = LOG(q1# / qterda#)
c2 = LOG(qterda# / q2#)
c3 = LOG(p2# / pterda#)
c4 = LOG(pterda# / p1#)
capm = (LOG(bb#) * c1 + LOG(aa#) * c2) /
       (R# * (c3 * c1 - c4 * c2))
'print the exact properties for the negative
'binomial 2-sprt
LPRINT "EXACT VALUES FOR THE NEGATIVE BINOMIAL 2-SPRT"
LPRINT
LPRINT USING "mu1 = ###.###"; MU1#
LPRINT USING "mu2 = ###.###"; MU2#
LPRINT USING "k = ##.##"; R#
LPRINT USING "alpha = #.##"; alpha#
LPRINT USING "beta = #.##"; beta#
LPRINT
LPRINT "      MU        OC      POWER      ASN      N95"
LPRINT "-------------------------------------------------"
LPRINT
FOR i = 1 TO 3
SELECT CASE i
CASE 1
     inc = p1 / 5
     bot = inc
     top = p1 + .0001
CASE 2
     inc = (pb - p1) / 10
     bot = p1 + inc
     top = pb + .0001
CASE 3
     IF CH$ = "3" THEN inc = p1 / 5
     ELSE inc = (1 - pb) / 5
     bot = pb + inc
     IF CH$ = "3" THEN top = pb + p1
     ELSE top = 1 - inc
END SELECT
```

```
FOR p0 = bot TO top STEP inc
    mu = R# * (1 - p0) / p0
    CALL NBPROB(p0, R#, pp( ), maxx)
CALL POW(p0, mu, lint, uint, lslope, uslope, alpha#,
         beta#, pp( ), maxx)
NEXT p0
NEXT i
BEEP
END SUB
```

```
SUB NBSHEET (MU1#, MU2#, R#, alpha#, beta#)

p1# = R# / (R# + MU2#)
p2# = R# / (R# + MU1#)
pb = R# / (R# + MU1#)
p1 = R# / (R# + MU2#)
q1# = 1# - p1#
q2# = 1# - p2#
'program starts to calculate pterda for keifer-weiss problem
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001

'call subprogram zero to obtain pstar
CALL ZERO(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
   pstar# = b#
ELSE
   ee = 1
END IF
qstar# = 1 - pstar#
zz1# = CDBL(LOG(pstar# / p1#))
zz2# = (qstar# / pstar#) * CDBL(LOG(q1# / qstar#))
inf1s# = R# * (zz1# - zz2#)
zz3# = (qstar# / pstar#) * CDBL(LOG(qstar# / q2#))
zz4# = CDBL(LOG(p2# / pstar#))
inf2s# = R# * (zz3# - zz4#)
a1star# = (-1) * CDBL(LOG(q1# / qstar#)) / inf1s#
a2star# = CDBL(LOG(qstar# / q2#)) / inf2s#
y# = a1star# / (a1star# - a2star#)

'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
rstar# = X#
sigmas# = SQR(R# * qstar# / (pstar# * pstar#))
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
zz5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
zz6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (zz5# - zz6#) / (a2star# - a1star#)
zz7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = 1# - qstar# * CDBL(EXP(zz7#))
qterda# = 1# - pterda#
MUTERDA# = (R# * qterda#) / pterda#
```

```
'find nbar by replacing pstar by pterda
zz8# = CDBL(LOG(pterda# / p1#))
zz9# = (qterda# / pterda#) * CDBL(LOG(q1# / qterda#))
inf1b# = R# * (zz8# - zz9#)
zz10# = (qterda# / pterda#) * CDBL(LOG(qterda# / q2#))
zz11# = CDBL(LOG(p2# / pterda#))
inf2b# = R# * (zz10# - zz11#)
a1bar# = (-1) * CDBL(LOG(q1# / qterda#)) / inf1b#
a2bar# = CDBL(LOG(qterda# / q2#)) / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
zz12# = CDBL(LOG(1# / bb#)) * (a2bar# / inf2b#)
zz13# = CDBL(LOG(1# / aa#)) * (a1bar# / inf1b#)
nbar# = (zz12# - zz13#) / (a2bar# - a1bar#)
y# = a1bar# / (a1bar# - a2bar#)
CALL NORTRY(y#, X#, d#, id)
rbar# = X#
sigmab# = SQR(R# * qterda# / (pterda# * pterda#))

uint = LOG(1 / bb#) / LOG(qterda# / q2#)
uslope = R# * LOG(p2# / pterda#) / LOG(qterda# / q2#)
lint = LOG(aa#) / LOG(q1# / qterda#)
lslope = R# * LOG(pterda# / p1#) / LOG(q1# / qterda#)
c1 = LOG(q1# / qterda#)
c2 = LOG(qterda# / q2#)
c3 = LOG(p2# / pterda#)
c4 = LOG(pterda# / p1#)
capm = (LOG(bb#) * c1 + LOG(aa#) * c2) /
       (R# * (c3 * c1 - c4 * c2))
LI = lint
LS = lslope
UI = uint
US = uslope
MA = capm

LPRINT
LPRINT TAB(20); "        NEGATIVE BINOMIAL 2-SPRT"
LPRINT
LPRINT TAB(24); "  MU1 ="; MU1#; "  MU2 ="; MU2#; " "
                       K = "; R#"
LPRINT TAB(27); "ALPHA ="; alpha#; "  BETA ="; beta#
CALL sheet(LI, LS, UI, US, MA)

END SUB
```

```
SUB BINOMIAL (p1#, p2#, alpha#, beta#)

200 CLS
    LOCATE 5, 15

    PRINT
    PRINT " ENTER VALUES FOR THE FOLLOWING PARAMETERS:"
    PRINT " ( p2 MUST BE GREATER THAN p1 )"
    PRINT
    INPUT "         FOR NULL HYPOTHESIS : p1 = ", p1#
    INPUT " FOR ALTERNATIVE HYPOTHESIS : p2 = ", p2#
    INPUT "                              ALPHA = ", alpha#
    INPUT "                               BETA = ", beta#
    PRINT

