Handbook of Sampling Methods for Arthropods in Agriculture

Edited By

Larry P. Pedigo
Department of Entomology
Iowa State University
Ames, Iowa

and

G. David Buntin
Department of Entomology
Georgia Experiment Station
Griffin, Georgia

Chapter 11. Sampling to Predict or Monitor Biological Control.
Jan P. Nyrop and Wopke van der Werf

CRC Press
Boca Raton Ann Arbor London Tokyo

Chapter 11

SAMPLING TO PREDICT OR MONITOR BIOLOGICAL CONTROL

Jan P. Nyrop and Wopke van der Werf

TABLE OF CONTENTS

In: Handbook of sampling methods for arthropods in agriculture
Larry P. Pedigo & G. David Buntin (Eds)

ISBN 0-8493-2923-X/94/$0.00 + .50

I. INTRODUCTION

A primary objective of integrated pest management is to move from crop protection systems that rely on broad spectrum pesticides to more environmentally benign systems built upon target-specific chemical management tools, host plant resistance, cultural practices, and biological control. Sampling and pest control decision making are foundations of most integrated pest management systems. Well-designed pest control decision rules can reduce the need for prophylactic chemical pesticide use and thereby assist in establishing environments wherein biological control agents can survive.

When making decisions about the need for pesticide use, it often suffices to estimate or classify pest density at a single point in time. If the phenology of a population is uncertain, it may be necessary to repeat sampling; however, a decision is still sought concerning pest density over a relatively narrow time span. Most currently used pest control decision rules are designed to meet this need.

With greater reliance on biological control, desired information on population density changes. While it is still necessary to ascertain that density remains below some critical level where economic injury occurs, it usually will be necessary to either make this determination many times during some period of interest, say a growing season, or to make a prediction about it. Conventional classification or estimation sampling procedures can be used to repeatedly sample a population through time; however, this usually will not be an effective use of sampling resources. In this chapter, we present methods that can be used to either predict, based on the ratio of pest to natural enemy, that a population will remain below a critical level, or to parsimoniously sample a population through time to answer the same question.

Little work has been done in the area of sampling for predicting or monitoring biological control.[1] We describe one method that can be used for prediction and two that can be used for monitoring. The first procedure is based on using the ratio of pests to natural enemies to predict the likelihood for biological control. Sequential methods for classifying pest-natural enemy ratios have previously been described.[2] Here, we review this work and present a FORTRAN computer program that can be used to design and analyze these sampling plans. The second and third methods are based on sampling a population repeatedly through time by using information on expected population growth along with sample information on present density to schedule future sampling. FORTRAN computer programs for designing and analyzing sampling protocols founded on this approach are also provided.

We will illustrate the methods using data that describe sampling distributions for European red mite, *Panonychus ulmi* (Koch), and its predator *Typhlodromus pyri* (Scheuten) in New York apples.[1,3]

II. PREDICTING BIOLOGICAL CONTROL VIA PREY-NATURAL ENEMY RATIOS

The ratio of pest to natural enemy, R, can sometimes be used to determine whether natural enemies are sufficiently abundant to control pest population growth.[4] If R is less than some critical ratio CR, value, it is assumed biological control will ensue. In addition to the ratio R, pest density, m_N, in relation to an intervention threshold, T, may also be of interest. If m_N exceeds T, some immediate action may be required. The critical ratio might also be conditional on m_N so that if m_N exceeds T, biological control might not occur even if $R \leq$ CR.

Consideration of R in relation to CR and m_N in relation to T results in four regions into which the pest and natural enemy populations can be jointly classified

(Figure 1). In region 1, $R \leq \text{CR}$ and $m_N \leq T$. Under these conditions biological control is likely to occur. In region 2, $R \leq \text{CR}$ and $m_N > T$. As a result, the pest may be regulated by natural enemies; however, significant plant damage is likely to occur before this happens. In region 3, $R > \text{CR}$ and $m_N \leq T$. With these conditions biological control may not occur; however, intervention is not needed now and another sample should be taken at a later date. Finally, in region 4 $R > \text{CR}$ and $m_N > T$. Under these conditions immediate intervention is necessary.

A sequential procedure for simultaneously classifying R and m_N can be constructed using confidence limits about estimates of the ratio and pest density. Stop

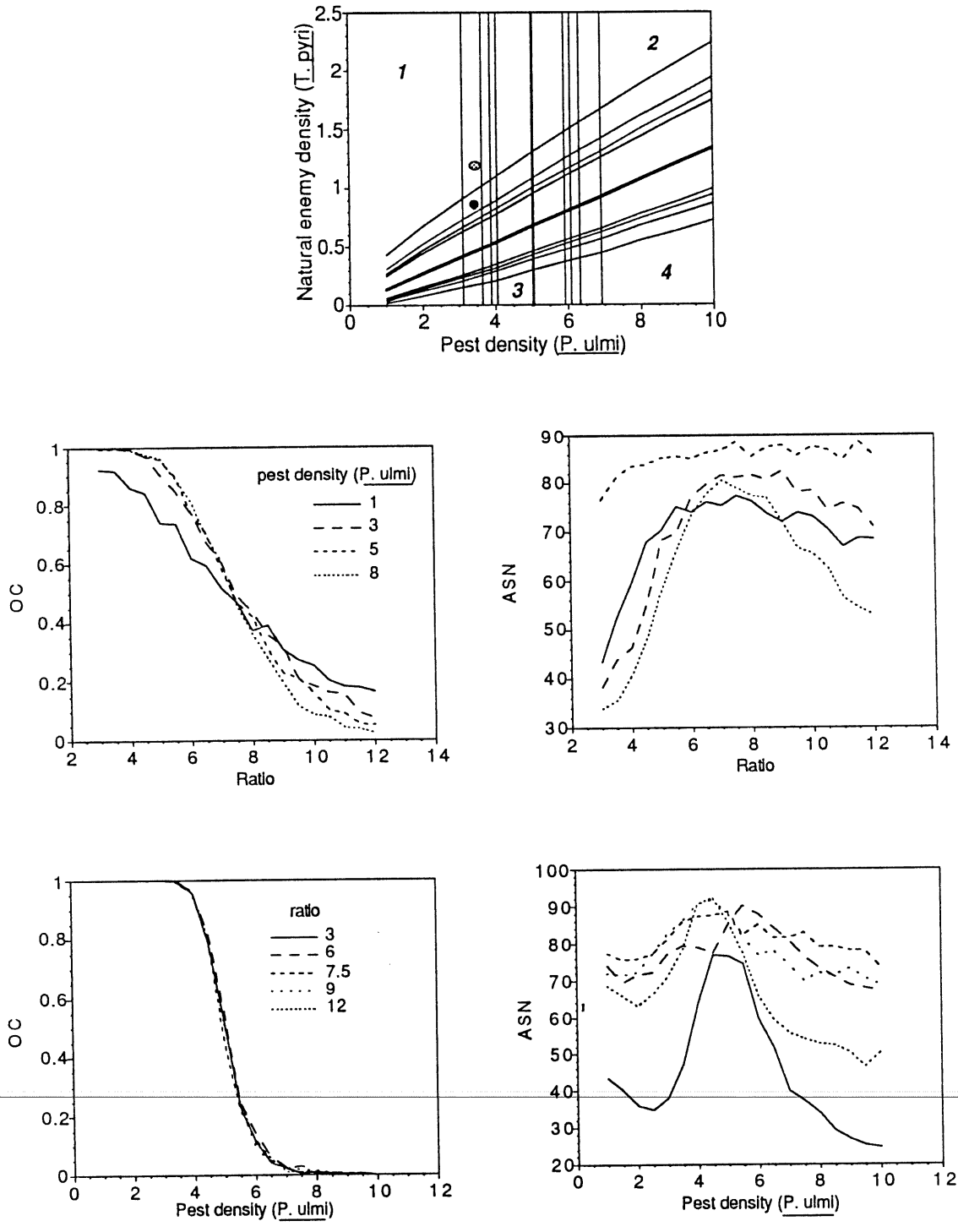

FIGURE 1. Stop lines and operating characteristic (OC) and average sample number (ASN) functions for a sequential ratio classification sampling plan.

limits are comprised of two sets, one with which the estimated prey density is compared and one with which the estimated predator density is compared. The stop limits for the prey are based on confidence limits about T. If an estimated density exceeds the upper confidence limit, sampling is terminated, and the population classified as $> T$. Conversely, if the estimated density is less than the lower confidence limit, density is classified as $\leq T$. Sampling continues until one of the stop boundaries is crossed. Binns (Chapter 8) provides a complete description of these stop lines.

Stop limits for the ratio classification work as follows: from a sample of size n_i the pest (m_N) and natural enemy (m_D) densities are estimated. The subscripts N and D denote numerator and denominator of the ratio. Both the sample size and m_N are used to determine the upper and lower ratio stop limits (ULR and LLR). Classification decisions are made by comparing the natural enemy density to ULR and LLR according to the following rules: If $m_D <$ LLR stop sampling and classify R as $>$ CR. If $m_D >$ ULR stop and classify R as $\leq$ CR. If neither of these conditions are met, take another sample of n_i observations, calculate the means (m_N and m_D) based on all the observations, and repeat the comparison. The stop limits are the densities of natural enemies (denominator of the ratio) given sample size n and prey density m_N that produce confidence limits for an estimated ratio which are greater than (LLR) and less than (ULR), the critical ratio CR. It is important to note that when both ratio and prey (pest) density are being classified, stop boundaries for both parameters must be crossed before sampling is terminated. A batch sampling procedure where n_i samples are examined before estimated means are compared with stop lines is used for two reasons. First, without batch sampling an unwieldy number of stop limits must be created because there is a new set of limits for each sample size. Second, batch sampling with $n_i \geq 20$ is currently necessary for determination of the operating characteristic (OC) and average sample number (ASN) functions. These functions must be simulated and require generation of correlated bivariate random deviates. Normal bivariate deviates are easily computed and with $n_i \geq 20$ the assumption of normality should be reasonably robust. It is not clear how correlated bivariate random variables that follow some other distribution might be simulated. The entire procedure is truncated so that after a total of n_t observations, a classification is made by comparing the estimated ratio to CR and estimated pest density to T.

The equation

$$r\left[\left(1 - z_{\alpha/2}^2 C_{ND}\right) \pm \frac{z_{\alpha/2}^2\sqrt{(C_N + C_D - 2C_{ND}) - z_{\alpha/2}^2(C_N C_D - C_{ND}^2)}}{\left(1 - z_{\alpha/2}^2 C_D\right)}\right] \quad (1)$$

is used to compute ratio confidence limits.[5] In this equation the subscripts N and D again denote the numerator (pest) and denominator (natural enemy) of the ratio. The other variables are defined as follows: $C_N = s_N^2/(nm_N^2)$, $C_D = s_D^2/nm_D^2$, $C_{ND} = \rho_{ND} s_N s_D/(nm_D m_N)$, s^2 = the sample variance, m_N and m_D are the estimated means for the numerator (pest) and denominator (natural enemy), ρ_{ND} is an estimated correlation coefficient between the numerator and denominator, $r = m_N/m_D$, and $z_{\alpha/2}$ is a standard normal deviate such that $P(Z \leq z_{\alpha/2}) = \alpha/2$.

Calculation of stop limits using Equation 1 requires that the correlation and variances for the numerator and denominator are known or can be written in terms of the means and the sample size. The variances usually can be modeled precisely as a function of the respective means using Taylor's power law (TPL), $s^2 = am^b$.[6] It also may be possible to model the correlation as a function of the means, or more simply, it may be a constant.

With the correlation specified and the variances modeled in terms of the means, stop lines can be determined for any value of n and m_N. To calculate ULR, values for n and m_N are chosen, and a variable x is initially set to m_N/CR. The upper confidence limit for $R = m_N/x$ is calculated and compared to CR. If CR is greater than or equal to the confidence limit, ULR $= x$; otherwise, x is increased by some value d, the confidence limits are calculated for a new R, and the comparison is made again. This procedure is repeated until CR is greater than or equal to the upper confidence interval. If the LLR are to be calculated, the procedure is identical with the exception that a lower confidence limit is calculated, the comparison criterion is that CR is less than or equal to the lower confidence limits and x is decreased by some value d. By setting d to a small value (ca., 0.01) x's can be found so that the confidence intervals are approximately equal to CR. Program "ratio" performs these calculations (see Appendix).

OC and ASN functions are constructed using simulation. Because both the pest-natural enemy ratio and the pest density are simultaneously classified, there are two sets of OC and ASN functions. For the ratio, the null hypothesis is that there are sufficient natural enemies for biological control, and hence the OC is the probability of accepting this hypothesis given any true ratio and a fixed pest density. For the pest density, the null hypothesis is that the density is less than the intervention threshold and the OC is the probability of accepting this hypothesis given any true pest density and a fixed ratio. The two sets of OC and ASN functions can be studied two ways. First, sets of OC and ASN functions can be plotted where each member of the set is indexed by a ratio or pest density (Figure 1). Alternately, the probability of making an incorrect classification (PIC) and ASN given any true ratio and pest density can be determined and plotted using an x-y-z ordinate system (Figure 2). The PIC function is a convenient way of representing the probability of an incorrect decision; however, three-dimensional figures are often difficult to interpret. To construct any of these functions, sampling is simulated from two jointly distributed populations: one representing the natural enemies and one representing the pest. Means from n_i observations are assumed to be bivariate normally distributed with variances defined as s^2/n_i and s^2 modeled using TPL. The FORTRAN program "ratio" computes stop lines, ASN, OC, and PIC functions (see Appendix).

To illustrate the method we use information on the sampling distribution of *P. ulmi* and *T. pyri*. Parameters for TPL for *P. ulmi* are $a = 4.32$ and $b = 1.2$. The same parameters for *T. pyri* are 2.38 and 1.2, respectively. Correlation between counts of this pest and natural enemy are variable; however, a conservative estimate is -0.25. Confidence intervals for a ratio become wider as the correlation decreases. Therefore, when a range of correlation coefficients can apply, use of the smallest value will produce the most conservative stop lines. The CR for this system is approximately 7.5, and one intervention threshold for *P. ulmi* is 5.0 per leaf. Stop lines were constructed using z values of 1.28 for both the ratio and prey density limits. The batch sample size (n_i) was set to 20, the maximum sample size was 100, and d was set to 0.01.

Stop lines are shown in Figure 1. Vertical lines are used with the pest density and those that run at approximately 45° from the origin are used with the natural enemy density. There are four sets of thin lines for the prey and natural enemy. Each set corresponds to a sample size of 20, 40, 60, or 80, with the interval between the lines becoming narrower as the sample size increases. The thick line in the center of each set of stop lines is used if the maximum sample size is reached by comparing the estimated mean to it.

An example will clarify use of the stop lines. Suppose a sample of 20 leaves is taken resulting in estimated pest and predator means of 3.4 and 0.85, respectively. This

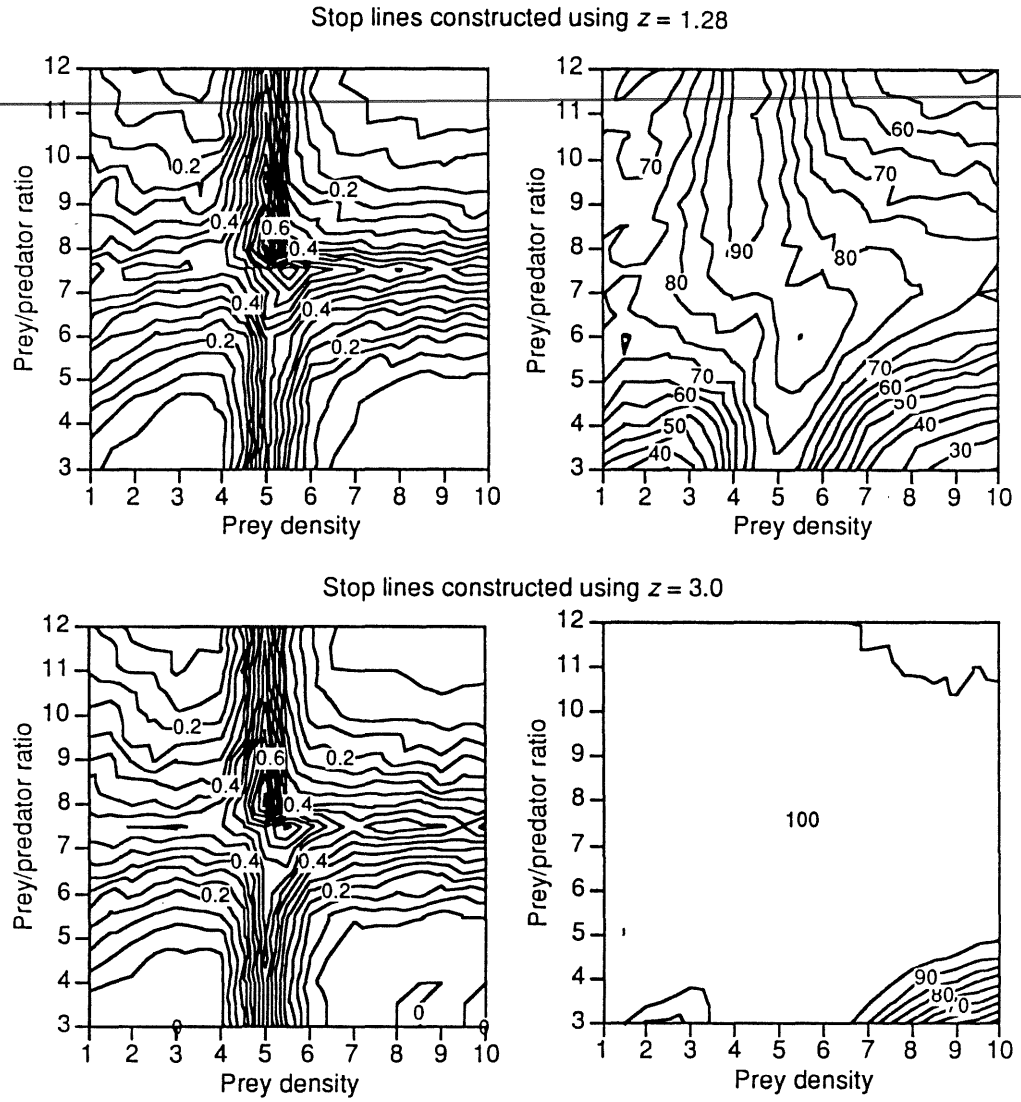

FIGURE 2. Probability of incorrect classification (PIC) and average sample number functions (ASN) for a ratio sequential classification sampling plan.

point is shown in Figure 1 as a dark circle; it lies to the right of the left-most stop limit for classifying pest density and below the uppermost limit for classifying the ratio. As a result, another sample must be taken. Suppose that after another sample of 20 leaves, the estimated density for pest and predator from the joint sample of 40 leaves was 3.5 and 1.2. This point is shown in Figure 1 as a shaded circle; it lies above the second stop line for the ratio (corresponding to a sample size of 40) and to the left of the second stop line for the pest. Sampling is terminated with the decision that the pest density is less than the intervention threshold and there are sufficient predators for biological control.

Also shown in Figure 1 are conditional OC and ASN functions for this sampling plan. Recall that the OC functions for the ratio are conditional on a particular pest density and OC functions for the pest are conditional on a ratio value. The OC functions for the ratio become steeper and improve at higher prey densities. This is because the variance of the ratio decreases with increasing prey density and because more samples are taken when the prey density is close to the intervention threshold. The OC functions for the prey are largely independent of the ratio; however, ASN values plotted in relation to prey density are greatly influenced by the ratio value. As expected, more samples are required to make a decision when the ratio is close to CR.

Shown in Figure 2 are three-dimensional plots of the ASN and PIC functions for the sampling plan described above as well as one where the z values were set to 3.0. Increasing z makes the stop limits wider. For $z = 3.0$, this results in a larger ASN and in many cases use of the maximum sample size of 100. The two sampling plans are compared to show that use of the sequential procedure with $z = 1.28$ results in approximately the same PIC as that for a plan based on $z = 3.0$ and with considerable savings in sample size.

The greatest drawback to using sequential ratio classification is that complete enumeration of counts is necessary. For many small organisms such as mites this is nearly impossible under field situations. As a result, sampling procedures for classifying or estimating densities of small, prolific organisms have been developed that substitute presence or absence of animals on sample units for enumeration. This also should be done for the ratio classification procedure and, if accomplished, would provide a tool that would be more readily used. It is straightforward to modify the stop lines so that they are expressed in terms of presence-absence counts. However, it is not clear how random variables that correctly model all sources of variation should be generated during simulations to determine performance of ratio classification schemes based on presence-absence counts.

III. MONITORING POPULATION DENSITY THROUGH TIME

The ratio of a pest to natural enemy cannot always be used as an index of the likelihood for biological control. At least three situations may result in this index being inappropriate or not usable. First, the CR, the ratio for which biological control can be expected, may not be constant with prey or predator density or other factors such as weather may influence this index. Second, more than one natural enemy species may be involved in the interaction and the relative species mix may influence the CR. For example, at least three phytoseiid predators can be found in commercial New York apple orchards (*T. pyri*, *T. longipilus*, and *A. fallacis*), as well as two stigmaeiid predators. This predator complex often provides effective biological control; however, a single CR for this complex is not tenable. Finally, it may not be practical to measure natural enemy density. Again, referring to the mite predator-prey (pest) system in apples, densities of predaceous mites can be estimated, although it is often difficult to do so in the field because phytoseiid predator mites superficially resemble their prey and are often concealed along the midrib of leaves. In other situations predators may be difficult to find or may be very mobile. When pest and natural enemy densities cannot both be estimated, ratios cannot be used, and another method for measuring the effectiveness of biological control is required.

One solution is to estimate or classify pest population density through time, and as long as the density remains below a threshold density that dictates intervention, biological control is effective. Using this scenario, it is not necessary to estimate or classify natural enemy density. However, it is often not practical or necessary to sample a pest population frequently through time. If, for example, a pest's density was much less than an intervention threshold, the population should not have to be sampled again as soon as a population whose density was only slightly less than an intervention threshold. What is needed is a sampling scheme that allows rapid classification of population density at a particular point in time, and that provides a measure of when the population should be resampled if the current density is less than the intervention threshold.

If knowledge of population growth is available, sample information can be combined with this knowledge to forecast future density. Such forecasts can be used as a

basis for scheduling future samples. Wilson[7] developed such a system that used estimates of pests and natural enemies and nonlinear regression to determine when the next sample should be taken. Here, we present two methods based on sequential classification of density and simple forecasts of population growth. Each of these methods can be viewed as belonging to a family of potential monitoring protocols that do two things. First, these protocols determine whether a pest density exceeds an intervention threshold. Second, if the density is less than the threshold, the sampling protocols determine how long one can wait before sampling the pest population again and be reasonably sure that density will not have grown so that it exceeds an intervention threshold. The first method makes use of tripartite sequential classification (TSC). Binns (Chapter 8) describes this sampling method in detail. We briefly review the technique to maintain continuity in the presentation. The second method uses both sequential classification and fixed sample size estimation. Both procedures cascade individual sampling plans through time to monitor a population trajectory.

A. CASCADED TRIPARTITE SEQUENTIAL CLASSIFICATION

Tripartite sequential classification sampling plans can be used to classify population density into one of three categories that are defined by two critical densities; cd_1 and cd_2, where $cd_1 < cd_2$. The three categories defined by the two critical densities are $u \leq cd_1$, $cd_1 < u \leq cd_2$, and $u > cd_2$ where u is the population mean. Based on the two critical densities, two dichotomous sequential classification (DSC) sampling plans are constructed. The dichotomous plans could be based on one of several different sequential classification schemes. We will use Wald's[8] sequential probability ratio test (SPRT). Using this test, the two dichotomous classification plans are constructed with the following null and alternate hypotheses:

DSC_1	DSC_2
$H_{10}: u = m_{10}$	$H_{20}: u = m_{20}$
$H_{11}: u = m_{11}, m_{10} < m_{11}$	$H_{21}: u = m_{21}, m_{20} < m_{21}$

Normally $cd_1 = (m_{10} + m_{11})/2$ and $cd_2 = (m_{20} + m_{21})/2$. It is also required that $m_{20} > m_{11}$. The mean density is classified as less than cd_i if H_{i0} is accepted and greater than cd_i if H_{i0} is rejected.