210 CLS
    LOCATE 5, 15

    PRINT
    PRINT " AT THIS TIME YOU MAY WANT TO:"
    PRINT
    PRINT "(1) MAKE CHANGES"
    PRINT "(2) COMPUTE VALUES FOR 2-SPRT DECISION BOUNDARIES"
    PRINT "(3) COMPUTE THE EXACT VALUES (OC AND ASN FUNCTION)"
    PRINT "(4) COMPUTE THE OUTPUT DATA SHEETS"
    PRINT "(5) EXIT"

    CH$ = INPUT$(1)
    SELECT CASE CH$
    CASE "1"
       GOTO 200
    CASE "2"
       CALL BI2SPRT(p1#, p2#, alpha#, beta#)
    CASE "3"
       CALL BIEXACT(p1#, p2#, alpha#, beta#, lint, uint,
                    lslope, uslope, pp(), maxx)
    CASE "4"
       CALL BISHEET(p1#, p2#, alpha#, beta#)
    CASE "5"
       GOTO 260
    CASE ELSE
       BEEP: GOTO 210
    END SELECT

230 CLS
    LOCATE 6, 15
    PRINT "CHOOSE ONE OF THE FOLLOWING:"
    PRINT "(1)    MAKE ANOTHER COMPUTATION"
    PRINT "(2)    DO ANALYSIS FROM ANOTHER DISTRIBUTION"
    PRINT "(3)    EXIT"
```

```
240 E$ = INPUT$(1)
    SELECT CASE E$
    CASE "1"
       GOTO 210
    CASE "2"
       GOTO 260
    CASE "3"
       GOTO 260
    CASE ELSE
       BEEP: GOTO 230
    END SELECT

260 END SUB
```

```
SUB BI2SPRT (p1#, p2#, alpha#, beta#)

R# = 1
q1# = 1# - p1#
q2# = 1# - p2#
'program starts to calculate pterda for the
'keifer-weiss problem
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001

'call subprogram ZEROBI to obtain pstar
CALL ZEROBI(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
    pstar# = b#
ELSE
    ee = 1
END IF
qstar# = 1 - pstar#
z1# = CDBL(LOG((pstar# * q1#) / (qstar# * p1#)))
z2# = CDBL(LOG(qstar# / q1#))
inf1s# = R# * (pstar# * z1# + z2#)
z3# = CDBL(LOG((pstar# * q2#) / (qstar# * p2#)))
z4# = CDBL(LOG(qstar# / q2#))
inf2s# = R# * (pstar# * z3# + z4#)
a1star# = z1# / inf1s#
a2star# = z3# / inf2s#
y# = a1star# / (a1star# - a2star#)

'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
rstar# = X#
sigmas# = SQR(R# * pstar# * qstar#)
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
z5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
z6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (z5# - z6#) / (a2star# - a1star#)
z7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = (pstar# / qstar# * CDBL(EXP(z7#))) /
          (1# + (pstar# / qstar# * CDBL(EXP(z7#))))
qterda# = 1# - pterda#

'find nbar by replacing pstar by pterda
z8# = CDBL(LOG((pterda# * q1#) / (qterda# * p1#)))
z9# = CDBL(LOG(qterda# / q1#))
inf1b# = R# * (pterda# * z8# + z9#)
```

```
z10# = CDBL(LOG((pterda# * q2#) / (qterda# * p2#)))
z11# = CDBL(LOG(qterda# / q2#))
inf2b# = R# * (pterda# * z10# + z11#)
a1bar# = z8# / inf1b#
a2bar# = z10# / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
z12# = CDBL(LOG(1# / bb#)) * (a2bar# / inf2b#)
z13# = CDBL(LOG(1# / aa#)) * (a1bar# / inf1b#)
nbar# = (z12# - z13#) / (a2bar# - a1bar#)
y = a1bar# / (a1bar# - a2bar#)
CALL NORTRY(y#, X#, d#, id)
rbar# = X#
sigmab# = SQR(R# * pterda# * qterda#)
uint = LOG(1 / aa#) / LOG(pterda# * q1# /
     (p1# * qterda#))
uslope = LOG(q1# / qterda#) / LOG(pterda# *
       q1# / (p1# * qterda#))
lslope = LOG(qterda# / q2#) / LOG(p2# * qterda# /
       (pterda# * q2#))
lint = LOG(bb#) / LOG(p2# * qterda# / (pterda# * q2#))
capm = (uint - lint) / (lslope - uslope)

LPRINT
LPRINT USING "NULL HYPOTHESIS : p1 = ###.###"; p1#
LPRINT USING "ALTERNATIVE HYPOTHESIS : p2 = ###.###"; p2#
LPRINT USING "ALPHA = #.##"; alpha#
LPRINT USING "BETA  = #.##"; beta#
LPRINT USING "THIRD HYPOTHESIS : p0 = ###.###"; pterda#
LPRINT USING "LOWER INTERCEPT FOR THE BOUNDARY :
              lint = ###.###"; lint
LPRINT USING "UPPER INTERCEPT FOR THE BOUNDARY :
              uint = ###.###"; uint
LPRINT USING "LOWER SLOPE FOR THE BOUNDARY :
              lslope = ###.###"; lslope
LPRINT USING "UPPER SLOPE FOR THE BOUNDARY :
              uslope = ###.###"; uslope
LPRINT USING "MAXIMUM SAMPLE SIZE FOR A DECISION :
              M = ###.###"; capm
LPRINT
BEEP

END SUB
```

```
SUB BIPROB (p0, pp( ), maxx)

pp(1, 1) = 1 - p0
pp(2, 1) = pp(1, 1)
pp(1, 2) = p0
pp(2, 2) = 1
maxx = 1

END SUB
```

```
FUNCTION BIFUNC# (p#)
SHARED p1#, p2#, alpha#, beta#, R#

q1# = 1 - p1#
q2# = 1 - p2#
q# = 1 - p#
bs1# = CDBL(LOG((p# * q1#) / (q# * p1#)))
bs2# = CDBL(LOG(q# / q1#))
binf1# = R# * (p# * bs1# + bs2#)
bs3# = CDBL(LOG((p# * q2#) / (q# * p2#)))
bs4# = CDBL(LOG(q# / q2#))
binf2# = R# * (p# * bs3# + bs4#)
ba1# = bs1# / binf1#
ba2# = bs3# / binf2#
capA1# = ba1# / ((ba1# - ba2#) * alpha#)
capA2# = ba2# / ((ba2# - ba1#) * beta#)
BIFUNC# = CDBL(LOG(capA2#)) * binf1# -
          CDBL(LOG(capA1#)) * binf2#

END FUNCTION
```

```
SUB ZEROBI (b#, C#, abser#, reler#, iflag) STATIC

maxit = 500
    rzero# = 0
    runit# = 1
    rtwo# = 2
    reight# = 8
    u# = runit#
100 u# = u# / rtwo#
    p# = runit# + u#
    IF (p# - runit#) > 0 THEN
        GOTO 100
    ELSE
        GOTO 101
    END IF
101 u# = u# * rtwo#
    re# = (reler# + u#) / rtwo# + ABS((reler# - u#)
          / rtwo#)
    ic = 0
    acbs# = CDBL(ABS(b# - C#))
    a# = C#
    fa# = BIFUNC#(a#)
    fb# = BIFUNC#(b#)
    fc# = fa#
    count = 2
    k1# = CDBL(ABS(fb#))
    k2# = CDBL(ABS(fc#))
    fx# = CDBL((k1# + k2#) / rtwo# + ABS((k1# - k2#)
          / rtwo#))
102 IF (k2# - k1#) < 0 THEN
        GOTO 103
    ELSE
        GOTO 104
    END IF
103 a# = b#
    fa# = fb#
    b# = C#
    fb# = fc#
    C# = a#
    fc# = fa#
104 cmb# = (C# - b#) / rtwo#
    acmb# = CDBL(ABS(cmb#))
    tol# = re# * CDBL(ABS(b#)) + abser#
    IF (acmb# - tol#) > 0 THEN
        GOTO 105
    ELSE
        GOTO 119
    END IF
105 IF (count - maxit) < 0 THEN
        GOTO 106
    ELSE
        GOTO 125
    END IF
```

```
106 p# = (b# - a#) * fb#
    q# = 1 - p#
    IF p < 0 THEN
        GOTO 107
    ELSE
        GOTO 108
    END IF
107 p# = -1 * p#
    q# = -1 * q#
108 a# = b#
    fa# = fb#
    ic = ic + 1
    IF (ic - 4) < 0 THEN
        GOTO 111
    ELSE
        GOTO 109
    END IF
109 IF (reight# * acmb# - acbs#) < 0 THEN
        GOTO 110
    ELSE
        GOTO 115
    END IF
110 ic = 0
    acbs# = acmb#
111 IF (p# - CDBL(ABS(q#)) * tol#) > 0 THEN
        GOTO 113
    ELSE
        GOTO 112
    END IF
112 IF cmb# < 0 THEN
        b# = b# - CDBL(ABS(tol#))
    ELSE
        b# = b# + CDBL(ABS(tol#))
    END IF
    GOTO 116
113 IF (p# - cmb# * q#) < 0 THEN
        GOTO 114
    ELSE
        GOTO 115
    END IF
114 b# = b# + p# / q#
    GOTO 116
115 b# = (C# + b#) / rtwo#
116 fb# = BIFUNC#(b#)
    IF fb# = 0 THEN
        GOTO 122
    ELSE
        GOTO 117
    END IF
```

```
117 count = count + 1
    IF fb# < 0 THEN
        k3# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k3# = CDBL(ABS(runit#))
    END IF
    IF fc# < 0 THEN
        k4# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k4# = CDBL(ABS(runit#))
    END IF
    IF (k3# - k4#) = 0 THEN
        GOTO 118
    ELSE
        GOTO 102
    END IF
118 C# = a#
    fc# = fa#
    GOTO 102
119 IF fb# < 0 THEN
        k3# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k3# = CDBL(ABS(runit#))
    END IF
    IF fc# < 0 THEN
        k4# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k4# = CDBL(ABS(runit#))
    END IF
    IF (k3# - k4#) = 0 THEN
        GOTO 124
    ELSE
        GOTO 120
    END IF
120 IF (CDBL(ABS(fb#)) - fx#) > 0 THEN
        GOTO 123
    ELSE
        GOTO 121
    END IF
121 iflag = 1
    GOTO 126
122 iflag = 2
    GOTO 126
123 iflag = 3
    GOTO 126
124 iflag = 4
    GOTO 126
125 iflag = 5

126 END SUB
```

```
SUB BIEXACT (p1#, p2#, alpha#, beta#, lint, uint,
             lslope, uslope, pp( ), maxx)

R# = 1
q1# = 1# - p1#
q2# = 1# - p2#
'program starts to calculate pterda for
'the keifer-weiss problem
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001

'call subprogram ZEROBI to obtain pstar
CALL ZEROBI(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
    pstar# = b#
ELSE
    ee = 1
END IF
qstar# = 1 - pstar#
z1# = CDBL(LOG((pstar# * q1#) / (qstar# * p1#)))
z2# = CDBL(LOG(qstar# / q1#))
inf1s# = R# * (pstar# * z1# + z2#)
z3# = CDBL(LOG((pstar# * q2#) / (qstar# * p2#)))
z4# = CDBL(LOG(qstar# / q2#))
inf2s# = R# * (pstar# * z3# + z4#)
a1star# = z1# / inf1s#
a2star# = z3# / inf2s#
y# = a1star# / (a1star# - a2star#)