When two DSC sampling plans are used simultaneously to form a tripartite sequential classification plan, composite stop lines are formed as shown in Figure 3. When using a TSC plan, sampling can be terminated with one of three possible decisions: (1) if samples fall in region 1 the decision is to accept H_{10}, which leads to the conclusion that $u \leq cd_1$; (2) if the samples fall in region 2, the decision is to accept H_{20} and concurrently reject H_{10}, leading to the conclusion that $cd_1 < u \leq cd_2$; and (3) if the samples fall into region 3, the decision is to reject H_{20}, leading to the conclusion that $u > cd_2$.

With a DSC plan there are two performance criteria: OC and ASN. The OC is defined as the probability of accepting the null hypothesis given any true mean. With a TSC plan there are three possible decisions and three corresponding probabilities of making these decisions given any true mean ($pdeci$, $i = 1, 2, 3$). Given any two $pdeci$, the third is one minus the sum of these two. As with a DSC plan, an ASN also exists for a TSC plan; however, it is not monotonically convex.

Tripartite sequential classification sampling schemes can be used to monitor a population through time by allowing the time interval between samples to be indexed by the less-than-threshold classification. For example, if a population was classified as

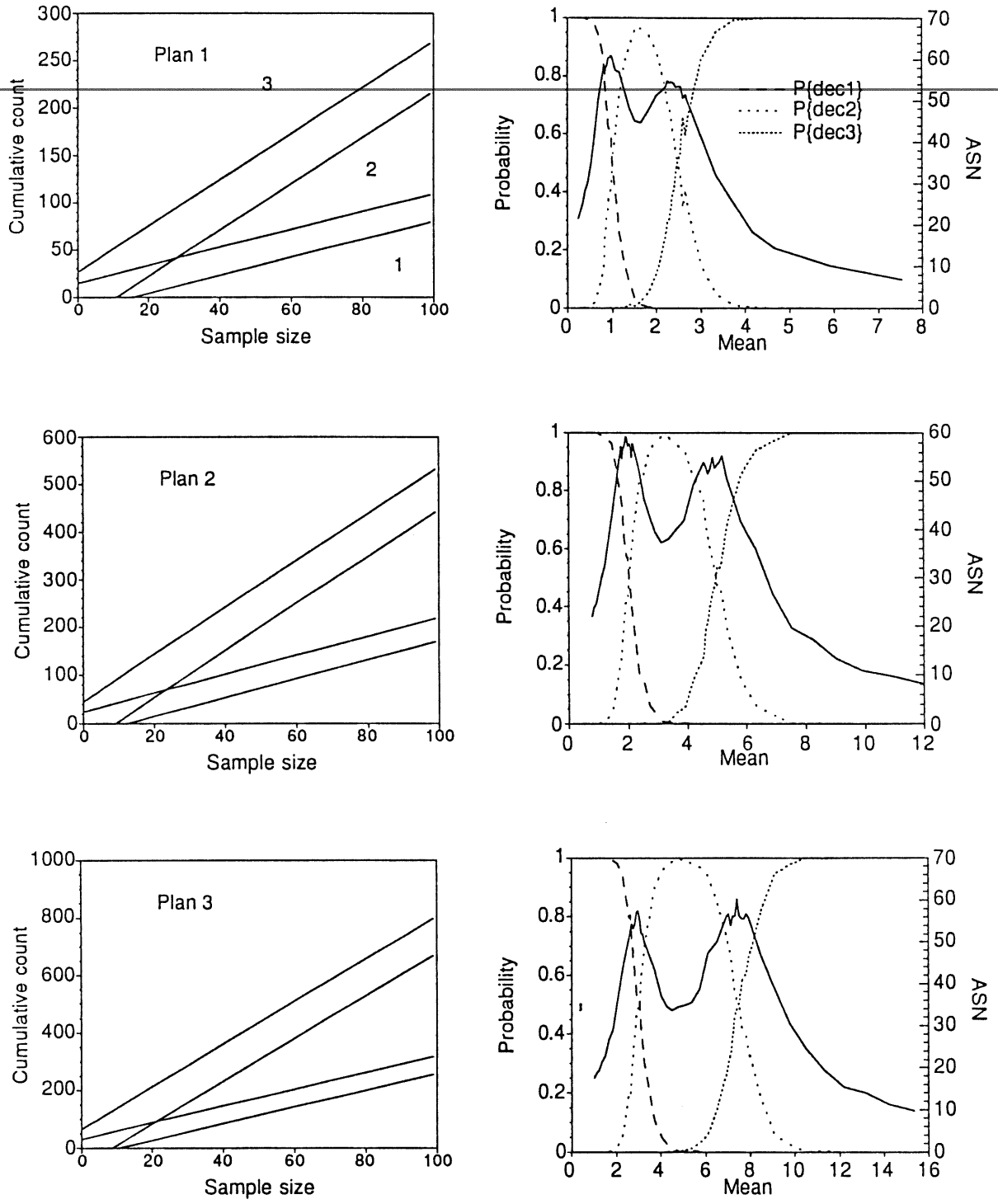

FIGURE 3. Stop limits and performance of three tripartite sequential classification sampling plans.

$\leq cd_1$ in Figure 3, the population might not be resampled for some specified time interval. If, however, it was classified as $> cd_1$ and $\leq cd_2$, it might be resampled again in a much shorter time interval. One way of determining the critical densities is to let cd_2 be the intervention threshold and to let cd_1 be a density, which, if allowed to grow unchecked for a period of time, would result in a density no greater than cd_2. The time period is the longest possible time to the next sample. If a density was classified as $> cd_1$ and $\leq cd_2$, the next sample might be taken one half of the specified time interval in the future. It is useful to call the shortest time interval between samples the sample interval (*sint*). Thus, if the mean is classified as $\leq cd_1$,

the time period to the next sample will be some multiple of *sint*, and if the mean is classified as $> cd_1$ and $\leq cd_2$, the time period to the next sample is *sint*. Different TSC plans can be used at different times when sampling the same population process. When one or more TSC plans are used to monitor a population in the manner described above, we will refer to the process as cascaded tripartite sequential classification sampling. The FORTRAN program "sprt" (see Appendix) can be used to construct and analyze TSC plans based on Wald's SPRT.

The performance of tripartite classification sampling plans, cascaded through time, can be evaluated using four criteria that describe expected performance when sampling the entire population process. In addition, the outcome of each sampling bout (i.e., each time the population density is sampled) also can be examined. The four overall performance criteria are an operating characteristic, a total average sample size, the expected number of sampling bouts, and expected loss. These criteria have their basis in DSC sampling plans; however, they differ in that they refer to a trajectory of population density through time and not to specific density values. The OC is defined as the probability of not intervening over all sample bouts. The total average sample size (TASN) is the expected number of samples required to sample the population process over the time of interest. The expected number of sampling bouts (ESB) is the average number of times the population process must be sampled. Finally, the expected loss (EL) is the expected cumulative density allowed without intervention. With DSC sampling, densities allowed without intervention have a probability of occurring defined by the OC function. When tripartite plans are cascaded through time, a decision to not intervene at one point in time may still allow for intervention at the next sampling time, albeit with a different density. An expected loss function for cascaded plans must account for this. With this introduction, we will now define these performance criteria mathematically.

Because population density is being sampled through time, some variables are time dependent and for these variables, time are denoted by a subscript. To keep the notation simple, it is necessary to consider two time scales. The first and obvious one is chronological time. The second time scale that will be convenient to use is the one for sample bouts (i.e., $1, 2, 3 \ldots$). For example, it will be clearer to write the density at sampling bout two as d_2 instead of d_{23} where 23 specifies the chronological time when sampling bout two occurred.

The population process to be sampled is defined by d_t where d is density and t is time. During the time interval of interest there are a known set of times (st) during which sampling might take place defined by a starting point (i.e., st = 1), a fixed time interval between samples (*sint*), and an ending point (st = end). We use the variable *sb* to denote the sampling bout where *sb* belongs to the set $\{1, 2, 3, \ldots, \text{last}\}$. For all combinations of density and time there exist functions that describe the outcome of sampling. These functions are the probability of decision functions (pdec$i$) and (ASN) function previously described for tripartite plans. As a review, *pdec*1 is the probability of no intervention and waiting two or more time intervals (*sint*) before sampling again, *pdec*2 is the probability of no intervention and waiting one sample time interval before sampling again, and *pdec*3 is the probability of intervening. The loss at any sampling time is defined as the cumulative density to that time point:

$$l_t = \int_0^t d_x \tag{2}$$

At the first sampling time, the density is d_1, where the temporal subscript now refers to the sample bout, the probability of sampling at this time is one ($ps_1 = 1$), the

probability of each of the three decisions is given by $pdeci_1$ where the subscript denotes the sampling bout, and the loss is given by l_1. At the second sampling time the density is d_2, the probability of sampling at this time is $ps_2 = pdec2_1(ps_1)$, and the probability of each of the three decisions is given by $ps_2(pdeci_2)$. These relationships apply because sampling at time two can only occur if a decision was made to resample after one time interval (*sint*) during the first sample bout and because the probability of reaching a particular decision during the second sampling bout must be conditioned on the probability of sampling at that time. At the third sampling time the density is d_3, the probability of sampling at this time is $ps_3 = pdec2_2(ps_2) + pdec1_1(ps_1)$, and the probability of each of the three decisions is given by $ps_3(pdeci_3)$. Sampling at time three can only occur if a decision was made to resample after one time interval during the second sample bout or if a decision was made to resample after two time-intervals during the first sample bout. Hence, the probability of sampling at the third sampling time is the sum of these two probabilities. The probability of sampling at periods 3 through last can be generalized as:

$$p_{sb} = pdec2_{sb-1}(ps_{sb-1}) + pdec1_{sb-2}(ps_{sb-2}) \tag{3}$$

Note that this equation will apply to all sampling periods provided $ps_0 = 1$, $pdec2_0 = 1$, $ps_{-1} = 0$, and $pdec1_{-1} = 0$.

The OC, total ASN, and ESB are now calculated as:

$$OC = 1 - \sum_{1}^{last} ps_{sb}(pdec3_{sb}) \tag{4}$$

$$TASN = \sum_{1}^{last} ps_{sb}(asn_{sb}) \tag{5}$$

$$ESB = \sum_{1}^{last} ps_{sb} \tag{6}$$

The OC and TASN are weighted sums of the respective values at each sample bout with the weights determined by the probability of sampling. The ESB is simply the sum of the probability of sampling because the value being weighted is one.

The last performance measure to be computed is EL. Recall that this is the expected cumulative density allowed to occur without intervention. At any sampling bout a measure of loss would be the cumulative density to that point times the product of the probability of sampling and the probability of intervening. Summing these loss measures over all sampling bouts provides a measure of loss when sampling the entire population process. This sum is easily computed provided two endpoints are considered. First, if the last sample period is reached, then the loss at that time is simply the probability of sampling times the loss. Second, at the next to last sampling period a decision to wait two time periods to sample again is equivalent to not intervening because sampling will not be repeated. Thus, the loss that would occur at the last sampling time should apply to these cases. The following equation accounts for these endpoints:

$$EL = \sum_{1}^{last-1} [l_{sb}\, ps_{sb}(pdec3_{sb})] + (ps_{end} l_{end}) + (ps_{end-1} l_{end}\, pdec1_{end-1}) \tag{7}$$

The measure of loss we have proposed is an expected value and therefore masks extreme values and provides no information on the distribution of possible values.

Loss also should be considered in terms of extreme values and in fact, managers may be more interested in the likelihood of a particular large cumulative density occurring when using a particular sampling plan than an expected value. Such values can be computed using the cumulative probability of intervening ($spdec3_t$) and the time dependent loss.

The cumulative probability of intervening is calculated as:

$$spdec3_{sb} = \sum_{1}^{sb} ps_j pdec3_j \tag{8}$$

A loss can be calculated for each sb entry and, therefore, a loss value for any specific *spdec*3 can be determined using interpolation.

The FORTRAN program "cascade" (see Appendix) can be used to evaluate the performance of TSC plans cascaded through time.

We illustrate the use and investigate some of the properties of cascaded TSC plans by studying plans developed for sampling *P. ulmi* in apples. Three TSC plans were constructed for use at different times during the period June 1 to August 31. Two sequential probability ratio tests based on the negative binomial distribution formed the basis for each TSC plan.

The following thresholds were used to define the DSC plans built around cd_2: June 1 to June 30—2.5 motile mites per leaf, July 1 to July 31—5.0 motiles per leaf, and August 1 and after—7.5 motiles per leaf. The cd_1 used to develop the second set of DSC plans were calculated as the densities that would result in no more than the intervention threshold density after 14 d, assuming the population grows exponentially with a growth rate of 0.065 ($N_t = N_0 e^{0.065t}$ where $t = 14$). This growth rate was an average determined by fitting an exponential model to 14 data sets that described *P. ulmi* dynamics. Thus, we set the minimum time interval to the next sample (*sint*) to 7 d and the resample interval if the mean was classified as $\leq cd_1$ to 14 d. Parameters used to construct the sampling plans are shown in Table 1.

Stop lines and probability of decision (*pdeci*) and ASN functions for these sampling plans are shown in Figure 3. Maximum ASN values occur when the density equals cd_1 or cd_2. These are also the points where pdec*i* functions intersect. The jaggedness of the functions in Figure 3 occurs because the functions were estimated using simulation ($n = 500$).

TABLE 1
Parameters Used to Construct Tripartite Sequential Classification Sampling Plans for Use with *P. ulmi*

Plan[a]	cd_i[b]	k[c]	H_0	H_1	Alpha	Beta
1.1	1.0	0.301	0.7	1.3	0.1	0.1
1.2	2.5	0.467	2.0	3.0	0.15	0.15
2.1	2.0	0.418	1.6	2.4	0.15	0.15
2.2	5.0	0.667	4.2	5.8	0.15	0.15
3.1	3.0	0.513	2.3	3.7	0.1	0.1
3.2	7.5	0.827	6.5	8.5	0.15	0.15

[a] For each plan there are two thresholds and two sets of SPRT parameters. The number following the decimal point signifies whether it is plan 1 or 2 of a tripartite scheme.

[b] The first value for each plan is cd_1 and the second is cd_2.

[c] Parameter k for negative binomial distribution computed using moments and based on a predicted variance calculated using the model $s^2 = am^b$.

Performance of the sampling plans cascaded through time was studied by applying the plans to a set of populations described by exponential growth (r = 0.03 to 0.075) and to seven actual population trajectories. The first sampling plan was applied during the time period 1 to 30, sampling plan 2 was applied during the period 31 to 60, and sampling plan 3 was used during the period 61 to end. Performance of cascaded dichotomous sampling plans was also studied. This was done to determine the saving in sampling costs that would result and what errors might result from using the tripartite plans. These dichotomous plans were constructed using the parameters for the tripartite plans based on cd_2 shown in Table 1. These plans were cascaded in time by applying the sampling plans every 7 d.

OC values obtained with the cascaded tripartite plans are shown adjacent to each population trajectory in Figure 4. Also shown in this figure are the time-dependent intervention thresholds. The OC is very steep when densities exceed the intervention threshold (cumulative density of 150 to 180). In fact, the OC is steeper than might be predicted based on the *pdec*3 values for each tripartite sampling plan. This is because the OC for cascaded plans is a summation of probabilities over all sample bouts; the sum of *pdec*3 values much less than one over a set of sampling bouts can still lead to large (i.e., close to 1.0) overall probability of intervention. A numerical example using a cascaded dichotomous plan will clarify this. Suppose sampling occurred three times where the densities were 2.5, 6, and 8 and the probabilities of intervening at these times (1 − OC) were 0.4, 0.5, and 0.6. The probability of sampling at the first time is 1.0, so the probability of sampling at time 2 is $1(1 - 0.4) = 0.6$. The cumulative probability of intervening at the second sampling time is $0.4 + (0.5 * 0.6) = 0.7$. The probability of sampling at the third time period is $0.6(1 - 0.5) = 0.3$ and the cumulative probability of intervening is $0.7 + (0.3 * 0.6) = 0.88$.

The four overall performance criteria for the four sampling plans are shown in Figure 5. In addition to expected loss, losses that would occur with probability 0.2 and 0.05 are also shown. The OC for tripartite and dichotomous plans are essentially equal. Slightly more total samples were required by the dichotomous plans. More importantly, the dichotomous plans required approximately twice the number of bouts

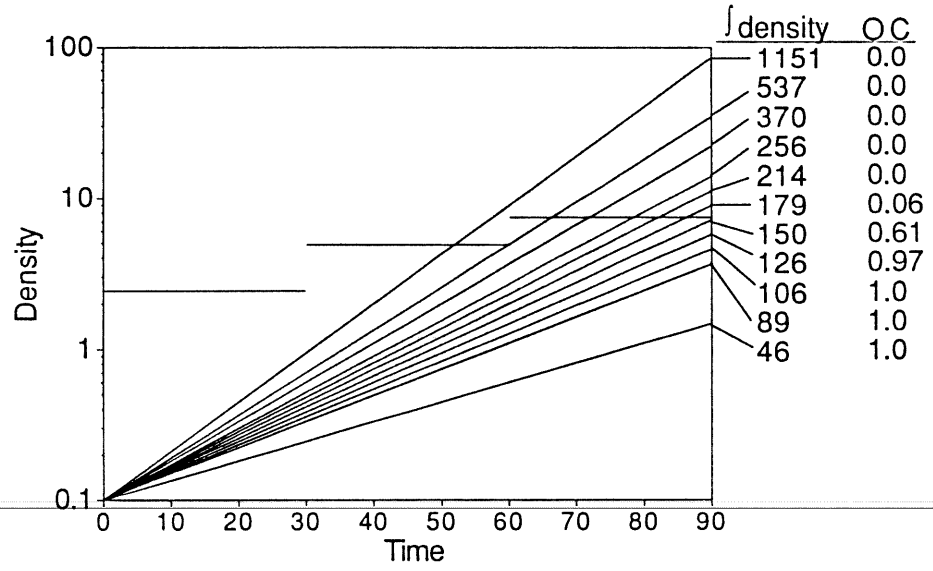

FIGURE 4. Populations with exponential growth, cumulative density, and OC values for cascaded tripartite sequential classification sampling plans applied to the populations. Short horizontal lines are intervention thresholds.

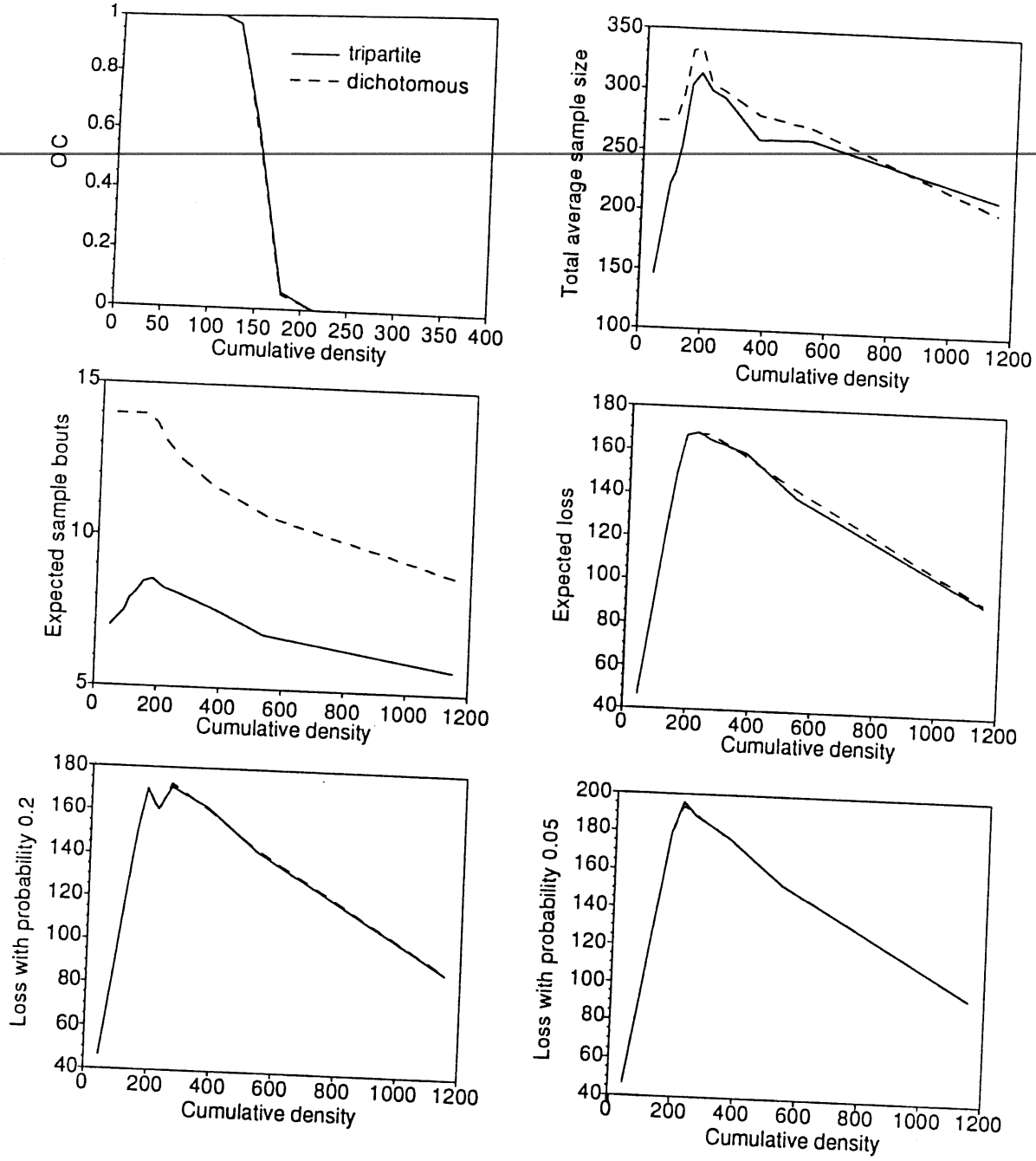

FIGURE 5. Performance of cascaded tripartite and dichotomous sampling plans.

as the tripartite plans. Because sample bouts are usually more costly than samples per bout, tripartite plans should significantly reduce overall sampling costs. Losses for the tripartite and dichotomous plans were nearly identical. Intervention thresholds for *P. ulmi* are designed to prevent a cumulative density of 500. This was always achieved with at least probability 0.95. As a result, the tripartite plans may, in fact, be overly conservative.

The seven population trajectories sampled are shown in Figure 6. Sampling was started at time 12 and the last sample was taken at time 89. Results of applying the tripartite and dichotomous sampling plans to these populations are given in Table 2.