'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
rstar# = X#
sigmas# = SQR(R# * pstar# * qstar#)
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
z5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
z6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (z5# - z6#) / (a2star# - a1star#)
z7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = (pstar# / qstar# * CDBL(EXP(z7#))) /
          (1# + (pstar# / qstar# * CDBL(EXP(z7#))))
qterda# = 1# - pterda#

'find nbar by replacing pstar by pterda
z8# = CDBL(LOG((pterda# * q1#) / (qterda# * p1#)))
z9# = CDBL(LOG(qterda# / q1#))
inf1b# = R# * (pterda# * z8# + z9#)
```

```
z10# = CDBL(LOG((pterda# * q2#) / (qterda# * p2#)))
z11# = CDBL(LOG(qterda# / q2#))
inf2b# = R# * (pterda# * z10# + z11#)
a1bar# = z8# / inf1b#
a2bar# = z10# / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
z12# = CDBL(LOG(1# / bb#)) * (a2bar# / inf2b#)
z13# = CDBL(LOG(1# / aa#)) * (a1bar# / inf1b#)
nbar# = (z12# - z13#) / (a2bar# - a1bar#)
y = a1bar# / (a1bar# - a2bar#)
CALL NORTRY(y#, X#, d#, id)
rbar# = X#
sigmab# = SQR(R# * pterda# * qterda#)
uint = LOG(1 / aa#) / LOG(pterda# * q1# /
       (p1# * qterda#))
uslope = LOG(q1# / qterda#) / LOG(pterda# *
         q1# / (p1# * qterda#))
lslope = LOG(qterda# / q2#) / LOG(p2# * qterda#
         / (pterda# * q2#))
lint = LOG(bb#) / LOG(p2# * qterda# / (pterda# * q2#))
capm = (uint - lint) / (lslope - uslope)

LPRINT "EXACT VALUES FOR THE BINOMIAL 2-SPRT"
LPRINT
LPRINT USING "p1 = ###.###"; p1#
LPRINT USING "p2 = ###.###"; p2#
LPRINT USING "alpha = #.##"; alpha#
LPRINT USING "beta = #.##"; beta#
LPRINT
LPRINT "     MU        OC       POWER      ASN       N95"
LPRINT "-------------------------------------------------"
LPRINT

FOR i = 1 TO 3
SELECT CASE i
CASE 1
     inc = p1# / 5
     bot = inc
     top = p1# + .0001
CASE 2
     inc = (p2# - p1#) / 10
     bot = p1# + inc
     top = p2# + .0001
CASE 3
     inc = (1 - p2#) / 5
     bot = p2# + inc
     top = 1 - inc
END SELECT
```

```
FOR p0 = bot TO top STEP inc
    mu = p0
CALL BIPROB(p0, pp( ), maxx)
CALL POW(p0, mu, lint, uint, lslope, uslope,
         alpha#, beta#, pp( ), maxx)
NEXT p0
NEXT i
BEEP

END SUB
```

```
SUB BISHEET (p1#, p2#, alpha#, beta#)

R# = 1
q1# = 1# - p1#
q2# = 1# - p2#
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001

'call subprogram ZEROBI to obtain pstar
CALL ZEROBI(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
    pstar# = b#
ELSE
    ee = 1
END IF
qstar# = 1 - pstar#
z1# = CDBL(LOG((pstar# * q1#) / (qstar# * p1#)))
z2# = CDBL(LOG(qstar# / q1#))
inf1s# = R# * (pstar# * z1# + z2#)
z3# = CDBL(LOG((pstar# * q2#) / (qstar# * p2#)))
z4# = CDBL(LOG(qstar# / q2#))
inf2s# = R# * (pstar# * z3# + z4#)
a1star# = z1# / inf1s#
a2star# = z3# / inf2s#
y# = a1star# / (a1star# - a2star#)

'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
rstar# = X#
sigmas# = SQR(R# * pstar# * qstar#)
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
z5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
z6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (z5# - z6#) / (a2star# - a1star#)
z7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = (pstar# / qstar# * CDBL(EXP(z7#))) /
          (1# + (pstar# / qstar# * CDBL(EXP(z7#))))
qterda# = 1# - pterda#

'find nbar by replacing pstar by pterda
z8# = CDBL(LOG((pterda# * q1#) / (qterda# * p1#)))
z9# = CDBL(LOG(qterda# / q1#))
inf1b# = R# * (pterda# * z8# + z9#)
z10# = CDBL(LOG((pterda# * q2#) / (qterda# * p2#)))
z11# = CDBL(LOG(qterda# / q2#))
```

```
inf2b# = R# * (pterda# * z10# + z11#)
a1bar# = z8# / inf1b#
a2bar# = z10# / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
z12# = CDBL(LOG(1# / bb#)) * (a2bar# / inf2b#)
z13# = CDBL(LOG(1# / aa#)) * (a1bar# / inf1b#)
nbar# = (z12# - z13#) / (a2bar# - a1bar#)
y = a1bar# / (a1bar# - a2bar#)
CALL NORTRY(y#, X#, d#, id)
rbar# = X#
sigmab# = SQR(R# * pterda# * qterda#)
uint = LOG(1 / aa#) / LOG(pterda# * q1# /
       (p1# * qterda#))
uslope = LOG(q1# / qterda#) / LOG(pterda# * q1# /
         (p1# * qterda#))
lslope = LOG(qterda# / q2#) / LOG(p2# * qterda# /
         (pterda# * q2#))
lint = LOG(bb#) / LOG(p2# * qterda# / (pterda# * q2#))
capm = (uint - lint) / (lslope - uslope)
LI = lint
LS = lslope
UI = uint
US = uslope
MA = capm

LPRINT
LPRINT TAB(25); "         BINOMIAL 2-SPRT"
LPRINT
LPRINT TAB(30); "  p1 ="; p1#; "    p2 ="; p2#
LPRINT TAB(28); " alpha ="; alpha#; "  beta ="; beta#
CALL SHEET(LI, LS, UI, US, MA)

END SUB
```

```
SUB POISSON (p1#, p2#, alpha#, beta#)

300 CLS
    LOCATE 5, 15

    PRINT
    PRINT " ENTER VALUES FOR THE FOLLOWING PARAMETERS:"
    PRINT " ( LAMBDA 2 MUST BE GREATER THAN LAMBDA 1 )"
    INPUT "         FOR NULL HYPOTHESIS : LAMBDA 1 = ",
                                                  p1#
    INPUT " FOR ALTERNATIVE HYPOTHESIS : LAMBDA 2 = ",
                                                  p2#
    INPUT "                                ALPHA = ", alpha#
    INPUT "                                 BETA = ", beta#
    PRINT
    PRINT

310 CLS
    LOCATE 5, 15
    PRINT " AT THIS TIME YOU MAY WANT TO:"
    PRINT
    PRINT " (1) MAKE CHANGES"
    PRINT " (2) COMPUTE VALUES FOR 2-SPRT DECISION BOUNDARIES"
    PRINT " (3) COMPUTE THE EXACT VALUES (OC AND ASN FUNCTION)"
    PRINT " (4) COMCPUTE THE OUTPUT DATA SHEET"
    PRINT " (5) EXIT"

    CH$ = INPUT$(1)
    SELECT CASE CH$
    CASE "1"
        GOTO 300
    CASE "2"
        CALL PO2SPRT(p1#, p2#, alpha#, beta#)
    CASE "3"
        CALL POEXACT(p1#, p2#, alpha#, beta#, lint, uint,
                     lslope, uslope, pp(), maxx)
    CASE "4"
        CALL POSHEET(p1#, p2#, alpha#, beta#)
    CASE "5"
        GOTO 360
    CASE ELSE
        BEEP: GOTO 310
    END SELECT
```

```
330 CLS
    LOCATE 6, 15
    PRINT "CHOOSE ONE OF THE FOLLOWING:"
    PRINT "(1)   MAKE ANOTHER COMPUTATION"
    PRINT "(2)   DO ANALYSIS FROM ANOTHER DISTRIBUTION"
    PRINT "(3)   EXIT"

340 E$ = INPUT$(1)
    SELECT CASE E$
    CASE "1"
        GOTO 310
    CASE "2"
        GOTO 360
    CASE "3"
        GOTO 360
    CASE ELSE
        BEEP: GOTO 330
    END SELECT

360 END SUB
```

```
SUB PO2SPRT (p1#, p2#, alpha#, beta#)

'program starts to calculate lambd-terda
'for the keifer-weiss problem
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001

'call subprogram ZEROPO to obtain pstar
CALL ZEROPO(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
'pstar is lambda star
    pstar# = b#
ELSE
    ee = 1
END IF
z1# = CDBL(LOG(pstar# / p1#))
z2# = CDBL(LOG(pstar# / p2#))
inf1s# = pstar# * (z1# - 1#) + p1#
inf2s# = pstar# * (z2# - 1#) + p2#
a1star# = z1# / inf1s#
a2star# = z2# / inf2s#
y# = a1star# / (a1star# - a2star#)

'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
'rstar is mustar
rstar# = X#
sigmas# = pstar#
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
z5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
z6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (z5# - z6#) / (a2star# - a1star#)
z7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = pstar# * CDBL(EXP(z7#))
'find nbar by replacing pstar by pterda
z8# = CDBL(LOG(pterda# / p1#))
z9# = CDBL(LOG(pterda# / p2#))
inf1b# = pterda# * (z8# - 1#) + p1#
inf2b# = pterda# * (z9# - 1#) + p2#
a1bar# = z8# / inf1b#
a2bar# = z9# / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
z12# = CDBL(LOG(1# / bb#)) * (a2bar# / inf2b#)
z13# = CDBL(LOG(1# / aa#)) * (a1bar# / inf1b#)
```

```
nbar# = (z12# - z13#) / (a2bar# - a1bar#)
y# = a1bar# / (a1bar# - a2bar#)
CALL NORTRY(y#, X#, d#, id)
rbar# = X#
sigmab# = pterda#
uint = LOG(1 / aa#) / LOG(pterda# / p1#)
uslope = (pterda# - p1#) / LOG(pterda# / p1#)
lslope = (p2# - pterda#) / LOG(p2# / pterda#)
lint = LOG(bb#) / LOG(p2# / pterda#)
capm = (uint - lint) / (lslope - uslope)

LPRINT USING "NULL HYPOTHESIS : LAMBDA 1
             = ###.###"; p1#
LPRINT USING "ALTERNATIVE HYPOTHESIS : LAMBDA 2
             = ###.###"; p2#
LPRINT USING "ALPHA = #.##"; alpha#
LPRINT USING "BETA = #.##"; beta#
LPRINT USING "THIRD HYPOTHESIS : LAMBDA TERDA
             = ###.###"; pterda#
LPRINT USING "LOWER INTERCEPT FOR THE BOUNDARY :
             lint = ###.###"; lint
LPRINT USING "UPPER INTERCEPT FOR THE BOUNDARY :
             uint = ###.###"; uint
LPRINT USING "LOWER SLOPE FOR THE BOUNDARY :
             lslope = ###.###"; lslope
LPRINT USING "UPPER SLOPE FOR THE BOUNDARY :
             uslope = ###.###"; uslope
LPRINT USING "MAXIMUM SAMPLE SIZE FOR A DECISION :
             M = ###.###"; capm

BEEP
END SUB
```

```
FUNCTION POFUNC# (p#)