Performance of the tripartite and dichotomous plans was similar except for the total number of samples and the number of bouts which were both larger for the dichotomous plans. Use of the tripartite plans reduced the number of bouts by one half to one third. With populations 1, 2, 4, and 6 intervention always occurred. These

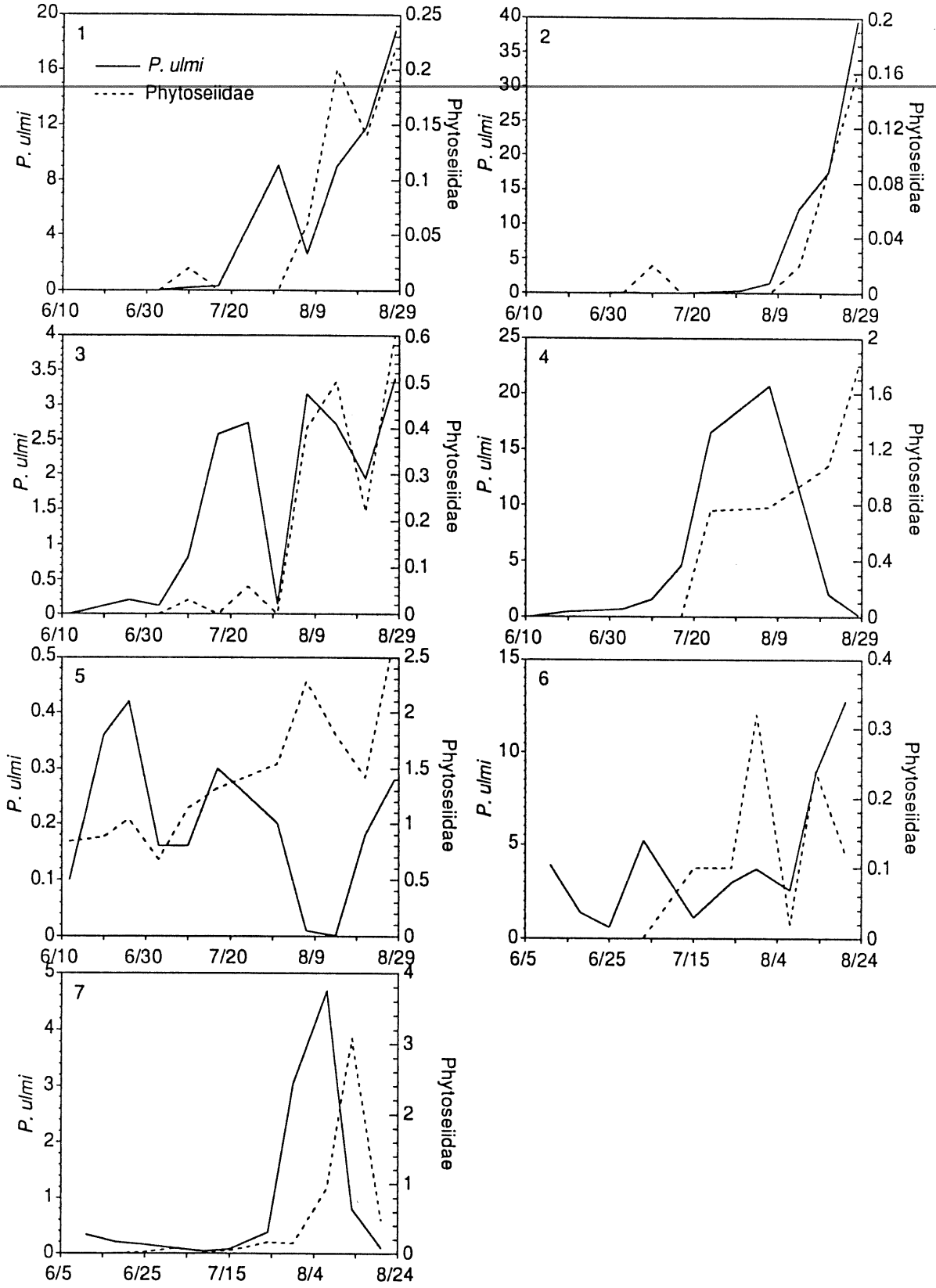

FIGURE 6. Dynamics of seven *P. ulmi*-phytoseiid mite populations.

populations all had cumulative densities at or near the level for which damage occurs (500). With population 4 biological control occurred; however, not before a high density of *P. ulmi* was present. For populations 3, 5, and 7, biological control was successful and there was no probability of intervention.

TABLE 2
Results of Cascading Tripartite and Dichotomous Complete Count and Tripartite Binomial Count Sequential Classification Sampling Plans to Sample Seven Population Trajectories

Population	∫ density	OC	ASN	Bouts	Loss	Loss with $P\{0.2\}$	Loss with $P\{0.05\}$
			Tripartite				
1	329.7	0	145.8	4.8	63.4	60.0	68.0
2	363.1	0	118.8	6.0	163.1	142.1	157.9
3	103.9	1	229.9	6.6	103.9	103.9	103.9
4	462.2	0	119.9	4.1	109.5	96.5	108.0
5	15.5	1	129.5	6.0	15.5	15.5	15.5
6	363.4	0	25.2	1.0	0	0	0
7	69.5	1	180.2	6.5	69.5	69.5	69.5
			Dichotomous				
1	329.7	0	199.1	7.7	60.0	57.63	67.1
2	363.1	0	190.6	10.0	58.4	48.90	56.1
3	103.9	1	236.7	12.0	103.7	103.90	103.9
4	462.2	0	161.0	6.7	92.1	91.60	106.7
5	15.5	1	235.3	12.0	15.5	15.50	15.5
6	363.4	0	24.8	1.1	0	0	0
7	69.5	1	239.4	12.0	69.5	69.50	69.5
			Binomial				
1	329.7	0.08	270.3	5.4	143.0	211.9	329.7
2	363.1	0.10	223.6	6.0	188.7	155.4	363.1
3	103.9	0.98	327.4	6.4	103.1	103.9	103.9
4	462.2	0.01	219.5	4.3	132.9	110.8	233.2
5	15.5	1.00	176.2	6.0	15.4	5.4	15.4
6	363.4	0.03	163.5	2.6	65.5	159.5	319.8
7	69.5	0.96	245.2	6.3	68.4	69.5	69.5

Note: The maximum potential number of sampling bouts is 12.

B. BINOMIAL COUNT TRIPARTITE PLANS

Counting small, numerous organisms such as mites in the field is tedious and often not practical. To circumvent this problem, sampling plans have been developed that substitute presence or absence of an organism on a sample unit for complete enumeration. The shortcoming of this approach is that sampling plans based on binomial counts are less precise than plans that use enumeration. This shortcoming can be ameliorated by defining a positive binomial score as a sample unit with more than T_p organisms where T_p is called a tally point (Jones, Chapter 9). However, when T_p becomes too large the benefits of binomial counts are reduced, and our experience has been that practitioners favor binomial plans with a tally count of zero. We developed binomial count tripartite plans for *P. ulmi* based on the negative binomial distribution using the FORTRAN programs in the Appendix and discovered that when they are cascaded through time, they performed much better than expected. The reason for this is identical to the explanation given for the large cumulative

probability of intervention when individual pdec3 values are modest. Here, we briefly summarize these findings because room does not permit a detailed exposition.

Parameters used to develop tally zero binomial sampling procedures are shown in Table 3. We also constructed plans using a tally count of four and found the performance, when they were used once in time, of the tally 4 sampling plans to be superior to the tally 0 sampling plans. This is not unexpected and parallels the conclusions of Nyrop and Binns.[3]

Performance of the tally 0 and tally 4 sampling plans cascaded through time was studied by applying the plans to the set of exponential populations. The OC for the tally 0 plan was not as steep as that for the tally 4 plan, and losses for the tally 0 plan were greater than for the tally 4 plans. Intervention thresholds for *P. ulmi* are designed to prevent a cumulative density of 500, which was achieved by the tally 0 plans with probability 0.95. This is an important result because it previously was suggested[3] that tally zero plans be avoided because of their poor precision. When tally 0 plans are cascaded they are quite adequate and tally 0 counts are easier to make than tally 4 counts.

Performance of the tally 0 binomial and complete enumeration sampling plans with the exponential populations is compared in Figure 7. By all accounts, the tally 0 binomial plan performed adequately and in one sense was superior to the complete enumeration plans because it is not as conservative as the complete enumeration plans. Of course, the complete count plans could be modified by using higher thresholds.

Performance of the tally 0 and complete count plans did not differ greatly, except that the ASN was greater for the binomial plans as were for the losses incurred with probability 0.2 and 0.05 (Table 2). These losses were higher for the tally 0 plans; however, they were still acceptable. With populations 1, 2, 4, and 6 intervention is practically assured. These populations all had cumulative densities at or near the level for which damage occurs (500). For populations 3, 5, and 7 biological control was successful, and there was very low or no probability of intervention. The ASN for the binomial plans probably could be reduced without sacrificing performance because much of the variability that influences the OC function is due to variation in the binomial count-mean density model which is not affected by sample size.

TABLE 3
Parameters Used to Construct Binomial Tripartite Sequential Classification Sampling Plans for Use with *P. ulmi*

Plan[a]	Tally	cd_i[b]	k[c]	Threshold[d]	H_0	H_1	Alpha	Beta
1.1	0	1.0	0.301	0.356	0.306	0.406	0.075	0.075
1.2	0	2.5	0.467	0.578	0.529	0.628	0.075	0.075
2.1	0	2.0	0.418	0.52	0.480	0.560	0.1	0.1
2.2	0	5.0	0.667	0.760	0.720	0.800	0.075	0.075
3.1	0	3.0	0.513	0.627	0.587	0.667	0.1	0.1
3.2	0	7.5	0.827	0.852	0.812	0.892	0.05	0.05

[a] For each plan there are two thresholds and two sets of SPRT parameters. The number following the decimal point signifies whether it is plan 1 or 2 of TSC.
[b] The first value for each plan is cd_1 and the second is cd_2.
[c] Parameter k for negative binomial distribution computed using moments and based on a predicted variance calculated using the model $s^2 = au^b$.
[d] cd_i expressed as the proportion of sample units with > Tally count.

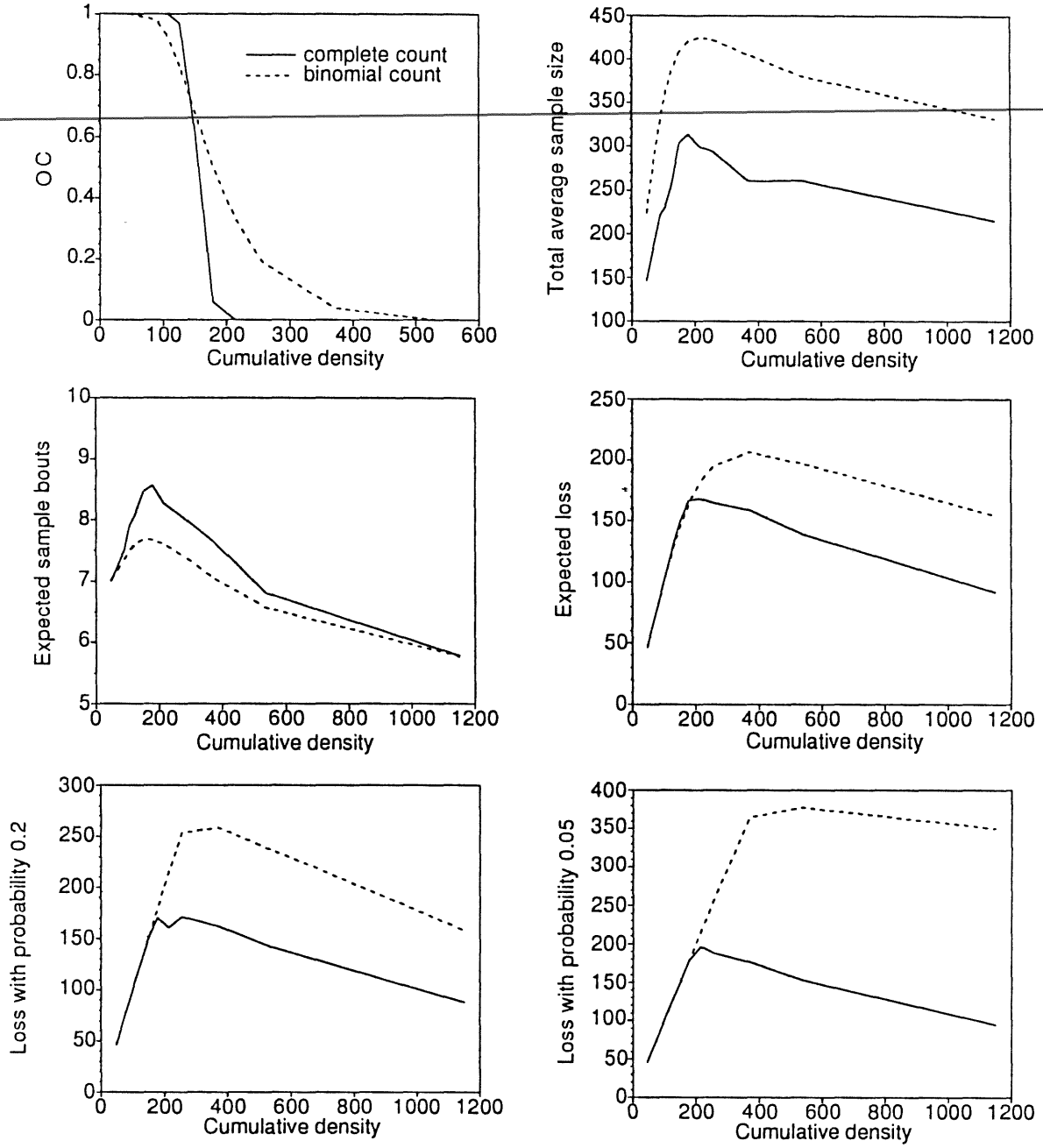

FIGURE 7. Performance of cascaded complete count and binomial tripartite sequential classification sampling plans.

C. ADAPTIVE FREQUENCY CLASSIFICATION MONITORING

If density is classified as less than an intervention threshold, tripartite classification sampling plans determine when the population should be sampled again by classifying density into one of two categories. If, for example, a decision was made to resample after the maximum waiting time, it does not matter whether the actual density was just slightly less than the critical density used to denote the maximum waiting time (cd_1) or much less than this value. However, the time sampling can be delayed depending on the actual density, might be longer if the true mean is much less than cd_1. The waiting time to the next sample bout might be more precisely determined by allowing density to be classified into four or more categories. However, such plans would require three or more dichotomous classification procedures and would likely be very cumbersome to develop. A more rational approach is to use both classification

and estimation to formulate sampling plans for monitoring a population through time. We have developed such a sampling protocol and call the scheme adaptive frequency classification monitoring (AFCM). As with TSC plans, AFCM plans are cascaded through time to monitor a population trajectory. Performance criteria for cascaded AFCM plans are identical to those for cascaded TSC plans. Here, we describe AFCM sampling and illustrate its use by developing plans for monitoring *P. ulmi*. FORTRAN programs "afcm" and "cascade" perform necessary computations (see Appendix).

In AFCM, a sequential classification procedure is used to determine whether a density exceeds an intervention threshold. Unlike TSC plans though, densities are not classified as less than the intervention threshold. Instead, an estimated density is used to determine the waiting time to the next sample bout. Wald's SPRT is used to construct a classification stop line for determining whether density exceeds the intervention threshold. The mean for the alternate hypothesis is set equal to the intervention threshold. The mean for the null hypothesis is set equal to a density which, if allowed to grow at the maximum growth rate for the minimum time until the next sample bout, would not exceed the intervention threshold. Note that the densities used to construct these hypotheses are different from those used to specify stop lines for the TSC plans. With TSC plans null and alternate hypotheses are constructed around the intervention threshold while with AFCM plans, the intervention threshold is the density used for the alternate hypothesis. With TSC plans null and alternate hypotheses are also constructed around a density which, if allowed to grow at the maximum growth rate for the maximum time until the next sample is taken, would equal the intervention threshold. With AFCM plans the density for the null hypothesis is defined using the minimum time to the next sample bout.

In AFCM schemes, if a decision is to not intervene, the time to the next sample bout is based on an upper confidence limit for an estimated density obtained via a fixed sample size. We reasoned that individuals using an AFCM sampling plan would not be motivated to spend much sampling effort if the density was below an intervention threshold. Thus, we established a maximum sample size, which, if reached before a decision was made to intervene, indicated that the density was below the intervention threshold and a waiting time to the next sample should be determined. While a waiting time could be determined for each possible estimated density, this is not practical. Instead, we specify waiting times as multiples of the resample interval (*sint*; the minimum time between sample bouts) and then determine ranges of estimated densities that correspond to each waiting time. How these ranges of densities are determined is explained below.

AFCM stop lines are a composite of the upper stop limit for the sequential classification procedure and the maximum sample size. Shown in Figure 8 is an illustrative stop line. The first step in constructing an AFCM stop line is to plot the upper stop line for the sequential classification plan. Let $thr(t)$ be the intervention threshold at the current sampling time (t). As for the TSC plans, we assume that population growth can be described by a simple exponential model, although some other model could also be used. The null and alternate hypotheses for the sequential classification procedure are then:

$$H_0: \quad u = thr(t)/[\exp(r \cdot sint)]$$

$$H_1: \quad u = thr(t).$$

Alpha and beta for the SPRT are chosen to produce a desired width between the stop

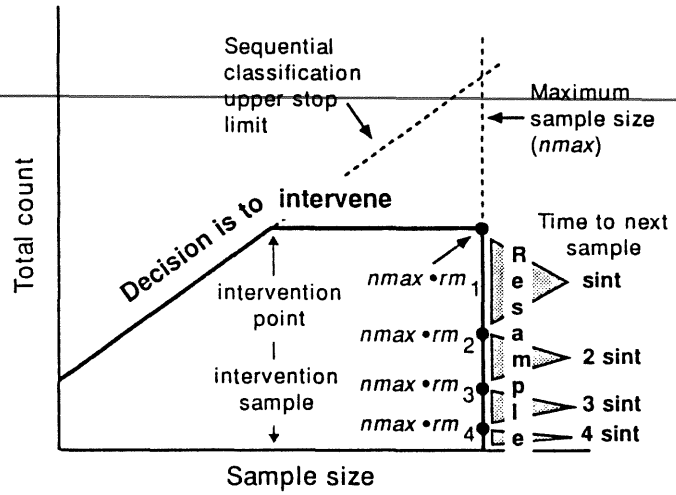

FIGURE 8. Adaptive frequency classification monitoring (AFCM) stop limit.

lines. Smaller alpha and beta result in wider stop lines that increase the average sample size and improve precision of the SPRT. With AFCM plans, this affects the proportion of sampling bouts that terminate with a decision to intervene before the maximum sample size is reached.

The second step is to plot a vertical line corresponding to the maximum sample size. Points on this line define the beginning and end of line segments which, if intersected by the total number of animals found after examining the maximum number of samples, determine the waiting time until the population is sampled again. Let ucl be an upper confidence limit for a mean estimated using the maximum sample size ($n_{\max}$). Mean densities (rm) with upper confidence intervals ucl that solve

$$\text{ucl} = \frac{thr(t + i * sint)}{e^{(r * i * sint)}} \tag{9}$$

are the points on the maximum sample size line that define the line segments. In Equation 9, i is an integer that ranges from one to the maximum number of *sint* time intervals in the future sampling might be delayed, thr($t + i * sint$) is the intervention threshold at time $t + \text{i} * sint$ and an exponential model has been used to define population growth. In Figure 8, i ranges from 1 to 4 and the mean densities with upper confidence intervals defined by Equation 9 are labeled as $rm_1 * n_{\max}$ (rm = resample mean). If the total number of animals (total) in $n_{\max}$ samples satisfies

$$rm_i \cdot n_{\max} > \text{total} \geq rm_{i+1} \cdot n_{\max} \tag{10}$$

then sampling is to be done $i \cdot sint$ time steps in the future. If total $> rm_1 \cdot n_{\max}$ a decision to intervene is made.

Equation 9 cannot be used directly to find the points (means multiplied by the maximum sample size) on the maximum sample size line that defines the line segments which correspond to specific waiting times. This is because Equation 9 is solved for an upper confidence limit; however, knowing the upper confidence limit for an estimated mean does not always allow determination of the mean itself. This problem is easily overcome on a computer by first determining ucl via Equation 9 and then performing a line search to find the mean density with this upper confidence interval. This is the strategy used in program "afcm."

It is important to note that the point on the maximum sample size line that defines the start of the line segment used to specify the shortest waiting time to the next sample bout ($n_{max} \cdot rm_1$) will usually lie below the upper sequential stop limit. This occurs because the SPRT uses a likelihood ratio test to determine whether sample data are most consistent with H_0 or with H_1, whereas the points on the maximum sample size line are defined using an upper confidence limit for the estimated density. If the point $n_{max} \cdot rm_1$ lies below the upper stop limit, a horizontal line can be drawn from this point to the upper stop limit and this line now defines part of the "intervene decision" stop limit (Figure 8). This can be done because once the total count exceeds $n_{max} \cdot rm_1$, a decision to intervene will always be made regardless of additional samples taken. If $n_{max} \cdot rm_1$ lies above the upper stop limit for the classification procedure, then the intersection of the stop limit and the maximum sample size line defines the start of the first line segment, which, if intersected by the total count, specifies a decision to sample again after *sint* time steps. The count corresponding to the intersection of the SPRT upper stop limit and $rm_1 \cdot n_{max}$ is called the intersection point and the sample size where this occurs is called the intersection sample.

A final point that must be made about AFCM stop limits, as we have formulated them, concerns the intervention threshold. In the AFCM protocol we have allowed the intervention threshold to be any nondecreasing continuous function of time and computation of waiting times to the next sample bout takes into account future threshold values. In contrast, the threshold for the TSC procedure was a discrete step function and computation of the waiting time to the next sample bout was based on the current threshold. By allowing the intervention threshold to be a nondecreasing continuous function of time and, by formulating waiting times based on future thresholds, waiting times to the next sample bout are lengthened. There is a price for this, though, because a different sampling plan must now be constructed for every possible sampling bout with a unique threshold. For example, if *sint* = 7 d and a population is to be monitored for 92 d with a different threshold for each day, 14 sets of stop lines must be determined. The large number of sampling plans required to monitor a population over several weeks or months is a possible drawback to AFCM-based monitoring plans.

AFCM plans are cascaded like TSC plans to monitor a population through time. Each AFCM plan has a set of performance characteristics consisting of an ASN function, pdec*i* functions for each of the resample decisions $(1, \ldots, i$ where i is the maximum *sint* interval sampling that may be delayed), and probability of a decision to intervene. Program "afcm" computes these performance characteristics for individual sampling plans. To determine how cascaded AFCM plans perform, the computations used with the cascaded TSC plans need only be slightly modified. All that is required is that, instead of sampling and decision probabilities being based on the outcome from two sample bouts in the past, i sample bouts in the past must now be considered. Program "cascade" does this.

To monitor *P. ulmi* we constructed AFCM plans based on a negative binomial distribution with the variance modeled using TPL. Additional parameters were *sint* = 7 d, $i = 4$, r for the exponential growth model = 0.065, alpha and beta for the SPRT = 0.1, $n_{max} = 50$, and alpha for the upper confidence limit = 0.3. Stop line parameters for protocols 1, 5, 9, and 13 from a total of 14 are shown in Table 4. Probability of decision functions (pdec*i*) and ASN functions for these plans are shown in Figure 9. Also shown in this figure are the proportion of decisions reached by (1) crossing the SPRT upper stop limit, (2) crossing the intersection point, and (3) reaching the maximum sample size. Probabilities of intervening were greater than specified via the SPRT parameters alpha and beta. SPRT alpha is the probability of

TABLE 4
Stop Line Parameters for Four AFCM Sampling Plans Used to Monitor European Red Mite

				SPRT		Intersection			
Plan	Threshold	Time	H_0	Intercept	Slope	Point	Sample size	Waiting time	Decision count
1	2.5	1	1.59	27.44	1.98	87	30	7	87
								14	65
								21	47
								28	33
5	4.91	29	3.12	36.38	3.89	149	30	7	149
								14	99
								21	65
								28	44
9	6.13	57	3.89	39.88	4.85	184	30	7	184
								14	120
								21	78
								28	51
13	7.29	85	4.63	42.9	5.77	216	30	7	216
								14	135
								21	84
								28	52

accepting the null hypothesis when the alternate hypothesis is true and for a conventional SPRT the probability of intervening approximately equals 1-alpha when the mean is equal to the alternate hypothesis. For the AFCM plans the probability of intervening was always in excess of this value because use of the intersection point as a stop limit increased the number of decisions to intervene. Similarly, when the density equals the null hypothesis, the probability of intervening is approximately equal to beta for a conventional SPRT. However, almost all decisions made when the density was close to the null hypothesis value were made by either reaching the maximum sample size or by crossing the intersection point. Both situations result in more decisions to intervene being made than if only the SPRT stop lines had been used. Overall, these factors cause the procedure to result in a high proportion of intervene decisions when the density is considerably less than the intervention threshold.

We used the AFCM protocols to sample the set of exponentially growing populations and the historical mite counts to which the cascaded TSC procedures were applied. Shown in Figure 10 are the performance criteria for cascaded TSC and AFCM plans used to monitor the exponential populations. When cascaded, the AFCM protocols resulted in more decisions to intervene than with the TSC protocol. The AFCM plans required approximately two thirds as many total samples as the TSC plans and approximately one half as many sample bouts. At low densities, this was due to the ability of AFCM plans to lengthen the waiting time beyond the 14-d maximum of the TSC plans. At high densities, the AFCM procedure had lower total average samples sizes and number of sample bouts because this procedure resulted in an intervention decision more quickly than the TSC protocols.

Shown in Table 5 are the results from applying the TSC and AFCM plans to the seven historical populations. For population 1 AFCM and TSC plans provided similar results. For population 2 the AFCM protocol did not always result in intervention (OC = 0.14). This occurred because at the third from last sample bout, 14% of the

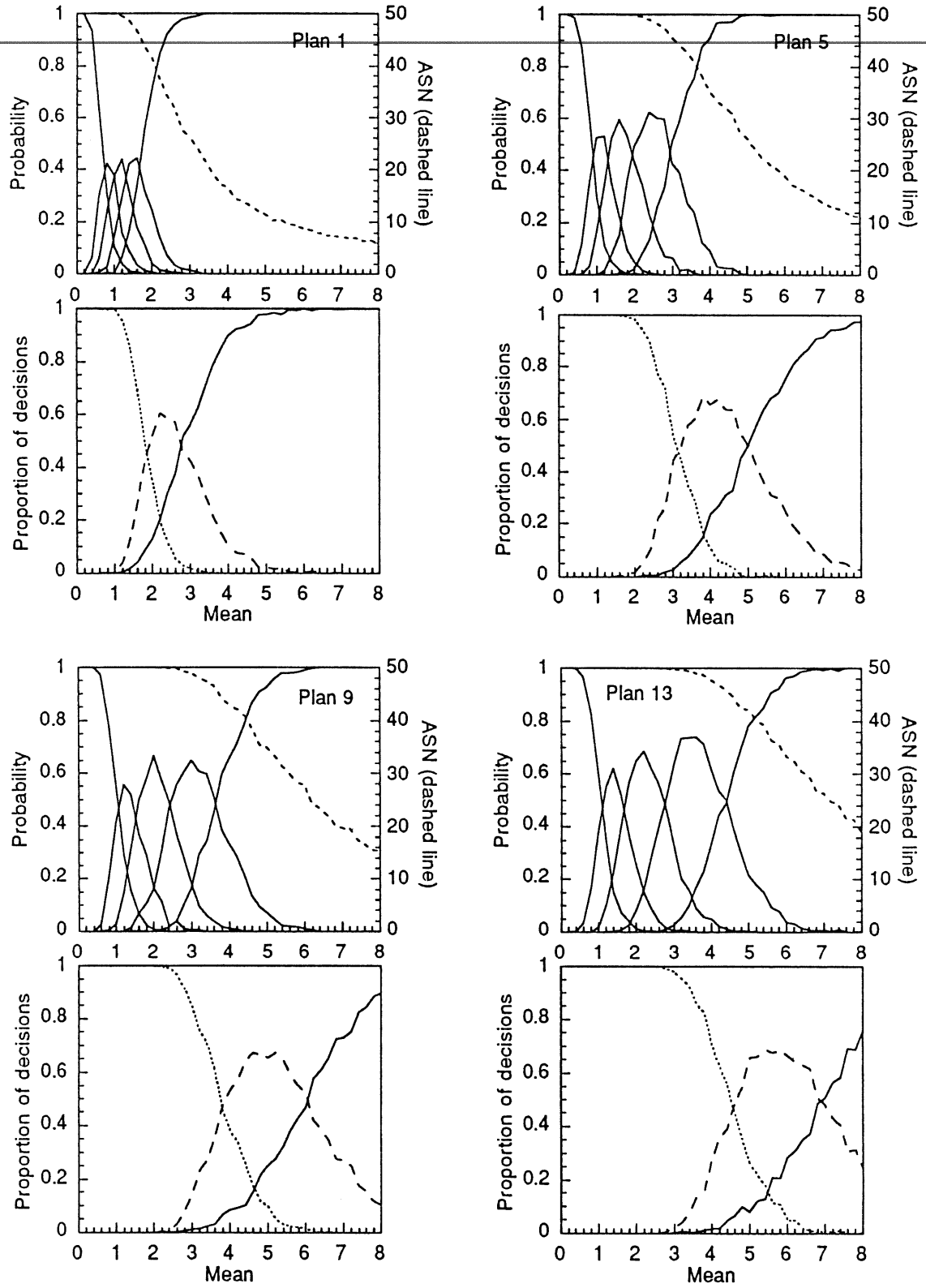

FIGURE 9. Probability of decision and average sample number functions for four AFCM sampling plans and the proportions of decisions made by (1) reaching the maximum sample size (small dashed line), (2) crossing the horizontal portion of the upper stop limit (large dashed line), and (3) crossing the SPRT upper stop limit (solid line). The probability of decision function that starts at 1.0 with mean = 0.0 is the decision to wait four time intervals until the next sample bout. This is followed by decisions to wait 3, 2, and 1 time interval to the next bout. The final function is for a treat decision.

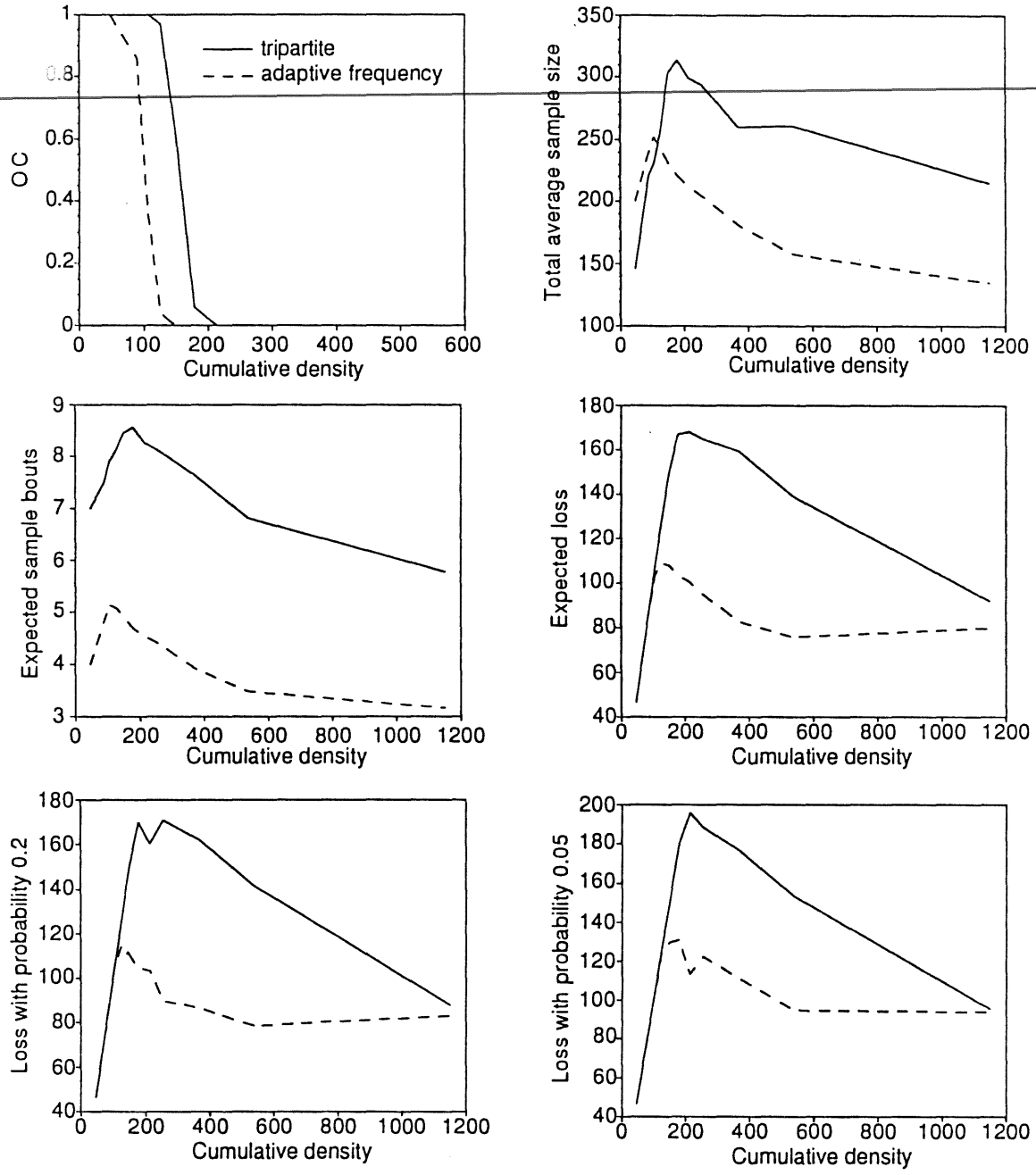

FIGURE 10. Performance of cascaded adaptive frequency classification monitoring plans and cascaded tripartite sequential classification sampling plans applied to exponentially growing populations.

decisions were to wait 4 weeks before sampling again, which was equivalent to a decision to not intervene because these populations were never sampled again. By extending the sampling period to 96 d from 92 d the OC was reduced to 0.0 and the ASN and expected number of sample bouts increased to 169.2 and 4.0, respectively. For population 3 the AFCM protocol resulted in a low rate of intervention (0.17). A higher rate of intervention occurred with population 7 where a density of 4.8 at time 75 resulted in a probability of intervention equal to 0.77. The expected number of sample bouts was always less with the AFCM protocol.

In comparison to TSC protocols, the AFCM plans resulted in a greater proportion of intervene decisions and the AFCM plans usually required fewer sampling resources (sample bouts plus number of samples per bout). Space does not permit development

TABLE 5
Results of Applying Cascaded Adaptive Frequency Classification Monitoring Plans and Tripartite Sequential Classification Sampling Plans to Seven Populations

	Adaptive frequency classification								
	Monitoring					Tripartite sequential classification			
1	329.7	0.0	167.19	4.0	184.4	0.0	145.8	4.8	63.4
2	363.1	0.14	166.24	3.9	307.7	0.0	118.8	6.0	163.1
3	103.9	0.83	216.14	4.3	97.7	1.0	229.9	6.6	103.9
4	462.2	0.0	116.5	3.0	129.3	0.0	119.9	4.1	109.5
5	15.5	1.0	150	3.0	15.5	1.0	129.5	6.0	15.5
6	363.4	0.0	22.88	1.0	0.4	0.0	25.2	1.0	1.8
7	69.5	0.38	196.1	4.1	52.9	1.0	180.2	6.5	69.5

and analysis of binomial count plans for the AFCM protocol; however, this is a straightforward endeavor.

IV. SUMMARY AND CONCLUSIONS

Monitoring for biological control is currently not well developed and additional research is needed. Pest-natural enemy ratios sometimes can be used to predict the likelihood for biological control. If such a ratio is appropriate, the ratio classification sampling method described in this chapter can be used. However, the usefulness of this method would be enhanced if binomial counts could be substituted for complete enumeration. Cascaded tripartite plans provide a method of monitoring a population without the need for determining the abundance of natural enemies. At least one extension to this approach should prove useful: one set of plans could be used if natural enemies are present in samples and another set based on a larger pest population growth rate could be used when natural enemies were not present. In this way the time interval between samples might be made longer. Adaptive frequency classification monitoring shows even greater promise as a monitoring tool. Further work on this procedure is needed to determine how to design AFCM sampling plans that are not overly conservative in recommending intervention without increasing required sampling resources.

APPENDIX

ratio.for: sequential classification of ratios

Background: Program ratio.for is designed to construct and analyze ratio sequential classification sampling plans. These plans are designed to simultaneously classify (1) a pest density as either greater or less than an intervention threshold and (2) a pest/natural enemy ratio as either greater or less than a critical value. Stop lines are based on confidence intervals about the intervention threshold and critical ratio. Samples are processed in batches where the batch size should be > 20. Means from each batch sample are assumed to be bivariate normal distributed with the variance modeled as a function of the mean via TPL and the correlation between means constant.

The program works as follows:

1. Model parameters and data required to compute ratio stop limits are input. Required data are TPL parameters for pest and natural enemy (ya, yb, xa, xb),

correlation coefficient used to describe statistical correlation between counts of prey and natural enemy (rho), the critical ratio (cr), the intervention density for the pest (t), standard normal z values for computing ratio and pest density stop lines (zcr, zt) and a delta value for computing the ratio stop limits ($xdel$). Larger z values result in more stringent stop lines. Delta should be ≤ 0.1. These values are displayed and confirmed. Parameters for specifying the range of prey densities (numerator of ratio) and range of sample sizes for which ratio stop limits are to be calculated are then input and confirmed.

2. Ratio and pest density stop limits are computed. Stop limits can be displayed on the screen and written to a file. Stop limits for the ratio are indexed by the sample size and numerator (pest) density.
3. Performance of the sampling plan is determined via simulation. Data to be entered are parameters for specifying the range of prey densities and ratios to be used in the simulation, the correlation coefficient to be used in the simulation, and the number of Monte Carlo runs. Output consists of performance parameters indexed by the prey density (Y) and prey/natural enemy ratio (R). Performance parameters are the probabilities of making classifications one through four (dec1 = $Y \leq T, R \leq CR$, dec2 = $Y > T$, $R \leq CR$, dec3 = $Y \leq T$, $R > CR$ the ASN, and dec4 = $Y > T$, $R > CR$), the average sample size, the the probability of an incorrect classification (PIC), the ratio-only OC and ASN (rOC, rASN), and the pest density-only OC and ASN (tOC, tASN). The PIC is based on Pdec1-Pdec4. The ratio-only and pest density-only OC and ASN are determined by making decisions using only the ratio or pest density stop lines. These parameters are only written to the tab delimited output file and are not displayed on the screen.

An example run is shown below. User inputs are underlined and comments are in italics.

```
> ratio
Enter name for output file (≤ 12 characters)
test.out
Input model parameters
TPL parameters (var  =  a*m**b)
Parameters for numerator (y):
a:
4.32
b:
1.42
Parameters for denominator (x):
a:
2.38
b:
1.2
correlation coefficient:
-0.25
critical ratio:
7.5
intervention threshold:
5.0
z for ratio:      z is a standard normal deviate
1.28
z for intervention threshold:
1.28
```

```
delta for determining x: values much smaller than 0.001 greatly increase computation time
0.01
        TPL parameters

   ya      yb      xa      xb      rho      cr      t      zcr      zt      xdel
 4.320   1.420   2.380   1.200   −0.25    7.50    5.00   1.28     1.28    0.010000

Parameters correct? (Y or N)
y
Input parameters for specifying numerator (y)
minimum:
1
maximum:
10
delta:
1
  ymin  ymax  ydel
   1     10    1
Parameters correct? (Y or N)
y
Input parameters for specifying sample sizes'
batch size
20
maximum sample size
100
  batch size  maximum
   20          100
Parameters correct? (Y or N)
y
Display stop limits? (Y or N)
n
Write stop limits to file? (Y or N)
y
stop limits determined for following range of y
minimum = 1.00 maximum = 10.00
input parameters for specifying y during simulations   it is not necessary to use the same range for y in
the simulation as was used to determine the stop limits; however, stop limits required in the simulation for y
values beyond those for which stop limits were actually determined will be linear interpolations
minimum:
1
maximum:
10
delta:
1
  ymin  ymax  ydel
   1     10    1
Parameters correct? (Y or N)
y
input parameters for specifying ratio   the same is true for the ratio as stated above for y
minimum:
3.5
maximum:
10.5
delta:
0.5
  ymin  ymax  ydel
   3.5   10.5   0.5
Paramaters correct? (Y or N)
y
```

Correlation used to compute stop lines = −0.25
input correlation for use in simulations:
−0.25
input number of Monte Carlo runs
500
Repeat simulations (Y or N)?
n

simulations are performed and output is written to the tab delimited file 'test.out'. A portion of the output is shown below:

TPL parameters

ya	yb	xa	xb	rho	cr	t	zcr	zt	xdel
4.320	1.420	2.380	1.200	−0.25	7.50	5.00	1.28	1.28	0.010000

		Ratio		Pest	
N	Numerator	Upper_limit	Lower_limit	Upper_limit	Lower_limit
20.	1.00	0.43	0.02	6.87	3.135
20.	2.00	0.67	0.08	6.87	3.135
20.	3.00	0.89	0.15	6.87	3.135
20.	4.00	1.09	0.21	6.87	3.135
20.	5.00	1.30	0.30	6.87	3.135
20.	6.00	1.49	0.38	6.87	3.135
20.	7.00	1.68	0.45	6.87	3.135
20.	8.00	1.88	0.55	6.87	3.135
20.	9.00	2.06	0.63	6.87	3.135
20.	10.00	2.24	0.72	6.87	3.135
40.	1.00	0.31	0.04	6.32	3.681
40.	2.00	0.52	0.12	6.32	3.681
40.	3.00	0.71	0.20	6.32	3.681
40.	4.00	0.89	0.29	6.32	3.681
40.	5.00	1.08	0.39	6.32	3.681
40.	6.00	1.26	0.48	6.32	3.681
40.	7.00	1.43	0.57	6.32	3.681
40.	8.00	1.61	0.68	6.32	3.681
40.	9.00	1.77	0.77	6.32	3.681

...

Correlation used in simulations = −0.250
Monte Carlo iterations = 500

Y	R	Pdec1	Pdec2	Pdec3	Pdec4	ASN	PIC	rOC	rASN	tOC	tASN
1.00	3.50	0.910	0.000	0.090	0.000	52.44	0.090	0.910	52.44	1.000	20.00
2.00	3.50	0.964	0.000	0.036	0.000	43.20	0.036	0.958	42.64	1.000	21.24
3.00	3.50	0.994	0.000	0.006	0.000	42.52	0.006	0.980	37.36	1.000	33.16
4.00	3.50	0.966	0.032	0.000	0.002	63.80	0.034	0.990	36.64	0.954	60.76
5.00	3.50	0.544	0.452	0.000	0.004	81.80	0.456	0.992	32.32	0.528	74.44
6.00	3.50	0.088	0.910	0.000	0.002	64.76	0.090	0.996	29.48	0.086	56.96
7.00	3.50	0.034	0.964	0.000	0.002	46.40	0.036	0.998	29.68	0.034	37.12
8.00	3.50	0.012	0.984	0.000	0.004	37.08	0.016	0.996	28.36	0.012	28.64
9.00	3.50	0.000	1.000	0.000	0.000	32.40	0.000	1.000	27.76	0.000	24.44
10.00	3.50	0.000	1.000	0.000	0.000	26.92	0.000	1.000	25.28	0.000	21.64
1.00	4.00	0.848	0.000	0.152	0.000	61.56	0.152	0.848	61.56	1.000	20.00
2.00	4.00	0.948	0.000	0.052	0.000	55.00	0.052	0.938	54.04	1.000	21.60
3.00	4.00	0.990	0.000	0.010	0.000	50.04	0.010	0.968	45.96	1.000	32.76
4.00	4.00	0.952	0.044	0.002	0.002	66.56	0.048	0.968	43.12	0.946	62.56

...