SHARED p1#, p2#, alpha#, beta#
'p1# is lambda1, p2# is lambda2, p# is lambda
ps1# = CDBL(LOG(p# / p1#))
ps2# = CDBL(LOG(p# / p2#))
pinf1# = p# * (ps1# - 1#) + p1#
pinf2# = p# * (ps2# - 1#) + p2#
pa1# = ps1# / pinf1#
pa2# = ps2# / pinf2#
capA1# = pa1# / ((pa1# - pa2#) * alpha#)
capA2# = pa2# / ((pa2# - pa1#) * beta#)
POFUNC# = CDBL(LOG(capA2#)) * pinf1# -
          CDBL(LOG(capA1#)) * pinf2#

END FUNCTION
```

```
SUB POPROB (p0, pp(), maxx)

pp(1, 1) = EXP(-1 * p0)
pp(2, 1) = pp(1, 1)
FOR i = 2 TO 200
pp(1, i) = pp(1, i - 1) * p0 / (i - 1)
pp(2, i) = pp(2, i - 1) + pp(1, i)
IF pp(2, i) > .9999 THEN GOTO 700
NEXT i
700 maxx = i - 1

END SUB
```

```
SUB ZEROPO (b#, C#, abser#, reler#, iflag) STATIC

maxit = 500
    rzero# = 0
    runit# = 1
    rtwo# = 2
    reight# = 8
    u# = runit#
400 u# = u# / rtwo#
    p# = runit# + u#
    IF (p# - runit#) > 0 THEN
        GOTO 400
    ELSE
        GOTO 401
    END IF
401 u# = u# * rtwo#
    re# = (reler# + u#) / rtwo# + ABS((reler# - u#)
          / rtwo#)
    ic = 0
    acbs# = CDBL(ABS(b# - C#))
    a# = C#
    fa# = POFUNC#(a#)
    fb# = POFUNC#(b#)
    fc# = fa#
    count = 2
    k1# = CDBL(ABS(fb#))
    k2# = CDBL(ABS(fc#))
    fx# = CDBL((k1# + k2#) / rtwo# + ABS((k1# - k2#)
          / rtwo#))
402 IF (k2# - k1#) < 0 THEN
        GOTO 403
    ELSE
        GOTO 404
    END IF
403 a# = b#
    fa# = fb#
    b# = C#
    fb# = fc#
    C# = a#
    fc# = fa#
404 cmb# = (C# - b#) / rtwo#
    acmb# = CDBL(ABS(cmb#))
    tol# = re# * CDBL(ABS(b#)) + abser#
    IF (acmb# - tol#) > 0 THEN
        GOTO 405
    ELSE
        GOTO 419
    END IF
405 IF (count - maxit) < 0 THEN
        GOTO 406
    ELSE
        GOTO 425
    END IF
```

```
406 p# = (b# - a#) * fb#
    q# = 1 - p#
    IF p < 0 THEN
        GOTO 407
    ELSE
        GOTO 408
    END IF
407 p# = -1 * p#
    q# = -1 * q#
408 a# = b#
    fa# = fb#
    ic = ic + 1
    IF (ic - 4) < 0 THEN
        GOTO 411
    ELSE
        GOTO 409
    END IF
409 IF (reight# * acmb# - acbs#) < 0 THEN
        GOTO 410
    ELSE
        GOTO 415
    END IF
410 ic = 0
    acbs# = acmb#
411 IF (p# - CDBL(ABS(q#)) * tol#) > 0 THEN
        GOTO 413
    ELSE
        GOTO 412
    END IF
412 IF cmb# < 0 THEN
        b# = b# - CDBL(ABS(tol#))
    ELSE
        b# = b# + CDBL(ABS(tol#))
    END IF
    GOTO 416
413 IF (p# - cmb# * q#) < 0 THEN
        GOTO 414
    ELSE
        GOTO 415
    END IF
414 b# = b# + p# / q#
    GOTO 416
415 b# = (C# + b#) / rtwo#
416 fb# = POFUNC#(b#)
    IF fb# = 0 THEN
        GOTO 422
    ELSE
        GOTO 417
    END IF
```

```
417 count = count + 1
    IF fb# < 0 THEN
        k3# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k3# = CDBL(ABS(runit#))
    END IF
    IF fc# < 0 THEN
        k4# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k4# = CDBL(ABS(runit#))
    END IF
    IF (k3# - k4#) = 0 THEN
        GOTO 418
    ELSE
        GOTO 402
    END IF
418 C# = a#
    fc# = fa#
    GOTO 402
419 IF fb# < 0 THEN
        k3# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k3# = CDBL(ABS(runit#))
    END IF
    IF fc# < 0 THEN
        k4# = CDBL(ABS(runit#)) * (-1)
    ELSE
        k4# = CDBL(ABS(runit#))
    END IF
    IF (k3# - k4#) = 0 THEN
        GOTO 424
    ELSE
        GOTO 420
    END IF
420 IF (CDBL(ABS(fb#)) - fx#) > 0 THEN
        GOTO 423
    ELSE
        GOTO 421
    END IF
421 iflag = 1
    GOTO 426
422 iflag = 2
    GOTO 426
423 iflag = 3
    GOTO 426
424 iflag = 4
    GOTO 426
425 iflag = 5

426 END SUB
```

```
SUB POEXACT (p1#, p2#, alpha#, beta#, lint, uint,
             lslope, uslope, pp( ), maxx)

'program starts to calculate lambda-terda for
'the keifer-weiss problem
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001

'call subprogram ZEROPO to obtain pstar
CALL ZEROPO(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
'pstar is lambda star
    pstar# = b#
ELSE
    ee = 1
END IF
z1# = CDBL(LOG(pstar# / p1#))
z2# = CDBL(LOG(pstar# / p2#))
inf1s# = pstar# * (z1# - 1#) + p1#
inf2s# = pstar# * (z2# - 1#) + p2#
a1star# = z1# / inf1s#
a2star# = z2# / inf2s#
y# = a1star# / (a1star# - a2star#)

'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
'rstar is mustar
rstar# = X#
sigmas# = pstar#
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
z5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
z6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (z5# - z6#) / (a2star# - a1star#)
z7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = pstar# * CDBL(EXP(z7#))
'find nbar by replacing pstar by pterda
z8# = CDBL(LOG(pterda# / p1#))
z9# = CDBL(LOG(pterda# / p2#))
inf1b# = pterda# * (z8# - 1#) + p1#
inf2b# = pterda# * (z9# - 1#) + p2#
a1bar# = z8# / inf1b#
a2bar# = z9# / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
```

```
z12# = CDBL(LOG(1# / bb#)) * (a2bar# / inf2b#)
z13# = CDBL(LOG(1# / aa#)) * (a1bar# / inf1b#)
nbar# = (z12# - z13#) / (a2bar# - a1bar#)
y = a1bar# / (a1bar# - a2bar#)
CALL NORTRY(y#, X#, d#, id)
rbar# = X#
sigmab# = pterda#
uint = LOG(1 / aa#) / LOG(pterda# / p1#)
uslope = (pterda# - p1#) / LOG(pterda# / p1#)
lslope = (p2# - pterda#) / LOG(p2# / pterda#)
lint = LOG(bb#) / LOG(p2# / pterda#)
capm = (uint - lint) / (lslope - uslope)

'print the exact properties for the poisson 2-sprt
LPRINT "EXACT PROPERTIES FOR THE POISSON 2-SPRT"
LPRINT
LPRINT USING "LAMBDA 1 = ###.###"; p1#
LPRINT USING "LAMBDA 2 = ###.###"; p2#
LPRINT USING "ALPHA  = #.##"; alpha#
LPRINT USING "BETA = #.##"; beta#
LPRINT
LPRINT "     MU        OC       POWER     ASN       N95"
LPRINT "-----------------------------------------------"
LPRINT

FOR i = 1 TO 3
SELECT CASE i
CASE 1
     inc = p1# / 5
     bot = inc
     top = p1# + .0001
CASE 2
     inc = (p2# - p1#) / 10
     bot = p1# + inc
     top = p2# + .0001
CASE 3
     inc = (1 - p2#) / 5
     bot = p2# + inc
     top = 1 - inc
BEEP
END SELECT

FOR p0 = bot TO top STEP inc
    mu = p0
    CALL POPROB(p0, pp(), maxx)
CALL POW(p0, mu, lint, uint, lslope, uslope, alpha#,
         beta#, pp(), maxx)
NEXT p0
NEXT i

END SUB
```

```
SUB POSHEET (p1#, p2#, alpha#, beta#)

'program starts to calculate lambd-terda
'for the keifer-weiss problem
ee = 0
b# = p1# + .005
C# = p2# - .005
reler# = 0
abser# = .00001

'call subprogram ZEROPO to obtain pstar
CALL ZEROPO(b#, C#, abser#, reler#, iflag)
IF iflag = 1 OR iflag = 2 THEN
'pstar is lambda star
    pstar# = b#
ELSE
    ee = 1
END IF
z1# = CDBL(LOG(pstar# / p1#))
z2# = CDBL(LOG(pstar# / p2#))
inf1s# = pstar# * (z1# - 1#) + p1#
inf2s# = pstar# * (z2# - 1#) + p2#
a1star# = z1# / inf1s#
a2star# = z2# / inf2s#
y# = a1star# / (a1star# - a2star#)

'call subprogram NORTRY to get mustar
CALL NORTRY(y#, X#, d#, id)
'rstar is mustar
rstar# = X#
sigmas# = pstar#
ap# = (a1star# - a2star#) * alpha# / a1star#
bp# = (a2star# - a1star#) * beta# / a2star#
z5# = CDBL(LOG(1# / bp#)) * (a2star# / inf2s#)
z6# = CDBL(LOG(1# / ap#)) * (a1star# / inf1s#)
nstar# = (z5# - z6#) / (a2star# - a1star#)
z7# = rstar# / (sigmas# * SQR(nstar#))
pterda# = pstar# * CDBL(EXP(z7#))

'find nbar by replacing pstar by pterda
z8# = CDBL(LOG(pterda# / p1#))
z9# = CDBL(LOG(pterda# / p2#))
inf1b# = pterda# * (z8# - 1#) + p1#
inf2b# = pterda# * (z9# - 1#) + p2#
a1bar# = z8# / inf1b#
a2bar# = z9# / inf2b#
aa# = (a1bar# - a2bar#) * alpha# / a1bar#
bb# = (a2bar# - a1bar#) * beta# / a2bar#
```

```
z12# = [illegible](LOG(1# / bb#)) * (a2bar# / inf2b#)
z13# = [illegible](LOG(1# / aa#)) * (a1bar# / inf1b#)
nbar# = [illegible]z12# - z13#) / (a2bar# - a1bar#)
y = a1[illegible] / (a1bar# - a2bar#)
CALL [illegible]XY(y#, X#, d#, id)
rbar# = [illegible]
sigmab# = pterda#
uint = [illegible](1 / aa#) / LOG(pterda# / p1#)
uslope = [illegible]pterda# - p1#) / LOG(pterda# / p1#)
lslope = [illegible]p2# - pterda#) / LOG(p2# / pterda#)
lint = [illegible](bb#) / LOG(p2# / pterda#)
capm = [illegible]int - lint) / (lslope - uslope)

LI = li[illegible]
LS = la[illegible]pe
UI = ui[illegible]
US = ua[illegible]pe
MA = ca[illegible]
LPRINT
LPRINT [illegible]AB(28); "          POISSON 2-SPRT"
LPRINT
LPRINT [illegible]AB(30); "Lambda1 ="; p1#; "    Lambda2 ="; p2#
LPRINT [illegible]AB(32); "Alpha ="; alpha#; "      Beta ="; beta#
CALL [illegible]T(LI, LS, UI, US, MA)