program ratio

```
c   Program written in Fortran 77 compiled using Microsoft FORTRAN
c   compiler v5.1.
c   This program requires a uniform {0,1} random number generator. This
c   version makes use of the RANDOM subroutine provided in Microsoft
c   Fortran. Small sections of code must be changed in the subroutines
c   if a similar is not provided by the compiler.
c   Program written by Jan Nyrop, Department of Entomology, NYSAES,
c   Cornell University, Geneva, NY 14456
c   Last modification date: 7 October, 1992


c ***************************************************************************************
c ******************************** Variable definitions *********************************
c ***************************************************************************************


c asn - average sample size for complete plan determined via simulation
c batch - batch sample size (real)
c cr - critical ratio of pest (y) to natural enemy (x); y/x
c file - name of output file
c i - integer index
c j - integer index
c k - integer index
c l - integer index
c llr - lower limit for ratio determined via interpolation during simulation
c m - integer index
c mci - Monte Carlo iterations (integer)
c mcruns - Monte Carlo iterations (real)
c n(25) - sample size; n(i) is the total number of samples after i batches
c ncount - number of samples taken during a Monte Carlo bout
c ni - batch sample size (integer)
c nsteps - maximum number of batch samples to take during a Monte Carlo bout
c nt - maximum sample size
c pdec1 - probability of making classification 1
c pdec2 - probability of making classification 2
c pdec3 - probability of making classification 3
c pdec4 - probability of making classification 4
c pic - probability of making an incorrect classification
c r1dec - classification 1 counter
c r2dec - classification 2 counter
c r3dec - classification 3 counter
c r4dec - classification 4 counter
c rasn - asn based only on classifying the ratio
c rdel - delta for determining the ratio during simulation
c respond - character keyboard input
c rflag - flag used to indicate when a ratio classification has been made
c rho - correlation between pest and natural enemy used to build stop limits
c rll(100,25) - ratio lower stop limits indexed by pest density (100) and sample size
  (25)
c rmax - maximum ratio used in simulations
c rmin - minimum ratio used in simulations
c roc - OC (P{r ≤ cr}) based only on classifying the ratio
c roccnt - counter for computing roc
c rsamp - counter for computing rasn
c rsteps - number of ratios used in simulation
c rul(100,25) - ratio upper stop limits indexed by pest density (100) and sample
  size (25)
c samples - number of samples taken during a Monte Carlo bout
c simrat - pest/natural enemy ratios used in simulations
c simrho - pest/natural enemy correlation used in simulations
c simx - natural enemy density used in simulations
c simy - pest density used in simulations
```

```
c stdxm - standard deviation of the mean for natural enemy
c stdym - - standard deviation of the mean for pest
c t - pest threshold density
c tab - character tab
c tasn - average sample number based only on classifying pest density
c tflag - flag used to indicate when a pest classification has been made
c tll(25) - pest lower stop limits indexed by sample size (25)
c toc - OC (P{pest ≤ t}) based only on classifying the pest density
c toccnt - counter for computing toc
c totsamp - total samples taken during a simulation for specified pest density and
ratio
c tsamp - counter for computing tasn
c tul(25) - pest upper stop limits indexed by sample size (25)
c ulr - upper limit for ratio determined via interpolation during simulation
c xdel - delta used when determining ratio stop limits
c xm - bivariate normal deviate generated by subroutine binorm used to represent
natural enemy density
c xmean - estimated natural enemy mean density during a Monte Carlo bout
c xtpla - TPL parameter a for natural enemy
c xtplb - TPL parameter b for natural enemy
c y(100) - pest densities used to compute ratio stop limits
c ydel - delta used to compute y(i) when calculating ratio stop limits
c ym - bivariate normal deviate generated by subroutine binorm used to represent
pest density
c ymax - maximum y used to compute ratio limits
c ymean - estimated pest mean density during a Monte Carlo bout
c ymin - minimum y used to compute ratio limits
c ysimstp - number of pest density values used in simulation
c ysteps - number of pest density values used to compute ratio limits
c ytpla - TPL parameter a for pest
c ytplb - TPL parameter b for pest
c zcr - standard normal z used with ratio limits
c zt - standard normal z used with pest limits
c****************** Variable declarations ******************
      real ymin,ymax,ydel,ytpla,ytplb,xtpla,xtplb,rho,cr,t,zcr,zt,xdel,
     + n(25),tul(25),tll(25),y(100),rul(100,25),rll(100,25)
      real rmin,rmax,rdel,rsteps,simrat,simy,simx,batch,r1dec,r2dec,
     +r3dec,r4dec,stdym,stdxm,mcruns,ym,xm,xmean,ymean,ulr,llr,
     +totsamp,pdec1,pdec2,pdec3,pdec4,samples,asn,roccnt,toccnt,
     +roc,toc,rsamp,tsamp,rasn,tasn,pic,simrho
      integer ni,nt,nsteps,ysteps,i,j,ncount,k,l,m,mci,ysimstp,rflag,tflag
      character*1 respond,tab
      character*12 file
      common/params/ytpla,ytplb,xtpla,xtplb,rho,cr,t,zcr,zt,xdel
        tab = char(9)

c open file for output
      write (*,'(a)') ' Enter name for output file (≤ 12 characters)'
      read (*,'(a)') file
      open (unit = 10,file = file,status = 'unknown')

c ***************************************************************************
c ************************* Define models and sampling plan *****************
c ***************************************************************************

1     write (*,*) 'Input model parameters'
      write (*,*) 'TPL parameters (var = a*m**b)'
      write (*,*) 'Parameters for numerator (y):'
      write (*,*) 'a:'
      read (*,*) ytpla
      write (*,*) 'b:'
      read (*,*) ytplb
```

```
      write (*,*) 'Parameters for denominator (x):'
      write (*,*) 'a:'
      read (*,*)xtpla
      write (*,*) 'b:'
      read (*,*) xtplb
      write (*,*) 'correlation coefficient:'
      read (*,*) rho
      write (*,*) 'critical ratio:'
      read (*,*) cr
      write (*,*) 'intervention threshold:'
      read (*,*) t
      write (*,*) 'z for ratio:'
      read (*,*) zcr
      write (*,*) 'z for intervention threshold:'
      read (*,*) zt
      write (*,*) 'delta for determining x:'
      read (*,*) xdel

c display and confirm
      write (*,10)
10      format ('   TPL parameters')
      write (*,11)
11      format ('   ya   yb   xa   xb   rho   cr   t   zcr   zt   xdel
     +')
      write (*,20) ytpla,ytplb,xtpla,xtplb,rho,cr,t,zcr,zt,xdel
20    format (1x,f5.3,1x,f5.3,1x,f5.3,1x,f5.3,1x,f5.2,1x,f4.2,1x,f5.2,
     + 1x,f4.2,1x,f4.2,1x,f8.6)
30    write (*,*) 'Parameters correct? (Y or N)'
      read (*,'(a)') respond
      if (respond.ne.'N'.and.respond.ne.'n') then
        if (respond.ne.'Y'.and.respond.ne.'y') goto 30
      end if
      if (respond .eq. 'N' .or. respond .eq. 'n') goto 1
      write (10,40)
40    format ('          TPL parameters')
      write (10,41)
41    format ('1   ya   yb   xa   xb   rho   cr   t   zcr   zt   xdel
     +')
      write (10,50) ytpla,ytplb,xtpla,xtplb,rho,cr,t,zcr,zt,xdel
50    format (1x,f5.3,1x,f5.3,1x,f5.3,1x,f5.3,1x,f5.2,1x,f4.2,1x,f5.2,
     + 1x,f4.2,1x,f4.2,1x,f8.6)
51    write (°,*) 'Input parameters for specifying numerator (y)'
      write (*,*) 'minimum:'
      read (*,*) ymin
      write (*,*) 'maximum:'
      read (*,*) ymax
      write (*,*) 'delta:'
      read (*,*)ydel
      ysteps  =  aint((ymax − ymin)/ydel) + 1
      ymax  =  ymin + ydel*(ysteps-1)
      write (*,60)
60    format ('   ymin   ymax   ydel')
      write (*,70) ymin,ymax,ydel
70    format (1x,f5.2,1x,f6.2,1x,f6.3)
80    write (*,*) 'Parameters correct? (Y or N)'
      read (*,'(a)') respond
      if (respond.ne.'N'.and.respond.ne.'n') then
        if (respond.ne.'Y'.and.respond.ne.'y') goto 80
      end if
      if (respond .eq. 'N' .or. respond .eq. 'n') goto 51
81    write (*,*) 'Input parameters for specifying sample sizes'
```

```
      write (*,*) 'batch size'
      read (*,*) ni
      write (*,*) 'maximum sample size'
      read (*,*) nt
      nsteps = aint(nt/ni)
      no = ni*steps
      write (*,90)
 90   format (' batch size  maximum')
      write (*,100) ni, nt
 100  format (7x,i4,3x,i6)
 110  write (*,*) 'Parameters correct? (Y or N)'
      read (*,'(a)') respond
      if (respond.ne.'N'and.respond.ne.'n') then
        if (respond.ne.'Y'.and.respond.ne.'y') goto 110
      end if
      if (respond .eq. 'N' .or. respond .eq. 'n') goto 81

c ********************************** Compute stop limits **********************************
c *****************************************************************************************
c ********************************** sequential limits **********************************

      do 220 i = 1,nsteps - 1
        n(i) = float(ni*i)
        call tlim (n(i),tul(i),tll(i))
        do 230 j = 1,ysteps
          y(j) = ymin +(ydel*(j - 1))
          call ratlim (n(i),y(j),rul(j,i),rll(j,i))
 230    continue
 220  continue

c ***** terminal stop limits *****

      n(nsteps) = float(nt)
      tul(nsteps) = t
      tll(nsteps) = t
      do 240 j = 1,ysteps
        rul(j,nsteps) = y(j)/cr
        rll(j,nsteps) = y(j)/cr
 240  continue

c ***** output stop lines *****

 250  write (*,*) 'Display stop limits? (Y or N)'
      read (*,'(a)') respond
      if (respond.ne.'N'.and.respond.ne.'n') then
        if (respond.ne.'Y'.and.respond.ne.'y') goto 250
      end if
      if (respond .eq. 'Y' .or. respond .eq. 'y') then
        write (*,251)
 251    format ('       Ratio limits              Pest limits
     +')
        write (*,252)
 252    format ('       N Numerator Upper_limit Lower_limit Upper_limit
     +Lower_limit')
        ncount = 0
        do 253 i = 1,nsteps
          do 254 j = 1,ysteps
            write (*,255) n(i),y(j),rul(j,i),rll(j,i),tul(i),tll(i)
 255        format (1x,f6.0,3x,f6.2,6x,f6.2,6x,f6.2,6x,f6.2,5x,f6.2)
            ncount = ncount + 1
            if (ncount.gt.20) then
```

```
                pause 'input blank line to continue display'
                ncount = 0
                write (*,256)
256             format ('       N Numerator Upper_limit Lower_limit',1x,
     +          'Upper_limit Lower_limit')
              end if
254         continue
253       continue
          endif
260       write (*,*) 'Write stop limits to file? (Y or N)'
          read (*,'(a)') respond
          if (respond.ne.'N'.and.respond.ne.'n') then
            if (respond.ne.'Y'.and.respond.ne.'y') goto 260
          end if
          if (respond .eq. 'Y' .or. respond .eq. 'y') then
            write (10,261) tab,tab,tab,tab
261         format (1x,a1,a1,' Ratio',a1,a1,'Pest')
            write (10,262) tab,tab,tab,tab,tab
262         format (' N',a1,'Numerator',a1,'Upper_limit',a1,'Lower_limit',
     + a1,'Upper_limit',a1,'Lower_limit')
            do 263 i = 1,nsteps
              do 264 j = 1,ysteps
                write (10,265) n(i),tab,y(j),tab,rul(j,i),tab,rll(j,i),
     + tab,tul(i),tab,tll(i)
265             format (1x,f6.0,a1,f6.2,a1,f6.2,a1,f6.2,a1,f6.2,a1,f6.3)
264           continue
263       continue
          endif


c ***************************************************************************************
c ******************************** Simulate performance *********************************
c ***************************************************************************************


c ***************** input parameters ****************


295       write (*,*) 'stop limits determined for following range of y'
          write (*,300) ymin,ymax
300       format (' minimum = ',f5.2,' maximum = ',f5.2)
305       write (*,*) 'input parameters for specifying y during simulations'
          write (*,*) 'minimum:'
          read (*,*) ymin
          write (*,*) 'maximum:'
          read (*,*) ymax
          write (*,*) 'delta:'
          read (*,*) ydel
          ysimstp = aint((ymax - ymin)/ydel) + 1
          ymax = ymin + ydel*(ysimstp - 1)
          write (*,310)
310       format (' ymin ymax ydel')
          write (*,320) ymin,ymax,ydel
320       format (1x,f5.2,1x,f6.2,1x,f6.3)
330       write (*,*) 'Parameters correct? (Y or N)'
          read (*,'(a)') respond
          if (respond.ne.'N/.and.respond.ne.'n') then
            if (respond.ne.'Y'.and.respond.ne.'y') goto 330
          end if
          if (respond .eq. 'N' .or. respond .eq. 'n') goto 305
335       write (*,*) 'input parameters for specifying ratio'
          write (*,*) 'minimum:'
          read (*,*) rmin
```

```
      write (*,*) 'maximum:'
      read (*,*) rmax
      write (*,*) 'delta:'
      read (*,*) rdel
      rsteps = aint((rmax - rmin)/rdel) + 1
      rmax = rmin + rdel*(rsteps - 1)
      write (*,340)
  340 format (' rmin rmax rdel')
      write (*,350) rmin,rmax,rdel
  350 format (1x,f5.2,1x,f6.2,1x,f6.3)
  360 write (*,*) 'Parameters correct? (Y or N)'
      read (*,'(a)') respond
      if (respond.ne.'N'.and.respond.ne.'n') then
        if (respond.ne.'Y'.and.respond.ne.'y') goto 360
      end if
      if (respond .eq. 'N' .or. respond .eq. 'n') goto 335
      write (*,365) rho
  365 format (' Correlation used to compute stop lines = ',f6.3)
      write (*,*) 'Input correlation for use in simulations:'
      read (*,*) simrho
      write (*,*) 'input number of Monte Carlo runs'
      read (*,*) mci
      mcruns = float(mci)
      batch = float(ni)
      write (10,366) simrho
  366 format (' Correlation used in simulations = ',f6.3)
      write (10,367) mci
  367 format (' Monte carlo iterations = ',i4)
      write (*,370)
  370 format ('Y R Pdec1 Pdec2 Pdec3 Pdec4 ASN PIC')
      write (10,380) tab,tab,tab,tab,tab,tab,tab,tab,tab,tab,tab
  380 format (' Y',a1,'R',a1,'Pdec1',a1,'Pdec2',a1,'Pdec3',a1,'Pdec4',
     +a1,'ASN',a1,'PIC',a1,'rOC',a1,'rASN',a1,'tOC',a1,'tASN')

c ******************** Computations ********************

c     ***** loop for ratio *****

      do 400 k = 1,rsteps
        simrat = rmin + rdel*(k - 1)

c     ***** loop for numerator (y) *****

          do 410 l = 1,ysimstp
            simy = ymin + ydel*(l - 1)
            simx = simy/simrat
            r1dec = 0.0
            r2dec = 0.0
            r3dec = 0.0
            r4dec = 0.0
            totsamp = 0.0
            roccnt = 0.0
            toccnt = 0.0
            rsamp = 0.0
            tsamp = 0.0
            stdym = sqrt((ytpla*simy**ytplb)/batch)
            stdsm = sqrt((xtpla*simx**xtplb)/batch)

c     ***** loop for Monte Carlo runs *****
```

```
      do 420 m = 1,mci
        xmean = 0.0
        ymean = 0.0
        rflag = 0
        tflag = 0
        do 430 i = 1,nsteps
          samples = float(i*ni)
          call binorm (symy,simx,stdym,stdxm,simrho,ym,xm)
          xmean = ((xmean*float((i-1)*ni))+(xm*batch))/samples
          ymean = ((ymean*float((i-1)*ni))+(ym*batch))/samples
          call interp (rul(1,i),y,ymean,ysteps,0,ulr)
          call interp (rll(1,i),y,ymean,ysteps,0,llr)
          if (rflag.eq.0) then
            if (xmean.ge.ulr) then
              rflag = 1
              roccnt = roccnt + 1.0
              rsamp = rsamp + samples
            end if
            if (xmean.lt.llr) then
              rflat = 1
              rsamp = rsamp + samples
            end if
          end if
          if (tflag.eq.0) then
            if (ymean.le.tll(i)) then
              tflag = 1
              toccnt = toccnt + 1.0
              tsamp = tsamp + samples
            end if
            if (ymean.gt.tul(i)) then
              tflag = 1
              tsamp = tsamp + samples
            end if
          end if
          if (ymean.le.tll(i).and.xmean.ge.ulr) then
            r1dec = r1dec + 1.0
            totsamp = totsamp + samples
            goto 500
          endif
          if (ymean.gt.tul(i).and.xmean.ge.ulr) then
            r2dec = r2dec + 1.0
            totsamp = totsamp + samples
            goto 500
          endif
          if (ymean.le.tll(i).and.xmean.t.llr) then
            r3dec = r3dec + 1.0
            totsamp = totsamp + samples
            goto 500
          endif
          if (ymean.gt.tul(i).and.xmean.lt.llr) then
            r4dec = r4dec + 1.0
            totsamp = totsamp + samples
            goto 500
          endif
430       continue
500       continue
420     continue

c ***************************************************************************
c ********************************** Output results **************************
c ***************************************************************************
```

```
            pdec1 = r1dec/mcruns
            pdec2 = r2dec/mcruns
            pdec3 = r3dec/mcruns
            pdec4 = r4dec/mcruns
            asn = totsamp/mcruns
            if (simy.le.t.and.simrat.le.cr) pic = 1-pdec1
            if (symy.gy.t.and.simrat.le.cr) pic = 1 - pdec2
            if (simy.le.t.and.simrat.gt.cr) pic = 1 - pdec3
            if (symy.gt.t.and.simrat.gt.cr) pic = 1 - pdec4
            roc = roccnt/mcruns
            toc = toccnt/mcruns
            rasn = rsamp/mcruns
            tasn = tsamp/mcruns
            write (*,510) simy,simrat,pdec1,pdec2,pdec3,pdec4,asn,pic
510         format (1x,f6.2,2x,f6.2,2x,4(f5.3,2x),f7.2,2x,f5.3)
            write (10,520) simy,tab,simrat,tab,pdec1,tab,pdec2,tab,pdec3,
     +tab,pdec4,tab,asn,tab,pic,tab,roc,tab,rasn,tab,toc,tab,tasn,
     +tab
520         format (1x,f6.2,a1,f6.2,a1,4(f5.3,a1),f7.2,a1,f5.3,a1,2(f5.3,a1,f7.2,a1))
410       continue
400     continue

600     write (*,*) 'Repeat simulations (Y or N)?'
        read (*,'(a)') respond
        if (respond.ne.'N'.and.respond.ne.'n') then
          if (respond.ne.'Y'.and.respond.ne.'y') goto 600
        end if
        if (respond .eq. 'Y' .or. respond .eq. 'y') goto 295
        close (10)
        stop
        end

c ***************************************************************************************
c ************************************ Subroutines ***********************************
c ***************************************************************************************

        subroutine tlim (n,ul,ll)

c   ***************************************************************************************
c   Subroutine computes stop limits for comparing pest density to
c   ***************************************************************************************

        real n,ul,ll,var,ytpla,ytplb,xtpla,xtplb,rho,cr,t,zcr,zt,xdel
        common/params/ytpla,ytplb,xtpla,xtplb,rho,cr,t,zcr,zt,xdel
        var = ytpla*t**ytplb
        ul = t+zt*sqrt(var/n)
        ll = t-zt*sqrt(var/n)
        return
        end

        subroutine ratlim(n,y,ul,ll)

c   ***************************************************************************************
c   Subroutine computes stop limits for comparing natural enemy density
c   to in order to classify the pest-natural enemy ratio
c   ***************************************************************************************

        real n,ul,ll,ytpla,ytplb,xtpla,xtplb,rho,cr,t,zcr,zt,xdel,
```

```
     +x,vy,vx,cyy,cxx,cyx,var1,var2,var3,var4,rhat,rhigh,rlow
      integer j
      common/params/ytpla,ytplb,xtpla,xtplb,rho,cr,t,zcr,zt,xdel
      do 10 j = 1,2

c   *** note: j = 1 for upper interval and j = 2 for lower interval ***

        x = y/cr
20      vy = ytpla*y**ytplb
        vx = xtpla*x**xtplb
        cyy = vy/(n*y**2)
        cxx = vx/(n*x**2)
        cyx = (rho*sqrt(vx)*sqrt(vy))/(n*x*y)
        var1 = (cyy+cxx-(2*cyx))-(zcr**2*(cyy*cxx-cyx**2))
        rhat = y/x

c   ***** test for imaginary roots of var1 *****

        if (var1.lt.0.0.and.j.eq.1) then
          x = x+xdel
          goto 20
        end if
        if (var1.t.0.0.and.j.eq.2) then
          x = x-xdel
          if (x.lt.0.0) then
            ll = -999.0
            goto 10
          endif
          goto 20
        end if

c   ***** compute confidence intervals *****

      var2 = 1-(zcr**2*cyx)
      var3 = 1-(zcr**2*cxx)
      var4 = zcr*sqrt(var1)
      if (j.eq.1) then
        rhigh = rhat*(var2+var4)/var3
      else
        rlow = rhat*(var2-var4)/var3
      endif

test if conditions have been met for confidence limits to be stop limits

      if (j.eq.1) then
        if (rhigh.lt.cr.and.rhigh.gt.0.0) then
          ul = x
        else
          x = x+xdel
          goto 20
        endif
      else
        if (rlow.gt.cr) then
          ll = x
        else
          x = x-xdel
          if (x.lt.0.0) then
            ll = -999.0
            goto 10
```

```
            endif
            goto 20
          endif
        endif
10      continue
        return
        end

        subroutine binorm (m1,m2,stdv1,stdv2,rho,x1,x2)

c   ********************************************************************************
c   Subroutine computes bivariate normally distributed random variables
c   using the method described by Johnson and Kotz in Distributions in
c   Statistics: Continuous Multivariate Distributions. John Wiley & Sons,
c   New York. The subroutine use a uniform {0,1} random variate which
c   must be provided if the FORTRAN compiler does not support
c   it [call random(r1) below].
c   ********************************************************************************

        real m1,m2,stdv1,stdv2,rho,r1,r2,u1,u2,y1,y2,x1,x2

c generate first uniform normal variate

        call random(r1)
        if (r1.le.0.0) r1 = 0.0000001
        call random(r2)
        u1 = sqrt(-2.*log(r1))*(cos(6.283*r2))

c generate second uniform normal variate

        call random(r1)
        if (r1.le.0.0) r1 = 0.0000001
        call random(r2)
        u2 = sqrt(-2.*log(r1))*(cos(6.283*r2))
        y1 = u1
        y2 = rho*u1 + sqrt(1 - rho**2)*u2
        x1 = m1 + y1*stdv1
        x2 = m2 + y2*stdv2
        return
        end

        subroutine interp (y,x,dummyx,k,trunc,yvalue)

c   ********************************************************************************
c   This subroutine performs a linear interpolation of a function y = f(x).
c   Vectors of y values (y) and x values (x) are passed along with a dummy
c   x value (dummyx) for which a y value (yvalue) is sought. The dimension
c   of the y and x arrays (k) must also be passed. In addition a variable
c   trunc is passed that when = 1 causes trunction of the function so that
c   when dummyx < x(1) or dummyx > x(k), yvalue is constrained to be y(1) or
c   y(k). When trunc is not equal to 1, yvalue is interpolated beyond y(1)
c   or y(k).
c   ********************************************************************************

        real y,x,dummyx,yvalue
        integer k,trunc,j
        dimension y(k),x(k)

        if (dummyx.le.x(1)) then
```

```
        if (trunc.eq.1) then
          yvalue = y(1)
          goto 20
        else
          yvalue = y(1)-((y(2)-y(1))/(x(2)-x(1))*(x(1)-dummyx))
          goto 20
        endif
      endif

      do 10 j = 2,k
        if (dummyx.le.x(j)) then
          yvalue = y(j - 1)+((y(j)-y(j-1))/(x(j)-x(j - 1))*(dummyx-x(j - 1)))
          goto 20
        endif
10    continue

      if (trunc.eq.1) then
        yvalue = y(k)
        goto 20
      else
        yvalue = y(k)+((y(k)-y(k-1))/(x(k)-x(k - 1))*(dummyx-x(k)))
        goto 20
      endif

20    continue
      return
      end
```

sprt: sequential classification

Background: Program sprt.for analyzes the performance of dichotomous and tripartite sequential classification sampling plans based on Wald's sequential probability ratio test (SPRT). Tripartite plans are based on two dichotomous plans. Five models can be used to describe sample observations; binomial, Poisson, and negative binomial distributions and two models where binomial counts are substituted for complete enumeration. One binomial count model is the empirical relationship $\ln(-\ln(1 - p_T)) = \gamma + \delta n(m)$ and the other is based on the negative binomial distribution

$$P_T = 1 - \sum_{v=0}^{T} \Pr\{x = v\}$$

In both of these equations P_T is the proportion of sample units with more than T organisms. T is referred to as a tally threshold.

OC and ASN functions for dichotomous plans based on the three probability distribution functions are computed using an approximate formula. When the negative binomial distribution is specified, the parameter k can either be a function of the mean by modeling the variance via TPL ($s^2 = \alpha m^{\beta}$) or k can be a constant. For the negative binomial distribution OC and ASN functions are first computed using a nominal value of k. When $k = f(\text{TPL})$ this is the k that corresponds to a mean equal to the average of the means used to specify the null and alternate hypotheses. OC and ASN functions for the negative binomial distribution can also be simulated. For $k = f(\text{TPL})$, k in the simulation is computed using TPL. When k is a constant a new k for the simulations is input. OC and ASN functions for binomial count plans are the expected values described by Nyrop and Binns.[3]

The program functions as follows:

1. A sample model is chosen and data required to describe the model are input. Required data for each model are listed below:

   (i) Binomial: none
   (ii) Poisson: none
   (iii) Negative binomial: if k = f(TPL), intercept and slope for TPL model.
   (iv) Empirical p – m: intercept, slope, variance of the slope, mean square error, number of data points, and mean of independent variable used in regression
   (v) NGB based p – m: intercept, slope variance of the slope, mean square error, number of data points, and mean of independent variable used in TPL regression

2. Parameters for the first SPRT are input. If a tripartite plan is to be constructed the first SPRT is the one used to classify the density into the lowest category. If binomial count models are used, a threshold density is first specified and then the proportion of occupied sample units is computed and displayed. The SPRT is based on this proportion.
3. OC and ASN functions for the specified plan are computed and displayed. Displayed values are interpolated from a more complete set of OC and ASN values. Interpolated values are shown to facilitate comparison of plans. If desired, the complete set of OC and ASN values can be written to the output file. If the OC and ASN values are not acceptable, new SPRT parameters can be entered.
4. Steps 2 and 3 are repeated for the second SPRT of a tripartite plan. If only a dichotomous SPRT is desired, program execution can be halted.
5. Both SPRT plans are displayed. If desired, parameters for either plan can be changed and OC and ASN functions are recomputed.
6. The two SPRT plans are used to construct a tripartite plan and the performance of this plan is simulated. To start the simulations the number of Monte Carlo runs as well as the maximum sample size are input. If the maximum sample size is reached, sampling is terminated by comparing the sample mean to the midpoints of the means used to form the null and alternate hypotheses. The means of populations for which sampling is simulated are automatically determined. This is done as follows: for each SPRT, OC and ASN values are determined for 29 means. The means used in the tripartite simulation are determined as:

   i) The first 18 means from the first SPRT
   ii) Means 19–29 from the first SPRT unless a mean from the first SPRT > the first mean from the second SPRT
   iii) All the means from the second SPRT

When two SPRTs are combined to form a tripartite plan it is possible for the lower stop limit of the uppermost set of limits to lie below the lower limit of the lowermost set of limits for some means. It is also possible for the upper stop limit for the lowermost set of limits to lie above the upper limit of the uppermost set of limits. When either of these situations occur in the simulation, a decision to stop sampling is not made until sample data lie beyond both sets of stop limits.