END SUB
```

```
SUB sheet (LI, LS, UI, US, MA)
900 LPRINT
    LPRINT "-------------------------------------------------"
           -------------------------------"
    LPRINT "   SAMPLE"; TAB(13); "LOWER"; TAB(20); "RUNNING";
    LPRINT TAB(29); "UPPER"; TAB(43); "SAMPLE"; TAB(52);
    LPRINT "LOWER"; TAB(59); "RUNNING"; TAB(69); "UPPER"
    LPRINT "   NUMBER"; TAB(13); "BOUND"; TAB(20); "TOTAL";
    LPRINT TAB(29); "BOUND"; TAB(43); "NUMBER"; TAB(52);
    LPRINT "BOUND"; TAB(59); "TOTAL"; TAB(69); "BOUND"
    LPRINT "-------------------------------------------------"
           -------------------------------"
    LPRINT

    X = 51

    FOR R = 1 TO 50
        IF R < MA THEN
          LB1 = CLNG(LI + (R * LS))
          UB1 = CLNG(UI + (R * US))
        ELSE
          LB1 = 0
          UB1 = 0
        END IF
        IF R < MA AND X < MA THEN
          LB2 = CLNG(LI + (X * LS))
          UB2 = CLNG(UI + (X * US))
        ELSE
          LB2 = 0
          UB2 = 0
        END IF
905 LPRINT TAB(5); R; TAB(13); LB1; TAB(20); "_________";
           TAB(29); UB1;
    LPRINT TAB(44); X; TAB(52); LB2; TAB(59); "_________";
           TAB(69); UB2;
    X = X + 1
    NEXT R

910 END SUB
```

```
SUB NORTRY (p#, X#, d#, ie) STATIC

'subprogram "nortry" to find mustar
ie = 0
   X# = 99999999
   d# = X#
   IF p# < 0 THEN
       GOTO 1
   ELSEIF p# = 0 THEN
       GOTO 4
   ELSE
       GOTO 2
   END IF
1  ie = -1
   GOTO 13
2  IF (p# - 1#) < 0 THEN
       GOTO 7
   ELSEIF (p# - 1#) = 0 THEN
       GOTO 5
   ELSE
       GOTO 1
   END IF
4  X = -99999999
5  d# = 0#
   GOTO 13
7  d# = p#
   IF (d# - .5) > 0 THEN
       GOTO 8
   ELSE
       GOTO 9
   END IF
8  d# = 1# - d#
9  T2# = LOG(1# / (d# * d#))
   T# = SQR(T2#)
   first# = 2.515517 + .802853 * T# +
            .010328 * T2#
   second# = 1! + 1.432788 * T# + .189269 *
             T2# + .001308 * T# * T2#
   X# = T# - first# / second#
   IF (p# - .5) > 0 THEN
       GOTO 12
   ELSE
       GOTO 11
   END IF
11 X# = -1 * X#
12 d# = .3989432# * EXP(-1 * X * X / 2!)

13 END SUB
```

```
SUB POW (p0, mu, lint, uint, lslope, uslope, alpha#,
         beta#, pp( ), maxx)

DIM prob(3, 1000), T(2, 1000)
CLS
q0 = 1 - p0
FOR i = 1 TO 1000
FOR j = 1 TO 3
prob(j, i) = 0
NEXT j
NEXT i

llim1 = 0
ulim1 = 0
T(1, 1) = 1
lastn = 0
lastt = 1
FOR n = 1 TO 1000
llim2 = INT(lint + n * lslope + .99999)
IF llim2 < 0 THEN llim2 = 0
ulim2 = INT(uint + n * uslope)
IF lastt = 1 THEN
   prest = 2
ELSE
prest = 1
END IF

times = ulim2 - llim2 + 1
FOR i = 1 TO times
T(prest, i) = 0
NEXT i
FOR j = llim1 TO ulim1
unum = ulim2 - j
IF unum > maxx THEN unum = maxx
lnum = llim2 - j
IF lnum < 0 THEN lnum = 0
from = j - llim1 + 1
FOR i = lnum TO unum
index = i + j - llim2 + 1
T(prest, index) = T(prest, index) + T(lastt, from)
                  * pp(1, i + 1)
NEXT i
prob(2, n) = prob(2, n) + T(lastt, from) *
             (1 - pp(2, unum + 1))
IF lnum > 0 THEN
   prob(1, n) = prob(1, n) + T(lastt, from) * pp(2, lnum)
NEXT j
find = 0
```

```
FOR i = llim2 TO ulim2
chk = i - llim2 + 1
find = find + T(prest, chk)
NEXT i
prob(3, n) = prob(1, n) + prob(2, n)
IF find < .00001 THEN GOTO 21
ulim1 = ulim2
llim1 = llim2
lastt = prest
NEXT n
maxn = 1000
GOTO 22
LOCATE 8, 12
21 PRINT n
maxn = n
22 oc = 0
power = 0
EN = 0
chk = 0
find = 0
FOR i = 1 TO maxn
oc = oc + prob(1, i)
power = power + prob(2, i)
EN = EN + prob(3, i) * i
find = find + prob(3, i)
IF find < .95 THEN GOTO 30
IF chk = 0 THEN n95 = i
chk = 1
30 NEXT i
LOCATE 11, 12
LPRINT USING "#####.###"; mu; oc; power; EN; n95
'print oc,power,en,n95

END SUB
```

VITA

Siew Joo Lim

Candidate for the Degree of

Master of Science

Thesis: A COMPUTER PROGRAM TO DEVELOP AND EVALUATE A 2-SPRT FOR THE NEGATIVE BINOMIAL, BINOMIAL, AND POISSON DISTRIBUTION

Major Field: Statistics

Biographical:

Personal Data: Born in Malacca, Malaysia, Oct. 31, 1964. Daughter o f Siak Chiew Lim and Ah Niah Tan.

Education: Recieved Bachelor of Science in Business Administration with a double major in Statistics from Oklahoma State University, Oklahoma in December, 1987; completed requirements for Master of Science degree at Oklahoma State University, in December, 1989.

Professional Experience: Graduate Teaching Asistance, Department of Statistics, Oklahoma State University, August, 1988, to May, 1989.