An example run using a binomial count model based on the negative binomial distribution is shown below. User inputs are underlined and comments are in italics.

```
> sprt
Enter name for output file (≤ 12 characters)
```

ngb.txt
Choose a sampling model
1) Binomial
2) Poisson
3) Negative binomial
4) Empirical p-m
5) NGB based p-m
enter a number 1-5-m
5
Input TPL parameters
intercept;
4.32
slope;
1.42
variance of the slope;
0.004
mse from the regression;
0.278
number of data points for regression;
147
mean of the independent variable {lnm};
0.728

NGB p-m model with TPL used

intercept	slope	var(slope)	mse	N	mean of lnm
4.320	1.420	0.0040	0.278	147.0	0.728

Parameters correct? (Y or N)
y
Enter plan number; 0 *plan number is initially set to 0*
1
threshold as density; 0.0 *a threshold density is required to compute P_T it is initially set to 0.0*
1.0
tally threshold; 0.0
0
Intervention proportion = 0.3564
mean for null hypothesis;
0.306
mean for alternative hypothesis;
0.406
alpha;
0.075
beta;
0.075

H0 mean	H1 mean	k	alpha	beta	
0.306	0.406	0.301	0.075	0.075	*k is computed using TPL*

Parameters correct? (Y or N)
y

OC	Mean	ASN
0.99	0.39	72.15
0.90	0.55	82.85
0.80	0.67	86.14
0.70	0.78	87.56
0.60	0.88	88.12
0.50	0.99	87.82
0.40	1.11	82.89
0.30	1.28	74.20
0.20	1.50	64.32
0.10	1.90	51.22
0.01	3.32	22.30

Redo OC and ASN? (Y or N)
n
Write OC and ASN functions to file? (Y or N)
n
Stop now? (Y or N)
n
Enter plan number; 1
2
threshold as density; 0.0
2.5
tally threshold; 0.0 *tally threshold for plan 2 = that for plan 1*
intervention proportion = 0.5785
mean for null hypothesis;
0.529
mean for alternative hypothesis;
0.628
alpha;
0.075
beta;
0.075

H0 mean	H1 mean	k	alpha	beta	
0.529	0.628	0.467	0.075	0.075	*H0 and H1 means for plan 2 are larger than those for plan 1*

Parameters correct? (Y or N)
y

OC	Mean	ASN
0.99	0.85	31.45
0.90	1.37	60.06
0.80	1.68	71.46
0.70	1.95	77.19
0.60	2.22	81.48
0.50	2.50	82.67
0.40	2.81	79.14
0.30	3.26	72.70
0.20	3.85	64.11
0.10	4.88	49.46
0.01	8.71	21.82

Redo OC and ASN? (Y or N)
n
Write OC and ASN functions to file? (Y or N)
n

Stop now? (Y or N)
n

NEG p-m model with TPL used

intercept	slope	var(slope)	mse	N	mean of lnm
4.320	1.420	0.0040	0.278	147.0	0.728

H0 mean	H1 mean	k	alpha	beta	H0 mean	H1 mean	k	alpha	beta
0.306	0.406	0.301	0.075	0.075	0.529	0.628	0.467	0.075	0.075

Low intercept	High intercept	Slope	Low intercept	High intercept	Slope
−5.73	5.73	0.355	−6.16	6.16	0.579

intercepts and slopes are those for stop lines

OC	Mean	ASN	OC	Mean	ASN
0.99	0.39	72.15	0.99	0.85	31.45
0.90	0.55	82.85	0.90	1.37	60.06
0.80	0.67	86.14	0.80	1.68	71.46
0.70	0.78	87.56	0.70	1.95	77.19
0.60	0.88	88.12	0.60	2.22	81.48
0.50	0.99	87.82	0.50	2.50	82.67
0.40	1.11	82.89	0.40	2.81	79.14
0.30	1.28	74.20	0.30	3.26	72.70
0.20	1.50	64.32	0.20	3.85	64.11
0.10	1.90	51.22	0.10	4.88	49.46
0.01	3.32	22.30	0.01	8.71	21.82

Redo a plan? (Y or N)?
n
starting simulations...
Maximum sample size?
100
Monte Carlo iterations?
500
Simulations are performed and output is written to the tab delimited file 'ngb.txt'.
A portion of the output is shown below:

Max. sample size = 100 iterations = 500

number of means = 45

Mean	P{dec1}	P{dec2}	P{dec3}	ASN
0.357	0.993	0.007	0.000	37.60
0.396	0.985	0.015	0.000	41.09
0.437	0.968	0.032	0.000	44.92
0.480	0.946	0.054	0.000	48.75
0.527	0.916	0.084	0.000	52.52
0.576	0.880	0.120	0.000	56.22
0.628	0.839	0.161	0.000	59.68
0.682	0.793	0.206	0.001	62.99
0.739	0.735	0.263	0.002	66.04
0.797	0.680	0.316	0.004	68.51
0.809	0.670	0.326	0.005	68.99
0.821	0.657	0.337	0.005	69.45

P{dec1} is the probability of classifying the density as ≤ 1.0, *P{dec2} is the probability of classifying the density as* > 1.0 *and* ≤ 2.5, *and P{dec3} is the probability of classifying the density as* > 2.5

```
      program sprt

c     Program written in Fortran 77: IBM version compiled using Microsoft
c     FORTRAN compiler v5.1.
c     This program requires a uniform {0,1} random number generator. This
c     version makes use of the RANDOM subroutine provided in Microsoft
c     Fortran. If the FORTRAN compiler used does not provide a similar
c     function, one must be provided and small sections of code in some
c     subroutines must be changed.
c     Program written by Jan Nyrop, Department of Entomology, NYSAES,
c     Cornell University, Geneva, NY 14456
c     Last modification date: 3 August, 1992

c ***************************************************************************
c ****************************** Variable definitions ***********************
c ***************************************************************************

c     a: intermediate SPRT variable
c     alpha(2): alpha for SPRT plans 1 and 2
c     asn(29,2): ASN corresponding to OC(29,plan(2))
c     asnprt(11,2): displayed ASN values corresponding to mean(11,sampling
      plan(2))
c     b: intermediate SPRT variable
c     beta(2): beta for SPRT plans 1 and 2
c     bvar1: variable used to compute binomial SPRT
c     c: intercept for empirical p-m model (ln(-ln(p)) = c + d(ln(m))
c     confirm: integer used to flag whether sampling model has been confirmed
c     count: number of samples generated during one simulated sampling bout
c     d: slope for empirical p-m model (ln(-ln(p)) = c + d(ln(m))
c     distr: distribution index
c     dummy: x value passed to interpolation routine for which a y value is
      returned
c     easn: expected average sample size when binomial counts are substituted for
      complete counts
c     edec: expected probability of a decision (e.g., OC) when binomial counts are
      substituted for complete count
c     file: name for output file
c     h(14): variable used in computation of dichotomous SPRT OC
c     highi(2): stop limit parameter for SPRT plans 1 & 2
c     i: plan number (1 or 2)
c     i2: do loop index
c     icount: index for simmean
```

```
c       initun: initial file unit number
c       init(2): used to determine if plans 1 & 2 have been initialized
c       ivalues: number of values passed to interpolation routine
c       j: index
c       jj: index
c       k(2): NGB k used to construct dichotomous SPRT
c       ksim: NGB k used in tripartite SPRT simulations
c       lna: ln(a) - intermediate SPRT variable
c       lnb: ln(b) - intermediate SPRT variable
c       lowi(2): stop limit parameter for SPRT plans 1 & 2
c       m(29,2): mean corresponding to an OC(29,plan(2))
c       m0(2): null hypothesis mean for SPRT plan 1 & 2
c       m1(2): alternate hypothesis mean for SPRT plans 1 & 2
c       mci: Monte Carlo iterations
c       mean(11,2): means corresponding to ocprint(11,plan(2))
c       mlnm: mean of ln(m) used in empirical p-m or TPL model
c       mse: mean square error for either empirical p-m or TPL model
c       mt: threshold density used to compute a threshold proportion for binomial
        count sampling
c       nmax: maximum sample size for simulating tripartite SPRT OC and ASN
c       ngbk: flag to indicate whether constant k or k = f(TPL) is to be used
c       oc(29,2): dichotomous SPRT OC function for m(29, plans(2))
c       ocprint(11): SPRT OC values for which means are displayed on screen
c       p: NGB p corresponding to an OC value
c       pdec1(58): P{decision m < m0(1)}
c       pdec2(58): P{decision m1(1) < m < m0(2)}
c       pdec3(58): P{decision m > m1(2)}
c       pi(2): threshold density expressed as a proportion of binomial counts
c       pmbin(29,2): mean density corresponding to threshold proportions used to
        compute OC and ASN functions for dichotomous SPRT
c       pnb0: NGB parameters for m1
c       pnb1: NGB parameters for m0
c       pxgtt: probability that x > t when x is distributed as NGB
c       q: NGB q
c       qnb0: NGB parameter for m1
c       qnb1: NGB parameter for m0
c       q0: binomial parameter for m0
c       q1: binomial parameter for m1
c       refm: a reference set of means for which pxgtt is calculated
c       respond: used to record keyboard input to questions (Y, N)
c       result: y value returned from interpolation routine
c       revoc(29,2): 1 - OC; used in interpolation routine
c       rn: number of data points in either TPL or empirical p-m model
c       seq3: number of  > m1(2) sequential decisions
c       seq2: number of  > m1(1) and  < m0(2) sequential decisions
c       seq2dec: P{m1(1) < m < m0(2)|m} for sequential decisions
c       seq1: number of  < m0(1) sequential decisions
c       seq1dec: P{m < m0(1)|m} for sequential decisions
c       simasn(58): ASN for simulated means
c       simmean(58): means for which P{dec} and ASN functions are to be simulated
c       sll(2): stop line
c       slope(2): stop limit parameters for SPRT plans 1 & 2
c       slu(2): stop line
c       spmbin(58): means for which P{dec} and ASN functions are to be simulated;
c                   means based on binomial count proportions
c       sum: sum of NGB variates during a simulated sampling bout
c       sumpdec: sum of P{deci} functions used to standardize so sum  =  1.0
c       t: tab character
c       tally: integer used to define a positive binomial count
c       temp:
c       ter3: number of  > m1(2) terminal decisions
```

```
c       ter2: number of  > m1(1) and  < m0(2) terminal decisions
c       ter2dec: P{m1(1) < m < m2(0)|m} for terminal decisions
c       ter1: number of  < m0(1) terminal decisions
c       ter1dec: P{m < m0(1)|m} for terminal decisions
c       thresh(2): threshold for SPRT plans 1 & 2
c       totcnt: total samples generated for simulated sampling from a mean
c       tpla: a for var  =  a(mean)^b
c       tplb: b for var  =  a(mean)^b
c       unit: file connection number
c       var: computed variance
c       vb: variance of the slope b
c       vd: variance of the slope d
c       vnb1: intermediate variable for computing stop limits
c       vnb2: intermediate variable for computing stop limits
c       weight: weights for computing expected asn and P{dec} functions for binomial count sampling
c       x: random variate generated during simulations
c       xbar: average of x values

        character*1 respond,t
        character*12 file
        integer i,i2,icount,init(2),ivalues,j,jj,mci,nmax,unit,initun,distr,ngbk,confirm
        real a,alpha(2),asn(29,2),asnprt(11,2),b,beta(2),count,
       +dummy,h(14),highi(2),k(2),ksim,lna,lnb,lowi(2),m(29,2),m0(2),
       +m1(2),mean(11,2),oc(29,2),ocprint(11),p,pdec1(58),pdec2(58),
       +pdec3(58),pnb0,pnb1,q,qnb0,qnb1,result,revoc(29,2),seq3,seq2,
       +seq2dec,seq1,seq1dec,simasn(58),q0,q1,bvar1,sumpdec,temp
        real simmean(58),sll(2),slope(2),slu(2),sum,ter3,ter2,ter2dec,
       +ter1,ter1dec,thresh(2),totcnt,tpla,tplb,vnb1,vnb2,x,xbar
        real c,d,vd,mse,rn,mlnm,weight(11),z(11),edec(58),easn(58),
       +mt(2),pi(2),pmbin(29,2),spmbin(58),temp1(58),temp2(58),temp3(58),
       +vb,tally,var,pxsum,refm(101),pxgtt(101)

        data h/5,4.5,4.,3.5,3.,2.5,2.,1.5,1.,.5,.4,.3,.2,.1/
        data ocprint/0.99,0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1,0.01/
        data init/0,0/
        data weight /0.1974,0.1856,0.1638,0.1358,0.1062,0.0776,+0.0534,0.0346,0.0212,
        0.012,0.0064/
        data z /0.0,0.375,0.625,0.875,1.125,1.375,1.625,1.875,2.125,2.375,2.625/

        t  =  char(9)
        tpla  =  0.
        tplb  =  0.
        i  =  0.

c open file for output

        write (*,'(a)') ' Enter name for output file (≤ 12 characters)'
        read (*,'(a)') file
        open (unit = 10,file  =  file,status  =  'unknown')

c the initialized value of unit is the screen

        initun  =  0
        unit  =  initun

c ***************************************************************************
c ************************ Model selection and parameterization ************************
c ***************************************************************************

        confirm  =  0
```

```
500   write (*,*) 'Choose a sampling model'
      write (*,*) '1) binomial'
      write (*,*) '2) Poisson'
      write (*,*) '3) Negative binomial'
      write (*,*) '4) Empirical p-m'
      write (*,*) '5) NGB based p-m'
      write (*,*) 'enter a number 1-5'
      read (*,*) distr
      if (distr.lt.1.or.distr.gt.5) goto 500
505   continue

      if (distr.eq.1) then
        write (unit,510)
510     format (' Binomial distribution used')
      endif

      if (distr.eq.2) then
        write (unit,520)
520     format (' Poisson distribution used')
      endif
      if (distr.eq.3) then
        if (confirm.eq.0) then
530       write (*,*) 'use 1) constant k or 2) k  =  f(TPL)'
          read (*,*) ngbk
          if (ngbk.ne.1.and.ngbk.ne.2) goto 530
        end if
        if (ngbk.eq.1) then
          write (unit,540)
540       format (' NGB model with constant k used')
        else
          if (confirm,eq.0) then
            write (*,*) 'Input TPL parameters'
            write (*,*) 'intercept;'
            read (*,*) tpla
            write (*,*) 'slope;'
            read (*,*) tplb
          end if
          write (unit,550) tpla,tplb
550       format (' NGB model with k  =  f(TPL) used',/,
     +      'a  =  ',f5.3,1x,'b  = ',f5.3)
        end if
      end if

      if (distr.eq.4) then
        if (confirm.eq.0) then
          write (*,*) 'Input empirical p-m model parameters'
          write (*,*) 'intercept;'
          read (*,*) c
          write (*,*) 'slope;'
          read (*,*) d
          write  (*,*) 'variance of the slope;'
          read (*,*) vd
          write (*,*) 'mse from the p-m regression;'
          read (*,*) mse
          write (*,*) 'number of data points for p-m regression;'
          read (*,*) rn
          write (*,*) 'mean of the independent variable {lnm};'
          read (*,*) mlnm
        end if
        write (unit,560)
```

```
560     format(' Empirical p-m model used',/,
     +    ' intercept  slope  var(slope)  mse  N  mean of lnm'}
          write (unit,565) c,d,vd,mse,rn,mlnm
565       format (3x,f6.3,1x,f6.3,5x,f7.4,1x,f6.3,1x,f4.0,3x,f8.3)
        endif

        if (distr.eq.5) then
          if (confirm.eq.0) then
            write (*,*) 'Input TPL parameters'
            write (*,*) 'intercept;'
            read (*,*) tpla
            write (*,*) 'slope;'
            read (*,*) tplb
            write (*,*) 'variance of the slope;'
            read (*,*) vb
            write (*,*) 'mse from the regression;'
            read (*,*) mse
            write (*,*) 'number of data points for regression;'
            read (*,*) rn
            write (*,*) 'mean of the independent variable {lnm};'
            read (*,*) mlnm
          endif
          write (unit,570)
570       format(' NGB p-m model with TPL used',/,
     +    ' intercept  slope  var(slope)  mse  N  mean of lnm')
          write (unit,575) tpla,tplb,vb,mse,rn,mlnm
575       format (3x,f6.3,1x,f6.3,5x,f7.4,1x,f6.3,1x,f4.0,3x,f8.3)
        end if

        if (confirm.eq.0) then
580       write (*,*) 'Parameters correct? (Y or N)'
          read (*,'(a)') respond
          if (respond.ne.'N'.and.respond.ne.'n') then
            if (respond.ne.'Y'.and.respond.ne.'y') goto 580
          end if
          if (respond .eq. 'N' .or. respond .eq. 'n') then
            goto 500
          else
            unit = 10
            confirm = 1
            goto 505
          end if
        end if

        unit = initun

c ***************************************************************************************
c ******************************** SPRT parameterization ********************************
c ***************************************************************************************

7       continue
        write (*,8) i
8       format (' Enter plan number; ',i3)
        read (*,*) i
        if (i.ne.1.and.i.ne.2) goto 7
        if (i.eq.1.and.init(1).eq.0) init(1) = 1
        if (i.eq.2.and.init(2).eq.0) init(2) = 1

9       if (distr.eq.4.or.distr.eq.5) then
          write (*,10) mt(i)
10        format (' threshold as density; ',f6.3)
          read (*,*) mt(i)
```

```
          if (distr.eq.4) then
            pi(i) = 1-exp(-exp(c+d*log(mt(i))))
          else
            write (*,11) tally
11          format (' tally threshold; ',f4.0)
            if (i.eq.1) then
              read (*,*) tally
            endif
            var = tpla*mt(i)**tplb
            k(i) = mt(i)**2/(var-mt(i))
            call ngbcdf(mt(i),k(i),tally,pxsum)
            pi(i) = 1.0-pxsum
          endif
          write (*,12) pi(i),k(i)
12        format (' intervention proportion = ',f6.4,
     + ' k based on TPL = ',f6.4)
        endif

        write (*,13) m0(i)
13      format ('mean for null hypothesis; ', f6.3)
        read (*,*) m0(i)
        write (*,14) m1(i)
14      format (' mean for alternative hypothesis; ',f6.3)
        read (*,*) m1(i)
        write (*,15) alpha(i)
15      format (' alpha; ',f5.3)
        read (*,*) alpha(i)
        write (*,16) beta(i)
16      format (' beta; ',f5.3)
        read (*,*) beta(i)
        thresh(i) = (m0(i)+m1(i))/2.
        if (distr.eq.3) then
          if (ngbk.eq.1) then
            write (*,17) k(i)
17          format (' NGB k; ',f5.3)
            read (*,*) k(i)
          else
            k(i) = thresh(i)**2/(tpla*thresh(i)**tplb-thresh(i))
          end if
        end if

        write (*,18)
18      format (' H0 mean   H1 mean          alpha   beta   NGB k')
        write (*,19) m0(i),m1(i),alpha(i),beta(i),k(i)
19      format(f7.3,3x,f7.3,11x,f4.3,4x,f4.3,4x,f5.3)
20      write (*,*) 'Parameters correct? (Y or N)'
        read (*,'(a)') respond
        if (respond.ne.'N'.and.respond.ne.'n') then
          if (respond.ne.'Y'.and.respond.ne.'y') goto 20
        end if
        if (respond .eq. 'N' .or. respond .eq. 'n') goto 9

c ***************************************************************************************
c ********************************** SPRT computations **********************************
c ***************************************************************************************

c variables used for all distributions

        a = (1-beta(i))/alpha(i)
        lna = log(a)
        b = beta(i)/(1-alpha(i))
        lnb = log(b)
```

```
c variables used for binomial and negative binomial

      if (distr.eq.1.or.distr.eq.4.or.distr.eq.5) then
        q1 = 1-m1(i)
        q0 = 1-m0(i)
        bvar1 = log((m1(i)*q0)/(m0(i)*q1))
      end if
      if (distr.eq.3) then
        pnb0 = m0(i)/k(i)
        pnb1 = m1(i)/k(i)
        qnb0 = 1 + pnb0
        qnb1 = 1 + pnb1
        vnb1 = (pnb1*qnb0)/(pnb0*qnb1)
        vnb2 = qnb0/qnb1
      end if

c calculate stop limit parameters

      if (distr.eq.1.or.distr.eq.4.or.distr.eq.5) then
        lowi(i) = lnb/bvar1
        highi(i) = lna/bvar1
        slope(i) = log(q0/q1)/bvar1
      else if (distr.eq.2) then
        lowi(i) = lnb/log(m1(i)/m0(i))
        highi(i) = lna/log(m1(i)/m0(i))
        slope(i) = (m1(i) - m0(i))/log(m1(i)/m0(i))
      else
        lowi(i) = lnb/log(vnb1)
        highi(i) = lna/log(vnb1)
        slope(i) = k(i)*(log(qnb1/qnb0)/log(vnb1))
      end if

c OC and ASN computations

      do 30 j = 1,14
        oc(j,i) = (a**h(j)-1)/(a**h(j)-b**h(j))
        revoc(j,i) = 1-oc(j,i)
        if (distr.eq.1.or.distr.eq.4.or.distr.eq.5) then
          m(j,i) = (1-(q1/q0)**h(j))/((m1(i)/m0(i))**h(j)-(q1/q0)
     + **h(j))
          asn(j,i) = (lnb*oc(j,i)+(1-oc(j,i))*lna)/m(j,i)*bvar1
     +log(q1/q0))
        else if (distr.eq.2) then
          m(j,i) = ((m1(i)-m0(i))*h(j))/((m1(i)/m0(i))**h(j)-1)
          asn(j,i) = (lnb*oc(j,i)+lna*(1-oc(j,i)))/(m(j,i)*
     +log(m1(i)/m0(i))+m0(i)-m1(i))
        else
          p = (1 - vnb2**h(j))/(vnb1**h(j) - 1)
          m(j,i) = p*k(i)
          asn(j,i) = (lnb*oc(j,i)+lna*(1-oc(j,i)))/(k(i)*log(vnb2)
     +k(i)*p*log(vnb1))
        end if
30    continue
      oc(15,i) = lna/(lna-lnb)
      revoc(15,i) = 1-oc(13,i)
      if (distr.eq.1.or.distr.eq.4.or.distr.eq.5) then
        m(15,i) = -log(q1/q0)/bvar1
        asn(15,i) = -lna*lnb/(log(m1(i)/m0(i))*log(q0/q1))
      else if (distr.eq.2) then
        m(15,i) = (m1(i) - m0(i))/log(m1(i)/m0(i))
        asn(15,i) = -(lna*lnb)/(m(j,i)*(log(m1(i)/m0(i)))**2)
```

```
      else
        p = log(qnb1/qnb0)/log(vnb1)
        q = 1+p
        m(15,i) = p*k(i)
        asn(15,i) = -(lna*lnb)/(k(i)*p*q*(log(vnb1))**2)
      end if
      do 35 index = 1,14
        j = 15-index
        jj = 15+index
        oc(jj,i) = ((1/a**h(j))-1)/((1/a**h(j))-(1/b**h(j)))
        revoc(jj,i) = 1-oc(jj,i)
        if (distr.eq.1.or.distr.eq.4.or.distr.eq.5) then
          m(jj,i) = (1-(1/(q1/q0)**h(j)))/((1/(m1(i)/m0(i))**h(j)) -
     +(1/(q1/q0)**h(j)))
          asn(jj,i) = (lnb*oc(jj,i)+(1-oc(jj,i))*lna)/(m(jj,i)*bvar1
     +log(q1/q0))
        else if (distr.eq.2) then
          m(jj,i) = ((m1(i) - m0(i))*(-h(j)))/(1/(m1(i)/m0(i))**h(j)-1)
          asn(jj,i) = (lnb*oc(jj,i)+lna*(1-oc(jj,i)))/(m(jj,i)*
     +log(m1(i)/m0(i))+m0(i)-m1(i))
        else
          p = (1 - 1/vnb2**h(j))/(1/vnb1**h(j)-1)
          m(jj,i) = p*k(i)
          asn(jj,i) = (lnb*oc(jj,i)+lna*(1-oc(jj,i)))/(k(i)*log(vnb2)
     +k(i)*p*log(vnb1))
        end if
35    continue

c ***************************************************************************************
c ****************** Incorporation of p-m model variablity in OC and ASN ******************
c ***************************************************************************************

c If distr = 4 (empirical binomial) or distr = 5 (NGB binomial)
c then do the following:
c 1) Convert m(j,i) which is expressed as a proportion to
c a density (pmbin(j,i)).
c 2) Compute effect of variation in p-m model or variation in
c NGB k on OC and ASN
c 3) Set revoc(j,i) and asn(j,i) to result from binemp and binngb
c subroutines

      if (distr.eq.4) then
        do 36 j = 1,29
          pmbin(j,i) = exp((log(-log(1-m(j,i)))-c)/d)
36      continue
        call binemp(29,pmbin(1,i),d,vd,mse,rn,mlnm,weight,z,
     + oc(1,i),asn(1,i),edec,easn)
      endif
      if (distr.eq.5) then
        do 37 j = 1,101
          refm(j) = float(j-1)
          call ngbcdf(refm(j),k(i),tally,pxsum)
          pxgtt(j) = 1 - pxsum
37      continue
        do 38 j = 1,29
          call tabex(refm,pxgtt,m(j,i),101,pmbin(j,i))
38      continue
        call binngb(29,pmbin(1,i),m(1,i),tpla,tplb,mse,rn,vb,mlnm,
     + oc(1,i),asn(1,i),tally,z,weight,edec,easn)
      endif
      if (distr.eq.4.or.distr.eq.5) then
        do 39 j = 1,29
```

```
          revoc(j,i) = 1-edec(j)
          asn(j,i) = easn(j)
39        continue
        endif

c ***************************************************************************************
c ********************************** display results ************************************
c ***************************************************************************************

        write (*,40)
40      format (' OC  Mean  ASN')
        do 45 j = 1,11
          ivalues = 29
          dummy = 1-ocprint(j)
          if (distr.eq.4.or.distr.eq.5) then
            call tabex(pmbin(1,i),revoc(1,i),dummy,ivalues,result)
            if (result.le. 0.0) then
              result = 0.000
            endif
          else
            call tabex(m(1,i),revoc(1,i),dummy,ivalues,result)
          endif
          mean(j,i) = result
          if (distr.eq.4) then
            dummy = 1-exp(-exp(c+d*log(result)))
          elseif (distr.eq.5) then
            call ngbcdf(result,k(i),tally,pxsum)
            dummy = 1-pxsum
          else
            dummy = result
          endif
          call tabex(asn(1,i),m(1,i),dummy,ivalues,result)
          asnprt(j,i) = result
          write (*,50) ocprint(j),mean(j,i),asnprt(j,i)
50        format (1x,f4.2,1x,f6.2,1x,f7.2)
45      continue

60      write (*,*) 'Redo OC and ASN? (Y or N)'
        read (*,'(a)') respond
        if (respond.ne.'N'.and.respond.ne.'n') then
          if (respond.ne.'Y'.and.respond.ne.'y') goto 60
        end if
        if (respond .eq. 'Y' .or. respond .eq. 'y') goto 9
61      write (*,*) 'Write OC and ASN functions to file? (Y or N)'
        read (*,'(a)') respond
        if (respond.ne.'N'.and.respond.ne.'n') then
          if (respond.ne.'Y'.and.respond.ne.'y') goto 61
        end if
        if (respond .eq. 'Y' .or. respond .eq. 'y') then
          write (10,62) i
62        format (' Complete OC and ASN functions for plan ',i2)
          write (10,63)
63        format (' H0 mean  H1 mean  k  alpha  beta')
          write (10,64) m0(i),m1(i),k(i),alpha(i),beta(i)
64        format(f7.3,3x,f7.3,3x,f5.3,3x,f4.3,4x,f4.3)
          write (10,65)
65        format (' Low intercept High intercept Slope')
          write (10,66) lowi(i), highi(i), slope(i)
66        format (6x,f8.2,9x,f7.2,1x,f6.3)
          if (distr.eq.4.or.distr.eq.5) then
            write (10,67)
```

```
67          format (' threshold proportion')
            write (10,68) mt(i),pi(i)
68          format (2(3x,f6.3,7x,f6.4,3x))
            if (distr.eq.5) then
              write (10,69) tally
69            format(' tally threshold  =  ',f5.1)
            end if
          end if
          write (10,70) t,t
70        format (' Mean',a1,'OC',a1,'ASN')
          if (distr.eq.4.or.distr.eq.5) then
            do 71 j  =  1,29
              write (10,72) pmbin(j,i),t,edec(j),t,asn(j,i)
72            format (1x,f6.2,a1,f4.2,a1,f7.2)
71          continue
          else
            do 73 j  =  1,29
              write (10,74) m(j,i),t,oc(j,i),t,asn(j,i)
74            format (1x,f6.2,a1,f4.2,a1,f7.2)
73          continue
          end if
        end if

c NGB model simulation

        if (distr.eq.3) then
75        write (*,*) 'Simulate OC and ASN? (Y or N)'
          read (*,'(a)') respond
          if (respond.ne.'N'.and.respond.ne.'n') then
            if (respond.ne.'Y'.and.respond.ne.'y') goto 75
          end if
          if (respond .eq. 'Y' .or. respond .eq. 'y') then
76          format (' Simulated OC and ASN functions for plan ',i2)
            write (10,77)
77          format (' H0 mean   H1 mean   k   alpha   beta')
            write (10,78) m0(i),m1(i),k(i),alpha(i),beta(i)
78          format(f7.3,3x,f7.3,3x,f5.3,3x,f4.3,4x,f4.3)
            write (10,79)
79          format (' Low intercept   High intercept   Slope')
            write (10,80) lowi(i), highi(i), slope(i)
80          format (6x,f8.2,9x,f7.2,1x,f6.3)
            call ngbsim (m(1,i), thresh(i),tpla,tplb,k(i),ngbk,lowi,highi,
     + slope,oc(1,i),asn(1,i),29)
          end if
        end if

90      write (*,*) 'Stop now? (Y or N)'
        read (*,'(a)') respond
        if (respond.ne.'N'.and.respond.ne.'n') then
          if (respond.ne.'Y'.and.respond.ne.'y') goto 90
        end if
        if (respond .eq. 'Y' .or. respond .eq. 'y') goto 1000

        if (init(1).eq.0.or.init(2).eq.0) goto 7

c ***************************************************************************
c ************************ display results for both plans and verify ************************
c ***************************************************************************

        call print (unit,m0,m1,k,alpha,beta,lowi,highi,slope,
       +ocprint,mean,asnprt,mt,pi,distr,tally)
```

```
      if (unit .eq. initun) then
100     write (*,*) 'Redo a plan? (Y or N)?'
        read (*,'(a)') respond
        if (respond.ne.'N'.and.respond.ne.'n') then
          if (respond.ne.'Y'.and.respond.ne.'y') goto 100
        end if
        if (respond .eq. 'Y' .or. respond .eq. 'y') then
          goto 7
        else
          unit = 10
          call print (unit,m0,m1,k,alpha,beta,lowi,highi,slope,
     + ocprint,mean,asnprt,mt,pi,distr,tally)
        end if
      end if

c ***************************************************************************************
c ************************ Simulation of tripartite plan performance ************************
c ***************************************************************************************

      write (*,*) 'starting simulations...'
      write (*,*) 'Maximum sample size? '
      read (*,*) nmax
      write (*,*) 'Monte Carlo iterations?'
      read (*,*) mci
150   format (' Max. sample size = ',i4,' iterations = ',i4)
      icount = 0
      do 200 j = 1,18
        icount = icount + 1
        simmean(icount) = m(j,1)
200   continue
      do 120 j = 19,29
        if (m(j,1).gt.m(1,2)) then
          temp = m(j-1,1)
          goto 220
        endif
        icount = icount + 1
        simmean(icount) = m(j,1)
210   continue
220   do 230 j = 1,29
        if (temp.lt.m(j,2)) then
          icount = icount + 1
          simmean(icount) = m(j,2)
        endif
230   continue
      write (*,240) icount
      write (10,240) icount
240   format (' number of means = ',i3)

      do 300 j = 1,icount
      write (*,250) j,icount
250   format (' simulating from mean ',i2,' of ',i2)

c set decision counters to zero

          seq1 = 0.
          seq3 = 0.
          seq2 = 0.
          ter1 = 0.
          ter3 = 0.
          ter2 = 0.
         totcnt = 0.
```

```
      if (distr.eq.3) then
        if (ngbk.eq.1) then
          if (simmean(j).le.((m1(1)+m0(2))/2.)) then
            ksim = k(1)
          else
            ksim = k(2)
          end if
        else
          ksim = simmean(j)**2/(tpla*simmean(j)**tplb - simmean(j))
        end if
      endif

c Monte Carlo loop

      do 400 i2 = 1,mci
        count = 0.
        sum = 0.
        if (distr.eq.1.or.distr.eq.4.or.distr.eq.5) then
          call binomial(simmean(j),x)
        else if (distr.eq.2) then
          call poisson(simmean(j),x)
        else
          call ngb(simmean(j),ksim,x)
        end if
        sum = sum + x
        count = count + 1.

c compute stop lines and compare samples to them

450   continue
      slu(1) = highi(1) + slope(1)*count
      sll(1) = lowi(1) + slope(1)*count
      slu(2) = highi(2) + slope(2)*count
      sll(2) = lowi(2) + slope(2)*count
      if ((sum.gt.slu(2).and.sum.gt.slu(1)).or.(sum.lt.sll(1).and.
     + sum.lt.sll(2))) then
        if (sum.lt.sll(1).and.sum.lt.sll(2)) then
          totcnt = totcnt + count
          seq1 = seq1 + 1.
        else
          totcnt = totcnt + count
          seq3 = seq3 + 1.
        end if
      else if (sum.gt.slu(1) .and. sum.lt.sll(2)) then
        seq2 = seq2 + 1.
        totcnt = totcnt + count
      else if (count .ge. nmax) then
    xbar = sum/float(count)
        if (xbar.le.thresh(1)) then
          ter1 = ter1 + 1.
          totcnt = totcnt + count
        else if (xbar.gt.thresh(1).and.xbar.le.thresh(2)) then
          ter2 = ter2 + 1.
          totcnt = totcnt + count
        else
          ter3 = ter3 + 1.
        totcnt = totcnt + count
        end if
        else
        if (distr.eq.1.or.distr.eq.4.or.distr.eq.5) then
          call binomial(simmean(j),x)
```

```
          else if (distr.eq.2) then
            call poisson(simmean(j),x)
          else
            call ngb(simmean(j),ksim,x)
          end if
          sum = sum + x
          count = count + 1.
          goto 450
        end if
400     continue

c compute OC values and probabilities of decision

        seq1dec = seq1/float(mci)
        ter1dec = ter1/float(mci)
        seq2dec = seq2/float(mci)
        ter2dec = ter2/float(mci)
        pdec1(j) = seq1dec + ter1dec
        pdec2(j) = seq2dec + ter2dec
        pdec3(j) = 1 - (pdec1(j) + pdec2(j))

c compute average sample size

        simasn(j) = totcnt/float(mci)
300     continue

c ***************************************************************************************
c ***************** Incorporation of p-m model variability in P{dec} and ASN *****************
c ***************************************************************************************

c If distr = 4 (empirical binomial) or distr = 5 (NGB binomial)
c then do the following:
c 1) convert simmean(j), currently a proportion, to a density
c [spmbin(j)]
c 2) compute effect of variation in p-m model on pdec and asn

        if (distr.eq.4) then
          do 460 j = 1,icount
            spmbin(j) = exp((log(-log(1-simmean(j))) - c)/d)
460       continue
          call binemp(icount,spmbin,d,vd,mse,rn,mlnm,weight,z,pdec1,
     +    simasn,temp1,easn)
          call binemp(icount,spmbin,d,vd,mse,rn,mlnm,weight,z,pdec2,
     +    simasn,temp2,easn)
          call binemp(icount,spmbin,d,vd,mse,rn,mlnm,weight,z,pdec3,
     +    simasn,temp3,easn)
endif
        if (distr.eq.5) then
          do 463 j = 1,icount
            call tabex(refm,pxgtt,simmean(j),icount,spmbin(j))
463       continue
          call binngb(icount,spmbin,simmean,tpla,tplb,mse,rn,vb,mlnm,
     +    pdec1,simasn,tally,z,weight,temp1,easn)
          call binngb(icount,spmbin,simmean,tpla,tplb,mse,rn,vb,mlnm,
     +    pdec2,simasn,tally,z,weight,temp2,easn)
          call binngb(icount,spmbin,simmean,tpla,tplb,mse,rn,vb,mlnm,
     +    pdec3,simasn,tally,z,weight,temp3,easn)
        endif
```

```
      if (distr.eq.4.or.distr.eq.5) then
        do 465 j = 1,icount
          pdec1(j) = temp1(j)
          pdec2(j) = temp2(j)
          pdec3(j) = temp3(j)
          simasn(j) = easn(j)
          simmean(j - = spmbin(j)
465     continue
      endif

c adjust P{dec} functions so that they sum to 1.0

      do 466 j = 1,count
        sumpdec = pdec1(j)+pdec2(j)+pdec3(j)
466   continue
      do 467 j = 1,count
        pdec1(j) = pdec1(j)/sumpdec
        pdec2(j) = pdec2(j)/sumpdec
        pdec3(j) = pdec3(j)/sumpdec
467   continue

      write(*,470)
470   format(' Mean P{dec1} P{dec2} P{dec3} ASN')
      write(10,471) t,t,t,t
471   format(' Mean',a1,'P{dec1}',a1,'P{dec2}',a1,'P{dec3}',a1,
     +'ASN')
      do 480 j = 1,icount
      write(*,481) simmean(j),pdec1(j),pdec2(j),pdec3(j),simasn(j)
481   format(1x,f7.3,3x,f7.3,3x,f7.3,3x,f7.3,3x,f7.2)
      write(10,482) simmean(j),t,pdec1(j),t,pdec2(j),t,pdec3(j),t,
     +simasn(j)
482   format(1x,f7.3,a1,f7.3,a1,f7.3,a1,f7.3,a1,f7.2)
480   continue
      pause

1000  continue
      close (10)
      end

      subroutine tabex(val,arg,dummy,k,value)

c ***************************************************************************************
c This subroutine uses linear interpolation to compute y = f(x).
c Sets of y values (val) and x arguments (arg) are passed along
c with a dummy x value (dummy) for which y (value) is to be
c determined. k is the size of the val and arg arrays. Values
c in the argument (arg) array must be in ascending order.
c ***************************************************************************************
      real val,arg,dummy,value
      integer k,j
      dimension val(k),arg(k)
      do 1 j = 2,k
      if (dummy.gt.arg(j)) go to 1
2     value = (dummy-arg(j-1))*(val(j)-val(j-1))/
      +(arg(j)-arg(j-1))+val(j-1)
      return
1     continue
      j = k
      go go 2
      end

      subroutine ngb(mean,k,x)
```

```
c ************************************************************************************
c This subroutine generates negative binomial distributed random
c variables using a rejection method. It requires uniform {0,1}
c random variates be generated as the variable r. If the Fortran
c compiler does not support a 'random' function, one must be provided.
c ************************************************************************************
      real x,mean,k,px,ppx,pxsum,r
      call random(r)
      x = 0.0
      px = 1/((1+mean/k)**k)
      ppx = px
      pxsum = px
      if (r .le. px) goto 4
5     x = x + 1
      px = ((k+x -1.)/x)*(mean/(mean+k))*ppx
      ppx = px
      pxsum = pxsum+px
      if (r .le. pxsum) goto 4
      if (x. gt. 200.) goto 4
      goto 5
4     continue
      return
      end

      subroutine binomial(p,x)

c ************************************************************************************
c This subroutine generates binomial distributed random
c variables x {x = 0 or 1} using a rejection method.
c The subroutine requires uniform {0,1} random variates be generated
c as the variable r. If the Fortran compiler does not support a
c 'random' function, one must be provided.
c ************************************************************************************
      real p,x,r
      call random(r)
      if (r.lt.p) then
        x = 1.
      else
        x = 0.0
      end if
      return
      end

      subroutine poisson(mean,poivar)

c ************************************************************************************
c Generates pseudo random Poisson distributed random variables using a
c rejection method. A uniform random variate (r) is required. If
c the FORTRAN compiler does not support a function (random in this routine)
c for generating uniform deviates, one must be supplied.
c ************************************************************************************
      real mean,poivar,r,px,pxsum

      call random(r)
      poivar = 0.
      px = exp(-mean)
      pxsum = px
      if (r.le.pxdum) return

10    continue
        poivar = poivar + 1.
        px = px*mean/poivar
        pxsum = pxsum+px
```

```
          if (r.le.pxsum) return
          if (poivar.ge.500) return
        goto 10

        return
        end

        subroutine binemp (entries,m,d,vd,mse,rn,mlnm,weight,z,novdec,
     +novasn,edec,easn)

c ***************************************************************************
c Subroutine computes expected P{dec} and ASN for sequential binomial sampling
c programs based on the empirical model ln(-ln(pT)) = a + bln(m) where pT is
c the proportion of samples with less than T animals and m is the mean. A
c maximum of 58 nominal values can be passed.
c ***************************************************************************

c variable definitions

c c,d; intercept and slope of p0-m relationship
c dec(k,j), asn(k,j); P{dec} and asn values indexed by the means (m(j))
c       and the bias weights k
c edec(j), easn(j); expected P{dec} and asn functions
c m(j) mean density corresponding to 1-p(j) where p(j) is the proportion of
c       samples with ≤ tally threshold
c mlnm; mean of independent variable of p-m relationship
c mse; mean square error of p0-m relationship
c negbm(j) means biased by P{x≤tally threshold} being less than nominal value
c novdec(j), novasn(j): P{dec} and asn values exclusive of variance in the p-m
c       relationship
c posbdec(j),negbdec(j),posbasn(j),negbasn(j): oc and asn values for biased p
c       values calculated for the set of means = m(j)
c posbm(j) means biased by P{x≤tally threshold} being greater than nominal value
c vd, rn; variance of d and data points used in p-m relationship
c vlnlnp(j); variance of ln(-ln(1-pT)
c weight(k) set of values for weighting biased P{dec} and ASN curves
c z(k): standard normal values associated with weight(k)
c declarations

        integer entries,j,k

        real d,vd,mse,rn,mlnm,novdec,novasn,m,weight(11),z(11),
     +vlnlnp(58),temp,posbm(58),negbm(58),posbdec(58),negbdec(58),
     +posbasn(58),negbasn(58),bdec(11,58),basn(11,58),edec,easn,
     +dummy,result

        dimension novdec(entries),novasn(entries),m(entries),
     +edec(entries),easn(entries)

c computations

        do 30 j = 1,entries
            vlnlnp(j) = mse/rn+(log(m(j))-mlnm)**2*vd+mse
            bdec(1,j) = novdec(j)
            basn(1,j) = novasn(j)
30        continue
        do 40 k = 2,11
            do 50 j = 1,entries
                temp = exp(z(k)*sqrt(vlnlnp(j))/d)
                posbm(j) = m(j)/temp
                negbm(j) = m(j)*temp
50            continue
```

```
c compute P{dec} and ASN values for biased means that have the same p value
c as the nominal means

            do 60 j  =  1,entries
                dummy  =  m(j)
                call tabex(novdec,posbm,dummy,entries,result)
                posbdec(j)  =  result
                call tabex(novdec,negbm,dummy,entries,result)
                negbdec(j)  =  result
                call tabex(novasn,posbm,dummy,entries,result)
                posbasn(j)  =  result
                call tabex(novasn,negbm,dummy,entries,result)
                negbasn(j)  =  result
                bdec(k,j)  =  (posbdec(j) + negbdec(j))/2.
                basn(k,j)  =  (posbasn(j) + negbasn(j))/2.
60            continue
40        continue

c compute expected P{dec} and ASN values

        do 70 j  =  1,entries
            edec(j)  =  0.0
            easn(j)  =  0.0
            do 80 k  =  1,11
              edec(j)  =  edec(j) + bdec(k,j) * weight(k)
              easn(j)  =  easn(j) + basn(k,j) * weight(k)
80          continue
70      continue
        return
        end
        subroutine binngb(entries,m,p,a,b,mse,rn,vb,mlnm,novdec,
     +novasn,tally,z,weight,edec,easn)

c ***************************************************************************************
c Subroutine computes expected P{dec} and ASN for sequential binomial sampling
c programs on a negative binomial model. Expected values are based on
c variation in k modeled using variation about TPL ( residuals about TPL are
c assumed to be normally distributed. A maximum of 58 nominal values can be
c passed.
c ***************************************************************************************

c variable definitions

c a   TPL parameter
c b   TPL parameter
c dummy   temporary variable
c easn   expected ASN
c edec   expected P{dec}
c entries   number of P{dec} and ASN values passed
c i   integer index
c j   integer index
c ktemp   temporary variable
c lnm   ln(mean)
c lnv   ln(variance)
c m   mean
c mlnm   mean of ln(mean) used in TPL
c mse   means square error for TPL regression
c negasn   asn corresponding to negative deviation about TPL
c negdec   P{dec} corresponding to negative deviation about TPL
c novasn   nominal ASN
c novdec   nominal P{dec}
c p   proportion of samples with more than T organisms (corresponds to m)
c posasn   asn corresponding to positive deviation about TPL
```

```
c posdec  P{dec} corresponding to positive deviation about TPL
c pxsum  NGB cumulative distribution function value
c rn  data points in TPL regression
c tally  T value or p−m model
c v  variance
c vb  variance of TPL b
c vlnv  variance of ln(variance predicted via TPL)
c weight  normal probability for z
c z  standard normal deviate

      integer entries,j,i
      real m,p,a,b,mse,rn,vb,mlnm,novdec,novasn,tally,z,weight,edec,
     +easn,v,lnv,lnm,vlnv,dummy,ktemp,pxsum,posdec,negdec,
     +posasn,negasn
      dimension m(entries),p (entries),novdec (entries),novasn(entries),
     +z(11),weight(11),edec (entries),easn(entries)

c computations

      do 1 j = 1,entries
        v = a*m(j)**b
        lnv = log(v)
        lnm = log(m(j))
        vlnv = mse/rn + (lnm − mlnm)**2*vb + mse
        edec(j) = 0.0
        easn(j) = 0.0
          do 35 i = 1,11
          dummy = lnv + z(i)*sqrt(vlnv)
          dummy = exp(dummy)
          ktemp = (dummy − m(j))/m(j)**2
          if (ktemp.lt.0.01) ktemp = 0.01
          ktemp = 1./ktemp
          call ngbcdf (m(j),ktemp,tally,pxsum)
          dummy = 1.0 − pxsum
          call tabex(novdec,p,dummy,entries,posdec)
          if (posdec.gt.1.) posdec = 1.0
          if (posdec.lt.0.) posdec = 0.
          call tabex(novasn,p,dummy,entries,posasn)
          if (posasn.lt.1.0) posasn = 1.0

c repeat for negative deviation

          dummy = lnv − z(i)*sqrt(vlnv)
          dummy = exp(dummy)
          ktemp = (dummy − m(j))/m(j)**2
          if (ktemp.lt.0.01) ktemp = 0.01
          ktemp = 1./ktemp
          call ngbcdf (m(j),ktemp,tally,pxsum)
          dummy = 1.0 − pxsum
          call tabex(novdec,p,dummy,entries,negdec)
          if (negdec.gt.1.) negdec = 1.0
          if (negdec.lt.0.) negdec = 0.
          call tabex(novasn,p,dummy,entries,negasn)
          if (negasn.lt.1.0) negasn = 1.0
          edec(j) = edec(j) + ((posdec + negdec)/2)*weight(i)
          easn(j) = easn(j) + ((posasn + negasn)/2)*weight(i)
35    continue
1     continue
      return
      end

      subroutine ngbcdf (m,k,t,pxsum)
```

```
c ************************************************************************************
c This routine computes the probability that x is ≤ T when x is
c distributed as a negative binomial random variable with m
c (the mean) and k (the dispersion parameter).
c The routine uses a recursive relationship for computing the
c probability mass function for the negative binomial distribution.
c ************************************************************************************

      real m,p,q,k,px,pxp,pxsum,x,t
10    continue
      p = m/k
      q = 1+p
      x = 0.
      pxsum = 0.
      px = 1/q**k
      pxsum = pxsum + px
      pxp = px
15    continue
      if (x.lt.t) then
        x = x+1
        px = ((k+x-1)*p/(x*q))*pxp
        pxp = px
        pxsum = pxsum+px
        goto 15
      end if
      return
      end

      subroutine print (unit,m0,m1,k,alpha,beta,lowi,highi,slope,
     +ocprint,mean,asnprt,mt,pi,distr,tally)
      real m0(2),m1(2),k(2),alpha(2),beta(2),lowi(2),highi(2),slope(2),
     +ocprint(11),mean(11,2),asnprt(11,2),mt(2),pi(2),tally
      integer unit,j,distr
      write (unit,1)
1     format ('H0 mean   H1 mean   k   alpha   beta',
     +' H0 mean   H1 mean   k   alpha   beta')
      write (unit,2) m0(1),m1(1),k(1),alpha(1),beta(1),
     +m0(2),m1(2),k(2),alpha(2),beta(2)
2     format(2(f7.3,3x,f7.3,3x,f5.3,3x,f4.3,4x,f4.3))
      write (unit,3)
3     format (' Low intercept   High intercept   Slope',
     +' Low intercept   High intercept   Slope')
      write (unit,4) lowi(1), highi(1), slope(1),
     +lowi(2), highi(2), slope(2)
4     format (2(6x,f8.2,9x,f7.2,1x,f6.3))
      write (unit,5)
5     format (' OC   Mean   ASN',1x,' OC   Mean   ASN')
      do 10 j = 1,11
        write (unit,11) ocprint(j),mean(j,1),asnprt(j,1),
     +  ocprint(j),mean(j,2),asnprt(j,2)
11      format (2(1x,f4.2,1x,f6.2,1x,f7.2))
10    continue
      if (distr.eq.4.or.distr.eq.5) then
        write (unit,15)
15      format (' threshold proportion   threshold proportion')
        write (unit,16) mt(1),pi(1),mt(2),pi(2)
16      format (2(3x,f6.3,7x,f6.4,3x))
        if (distr.eq.5) then
          write (unit,17) tally
17        format(' tally threshold = ',f5.1)
        endif
```

```
      endif
      return
      end

      subroutine ngbsim (m,thresh,tpla,tplb,nomk,ngbk,lowi,highi,
     +slope,nomoc,nomasn,values)

c ***************************************************************************************
c This subroutine simulates the performance of an SPRT when a negative
c binomial model is used and k is either 1) a constant, or 2) a function
c of TPL.
c ***************************************************************************************
      real m,thresh,tpla,tplb,lowi,highi,slope,nomoc,nomasn,ngbsoc,
     +ngbsasn,seql,seqh,terl,terh,totcnt,k,sum,slu,sll,xbar,var,x,nomk
      integer values,i,j,nmax,mci,count,ngbk
      character*1 respond,tab
      dimension m(values),nomoc(values),nomasn(values)
      tab = char(9)

c variables unique to subroutine:

c count, sum; number of samples, sum of animals in total samples from one run
c k; k from negative binomial distribution used in sensitivity analysis
c mci; number of Monte Carlo iterations
c mci; number of monte carlo iterations
c ngbsoc, ngbsasn; oc and asn values determined via simulation
c nmax; maximum number of samples to be taken
c nmax; maximum number of samples to be taken
c seql, seqh, terl, terh; number of above and below threshold
c sequential decisions, number of above and below terminal decisions
c slu, sll; upper and lower stop lines
c totcnt; total number of samples drawn
c tpla, tplb; parameters for Taylors power law
c xbar; mean of sample observations
c
c variables passed to subroutine

c lowi, highi, slope; intercepts and slopes for stop lines
c m; means for which oc and asn are to be computed
c ngbk; indicates whether 1) constant k or 2) k = f(TPL)
c nomk; nominal value of k used to construct stop lines
c nomoc, nomasn; nominal oc and asn values
c thresh; average of means for null and alternate hypothesis
c pla, tplb; a and b for Taylor's power law

c obtain necessary data

1     write (*,*) 'input maximum sample size'
      read (*,*) nmax
      write (*,*) 'input monte carlo iterations'
      read (*,*) mci
      k = 0.0
      if (ngbk.eq.1) then
        write (*,5) nomk
5       format (' Nominal k = ',f5.3,/,
     +  ' input k value for simulations')
        read (*,*) k
      end if
      write (*,10) nmax, mci,k
10    format (' maximum samples = ',i4,' monte carlo iterations = ',i5
     +' k* = ',f5.3,/,' '*if k = 0, k = f(TPL)')
```

```
11      write (*,*) 'parameters correct? (y or n)'
        read (*,'(a)') respond
          if (respond.ne.'N'and.respond.ne.'n') then
            if (respond.ne.'Y'.and.respond.ne.'y') goto 11
  end if
        if (respond.eq. 'N' .or. respond.eq. 'n') goto 1
        write (10,*) 'Simulation of SPRT performance with variable k'
        write (10,12) nmax,mci, k
12      format (' maximum samples = ',i4,' monte carlo iterations = ',i5,
     +' k* = ',f5.3,/, *if k = 0, k = f(TPL)')

        write (*,20)
20        format (' Mean  Nom_OC  Sim_OC  Nom_ASN  Sim_ASN')
        write (10,21) tab,tab,tab,tab
21      format (' Mean',a1,'Nom_OC',a1,'Sim_OC',a1,'Nom_ASN',a1,
     +'Sim_ASN')

        do 100 i = 1,values

c set decision counters to zero and compute k

          seql = 0.
          seqh = 0.
          terl = 0.
          terh = 0.
          totcnt = 0.
          if (ngbk.eq.2) then
            var = tpla*m(i)**tplb
            k = (m(i)**2)/(var – m(i))
          end if

c Monte Carlo loop

          do 110,j = 1,mci
            count = 0.
            sum = 0.
       call ngb(m(i),k,x)
            sum = sum + x
            count = count + 1

c compute stop lines and compare samples to them

120         continue
            slu = highi + slope*float(count)
            sll = lowi + slope*float(count)
            if (sum.gt.slu.or.sum.lt.sll) then
              if (sum.lt.sll) then
                ttcnt = totcnt + count
                seql = seql + 1.
              else
                totcnt = totcnt + count
                seqh = seqh + 1.
              end if
            else if (count .ge. nmax) then
              xbar = sum/float(count)
              if (xbar.lt.thresh) then
                terl = terl + 1.
                totcnt = totcnt + count
              else
                terh = terh + 1.
                totcnt = totcnt + count
              end if
```

```
            else
              call ngb(m(i),k,x)
              sum  =  sum + x
              count  =  count + 1
              goto 120
            end if
110     continue

c compute OC values
        ngbsoc  =  (seql + terl)/float(mci)

c compute average sample size

        ngbsasn  =  totcnt/float(mci)

c write results

        write (*,130) m(i),nomoc(i),ngbsoc,nomasn(i),ngbsasn
130     format (1x,f6.2,2x,f5.3,3x,f5.3,3x,f7.2,2x,f7.2)
        write (10,131) m(i),tab,nomoc(i),tab,ngbsoc,tab,nomasn(i),tab,
     +   ngbsasn
131     format (1x,f6.2,a1,f5.3,a1,f5.3,a1,f7.2,a1,f7.2)

100     continue

        if (ngbk.eq.1) then
          write (*,*) 'repeat simulations for new k (y or n)?'
          read (*,'(a)') respond
200     if (respond.ne.'N'.and.respond.ne.'n') then
            if (respond.ne.'Y'.and.respond.ne.'y') goto 200
          end if
          if (respond.eq. 'Y' .or. respond.eq. 'y') goto 1
        end if

        return
        end
```

cascade: cascading of sampling decisions to monitor density through time

Background: This program evaluates the performance of sampling plans cascaded through time. Each sample plan independently results in decisions to either resample the population i time intervals in the future or to intervene. A maximum of 20 sampling plans can be cascaded. The program works by first building a table of sampling statistics indexed by sampling time and then by computing summaries of these tables. Extended output yields the time-indexed tables. Two input data files are required; one provides the sampling statistics (P{dec} and ASN) for each sampling plan used and time schedule for using each plan, and the other data file describes the population trajectories to be sampled. At each sampling bout (time) a particular sampling plan is used that can result in one of ndel + 1 decisions; intervene, or resample after $i = 1, \ldots,$ ndel time periods. A maximum of five resample time intervals can be used. Probabilities for these decisions and the average sample number are determined via interpolation from an input data file. Each sample plan can have a maximum of 100 data points that describe the probabilities for these decisions [pdec(i, j, k); $i = 1, \ldots, 100$, $j = 1, \ldots, 5$ (plan number), $k = 1, \ldots, 6$ (decision)]. A table is built for each possible sampling bout that consists of the time [determined by the beginning sampling time (begtime) and sample interval (deltime)],

the population density (dent) which is interpolated from an input file, the probability of sampling at that time (psamp), the six possible decision probabilities, the cumulative probability of making a decision to intervene (sumptrt) and average sample number, and the loss (losst) which is defined as the cumulative population density.

Summary statistics [indexed by the population (pindex)] consist of the cumulative density for the population, the OC (oc), the average total sample size (avgtotn), the expected number of sampling bouts (expbout), the expected loss (exploss), and losses with probability 0.5, 0.2, and 0.05. At the end of the summary table is the statement "if P() loss = Cum.den, SUM P(dec3) < 1 − P()." This means that if the loss with probability x equals the cumulative density, then the cumulative probability of a decision to intervene is less than x.

The program functions as follows:

1. Files that contain the P{dec} and ASN data and the population trajectories to be sampled are specified along with a file to which output is written. Data are read from the respective files which have the following formats. P{dec} and ASN data file. The first line contains two integers separated by a tab, space, or comma. The first number specifies the number of sampling plans and the second the number of time intervals (≤ 5) sampling might be delayed. The second line contains two integers delimited by a tab, space, or comma that specify the number of data points (n) for the first sampling plan and the time when the sampling plan should first be used. The third through $n + 2$ lines contain the mean, P{dec(1)}, P{dec(2)},..., P{dec(ndel + 1)}, and ASN values for the first sampling plan delimited by a tab, space, or comma. The $n +$ third line contains the number of data points for the second sampling plan and when the second plan should first be used. The data file then repeats to conclusion. An example is shown below:

```
3       2
46      0
0.245   1.000   0.000   0.000   21.59
0.276   1.000   0.000   0.000   22.82
0.312   1.000   0.000   0.000   23.89
0.353   1.000   0.000   0.000   25.39
0.402   0.998   0.002   0.000   28.34
0.459   0.994   0.006   0.000   30.21
0.526   0.994   0.006   0.000   34.66
0.606   0.980   0.020   0.000   40.71
0.700   0.936   0.064   0.000   50.40
0.812   0.862   0.138   0.000   58.41
0.837   0.772   0.228   0.000   56.41
0.863   0.720   0.280   0.000   58.13
0.890   0.676   0.324   0.000   56.68
...
49      30
0.789   1.000   0.000   0.000   21.93
0.856   1.000   0.000   0.000   23.54
...
```

Population trajectory data file: the first line contains the number of populations in the file (maximum of 15). Subsequent lines have the time and densities for each population delimited by a tab, space, or comma. An example with seven populations and time values of 12, 20, . . . , 89 is shown below:

```
7
12    0      0      0      0      0.1    3.88   0.34
20    0      0      0.12   0.42   0.36   1.36   0.2
26    0      0.02   0.2    0.52   0.42   0.58   0.16
33    0      0.1    0.12   0.64   0.16   5.22   0.09
40    0.22   0.02   0.82   1.58   0.16   1.9    0.04
47    0.32   0.06   2.58   4.58   0.3    1.1    0.08
61    9.05   0.38   0.14   16.53  0.2    3.7    0.38
68    2.6    1.44   3.16   20.74  0.01   2.58   3.04
75    8.98   12.22  2.72   8.4    0      8.9    4.68
82    11.82  17.68  1.94   2.02   0.18   12.7   0.8
89    18.84  39.46  3.38   0.16   0.28   18.6   0.1
```

The file containing the P{dec} and ASN data can be quickly constructed from the output file generated by programs 'sprt' or 'afcm.' It is important that the P{dec} functions in these files be convex at the tails; otherwise, interpolated values may be seriously in error. For example, if the last three values for the P{dec(3)} function were 1.0, 1.0, 0.99 when ndel = 2; these should be changed to 1.0, 1.0, 1.0.

2. Whether or not extended output is desired is specified.
3. When samples are to be taken is specified by providing a starting time, an ending time, and a time interval between samples. If the time interval lies outside the time horizon for which population trajectory data are available, densities are truncated to either the first or last value. Note that while sampling plans constructed for cascading usually are designed with a specific time interval between samples in mind, this feature allows other time intervals to be used for sensitivity analysis.
4. A population scale factor can be applied if desired. This factor scales the densities in the population trajectory file by the multiple provided. A value of 1.0 must be entered if no scaling is desired.
5. Computations are made and results are written to the output file. If desired computations can be repeated for a new population scale factor or for a new sampling time schedule.

An example is shown below. Ulser inputs are underlined and comments are in italics.

```
> cascade
Enter name of file with P{dec} & ASN data
ngb.in
Enter name of file with trajectory data
histpop.in
P{dec} & ASN file  =  ngb.in Population data file  =  histpop.in
Is this correct? (Y or N)
y
```

Enter name of file for output
ngb.out
Select extended(1) or summary(2) output
1
Write P{dec} & ASN data to file? (Y or N)
n
Starting time for plan 1 = 0
Starting time for population trajectories = 12
Ending time for population trajectories = 89
Input time for first sample:
12
Input time interval between samples:
7
Input time for last sample:
89
Sampling schedule:

Start	Interval	Last sample at
12	7	89

Is this correct? (Y or N)
y
Input population scale factor: 0.0
1.0
computations are performed and displayed on the screen
Repeat for new population scale? (Y or N)
n
Repeat for new time model? (Y or N)
n
a portion of the output file is shown below:
P{dec} & ASN file = ngb.in Population data file = histpop.in
Results for population # 1

Time	Density
12.00	0.00
20.00	0.00
26.00	0.00
33.00	0.00
40.00	0.22
47.00	0.32
61.00	9.05
68.00	2.60
75.00	8.98
82.00	11.82
89.00	18.84

Time	Density	P{samp}	Pd	Pd2	Pd3	Pd4	Pd5	Pd6	p{trt}	ASN	Loss
12.00	0.00	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	21.59	0.00
19.00	0.00	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	21.59	0.00
26.00	0.00	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	21.59	0.00
33.00	0.00	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	21.93	0.00
40.00	0.22	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	21.93	0.77
47.00	0.32	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	21.93	2.66
54.00	4.68	1.000	0.000	0.655	0.345	0.000	0.000	0.000	0.345	51.48	35.46
61.00	9.05	0.655	0.000	0.069	0.931	0.000	0.000	0.000	0.955	39.45	68.25
68.00	2.60	0.045	0.800	0.200	0.000	0.000	0.000	0.000	0.955	51.74	109.03
75.00	8.98	0.009	0.000	0.084	0.916	0.000	0.000	0.000	0.963	40.36	149.56
82.00	11.82	0.037	0.000	0.002	0.998	0.000	0.000	0.000	1.000	17.19	222.36
89.00	18.84	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	9.72	329.67

Note: Pd1–Pd6 values are output even when the number of delays are < 6. The last Pdi values that are not all zeroes are the probabilities for an intervention decision.

Results for population # 2

Time	Density
12.00	0.00
20.00	0.00
26.00	0.02

...

Start	Interval	Last sample at	Possible bouts
12.0	7.0	89.0	12.0

Population scale factor = 1.000

Pop.	Cum.den.	OC	ASN	Bouts	Exp.loss	P(0.5)loss	P(0.2)loss	P(0.05)loss
1	329.67	0.00	145.79	4.75	63.35	43.79	59.94	68.01
2	363.08	0.00	118.75	6.00	163.09	110.76	142.16	157.86
3	103.85	1.00	229.89	6.64	103.85	103.85	103.85	103.85
4	462.20	0.00	119.94	4.08	109.47	73.64	96.52	107.96
5	15.45	1.00	129.46	6.00	15.45	15.45	15.45	15.45
6	363.41	0.00	25.19	1.05	1.78	0.00	0.00	0.00
7	69.53	1.00	180.16	6.48	69.49	69.53	69.53	69.53

** if P()loss = Cum.den, SUM(pdec3) < 1 − P()

```
      program cascade

c     Program written in Fortran 77: IBM version compiled using Microsoft
c     FORTRAN compiler v5.1.
c     Program written by Jan Nyrop, Department of Entomology, NYSAES,
c     Cornell University, Geneva, NY 14456
c     Last modification date: 2 February, 1993

c ***************************************************************************
c ******************************** Variable descriptions ********************
c ***************************************************************************

c asn(j,i): ASN values for mean j and sampling plan i
c avgden: average of two densities used to compute cumulative density
c avgtotn(k): average total number of samples taken for each of k
c     trajectories
c begtime: time for first sample
c bout: sample bout index
c cplan: sampling plan currently being used
c delta: time interval used to compute cumulative density
c deltime: sampling time interval
c dent: population density at current sampling time
c endtime: last sampling time
c expbout(k): expect number of sampling bouts for each of k trajectories
c exploss(k): expected loss for each of k population trajectories
c file1: file name
c file2: file name
c file3: file name
c i: integer index
c iend: integer index
c ii: integer index
```

```
c iin1: file unit number
c iin2: file unit number
c input: character to record keyboard input
c intasn: interpolated asn value
c intpd(i): interpolated P{dec(i)} value
c iout: file unit number
c ipoint: integer index
c ipops: integer index
c j: integer index
c losst: loss (cumulative density) at sampling time t
c losssb(sb): loss at sampling bout sb
c lossp1(k): interpolated loss for probability p1 and population k
c lossp2(k): interpolated loss for probability p2 and population k
c lossp3(k): interpolated loss for probability p3 and population k
c lsttime: last sampling time
c mean(j,i): mean for P{dec} and ASN functions for tripartite plan i
c mult: multiplier for scaling population density
c ndel: number of time intervals resampling might be delayed
c oc(k): OC value for each of k population trajectories
c oldpopsc: old population scale value
c pdec(j,k,i): probability of making decision i with plan k for mean j
c pindex: population index
c plans: number of sampling plans used
c points(i): number of P{dec} and ASN data points in sampling plan i
c popscale: population scale factor
c potbouts: number of potential sampling bouts
c prpdec(j,i): probability of making decision j i time intervals previous
c prpsamp(i): probability of sampling i time intervals previous
c psamp: probability of sampling at a particular time
c respond: keyboard input
c respond2: keyboard input
c sched(i): time to begin using sampling plan i
c sum1: summed values
c sum2: summed values
c sum3: summed values
c sumden(j,i): cumulative density through time for population i
c sumptrt(sb): probability of intervention summed through sample bout sb
c time: sampling time
c trcnt: number of population trajectories to be sampled
c trden(j,i): population density for trajectory i
c trtime(j): times corresponding to trden(j,i)

c ********************************************************************************
c ********************************** Declarations **********************************
c ********************************************************************************

      real mean(100,20),pdec(100,20,6),
     +asn(100,20),oc(15),expbout(15),avgtotn(15),exploss(15),sched(20),
     +trtime(100),trden(100,15),sumden(100,15),losssb(50),sumptrt(50),
     +lossp1(15),lossp2(15),lossp3(15),prpsamp(5),prpdec(5,5),intpd(6)
      real time,dent,psamp,intasn,sum1,sum2,temp,
     +sum3,deltime,endtime,losst,begtime,popscale,mult,oldpopsc,
     +lsttime,delta,avgden,potbouts

      integer plans,cplan,respond,pindex,trcnt,ipops,k,
     +points(20),iin1,iin2,iout,i,iend,j,ii,ipoint,bout,ndel

      character*12 file1, file2, file3
      character*1 t,input
```

```
      t = char(9)
      oldpopsc = 1.0
      popscale = 0.0

      do 500 k = 1,6
        do 510 j = 1,20
          do 530 i = 1,100
            pdec(i,j,k) = 0.0
520       continue
510     continue
        intpd(k) = 0.0
500   continue

c ***************************************************************************************
c ******************************** Model parameterization ********************************
c ***************************************************************************************

      iin1 = 20
      iin2 = 21
      iout = 15

1     write (*,'(a)') ' Enter name of file with P{dec} & ASN data'
      read (*,'(a)') file1
      write (*,'(a)') ' Enter name of file with trajectory data'
      read (*,'(a)') file2
      write (*,2) file1,file2
2     format (' P{dec} & ASN file = ',a12,' Population data file = ',
     +a12)
3     write (*,*) 'Is this correct? (Y or N)'
      read (*,'(a)') input
        if (input.ne.'N'.and.input.ne.'n') then
          if (input.ne.'Y'.and.input.ne.'y') goto 3
        end if
        if (input .eq. 'N' .or. input .eq. 'n') goto 1
      write (*,'(a)') ' Enter name of file for output'
      read (*,'(a)') file3
      open (unit = iin1,file = file1,status = 'old')
      open (unit = iin2,file = file2,status = 'old')
      open (unit = iout,file = file3,status = 'unknown')
      write (iout,4) file1,file2
4     format (' P{dec} & ASN file = ',a12,' Population data file = ',
     +a12)

c determine whether extended or summary output is desired

10    write (*,*) 'Select extended(1) or summary (2) output'
      read (*,*) respond
      if ((respond.ne.1).and.(respond.ne.2)) goto 10

c read P{dec} and ASN data

      read (iin1,*) plans,ndel
      do 15 i = 1,plans
        read (iin1,*) points(i),sched(i)
        iend = points(i)
        do 20 j = 1,iend
          read (iin1,*) mean(j,i),(pdec(j,i,k),k = 1,ndel + 1),asn(j,i)
20      continue
15    continue
```

```
c read population trajectory data

      read (iin2,*) ipops
      trcnt - 0
22    if (.not. eof(iin2)) then
        trcnt  =  trcnt + 1
        read (iin2,*) trtime(trcnt), (trcnt), (trden(trcnt,j), j  =  1,ipops)
        goto 22
      endif
      close (iin2)

c if extended results are requested determine whether P{dec} and ASN functions
c should be written to file:

      if (respond.eq.1) then
25      write (*,*) 'Write P{dec} & ASN data to file? (Y or N)'
        read (*,'(a)') input
        if (input.ne.'N'.and.input.ne.'n') then
          if (input.ne.'Y'.and.input.ne.'y') goto 25
        end if
        if (input .eq. 'Y' .or. input .eq. 'y') then
          do 30 i  =  1,plans
            write (iout,31) i
31          format (' P{dec} and ASN functions for sampling plan ',i2)
            write (iout,33) t,t,t,t,t,t,t
33          format (' Mean',a1,'Pd1',a1,'Pd2',a1,'Pd3',a1,'Pd4',a1,
     +              'Pd5',a1,'Pd6',a1,'ASN')
            iend  =  points(i)
            do 35 j  =  1,iend
              write (iout,36) mean(j,i),t,pdec(j,i,1),t,pdec(j,i,2),t,
     +                        pdec(j,i,3),t,pdec(j,i,4),t,
     +                        pdec(j,i,5),t,pdec(j,i,6),t,asn(j,i)
36            format (1x,f6.2,a1,6(f5.3,a1),f7.2)
35          continue
30        continue
        end if
      end if

c determine time interval between samples and last sampling time

39    continue
      write (*,40) sched(1)
40    format (' Starting time for plan 1  =  ',f6.1)
      write (*,41) trtime(1)
41    format (' Starting time for population trajectories  =  ',f6.1)
      write (*,42) trtime(trcnt)
42    format (' Ending time for population trajectories  =  ',f6.1)
      write (*,*) 'Input time for first sample:'
      read (*,*) begtime
      write (*,*) 'Input time interval between samples:'
      read (*,*) deltime
      write (*,*) 'Input time for last sample:'
      read (*,*) endtime
      potbouts  =  aint((endtime - begtime)/deltime) + 1
      lsttime  =  begtime + ((potbouts - 1)*deltime)
      write (*,*) 'Sampling schedule:'
      write (*,43)
43    format ('Start   Interval   Last sample at')
      write (*,44) begtime,deltime,lsttime
44    format (1x,f6.1,2x,f6.1,5x,f6.1)
```

```
45    write (*,*) 'Is this correct? (Y or N)'
      read (*,'(a)') input
        if (input.ne.'N'.and.input.ne.'n') then
          if (input.ne.'Y'.and.input.ne.'y') goto 45
        end if
        if (input .eq. 'N' .or. input .eq. 'n') goto 39

c apply population scale factor

46    write (*,47) popscale
47    format ('Input population scale factor: ',f7.3)
      read (*,*) popscale
      mult = popscale/oldpopsc
      oldpopsc = popscale
      do 48 i = 1,trcnt
        do 49 j = 1,ipops
          trden(i,j) = trden(i,j)*mult
49      continue
48    continue

c ***************************************************************************************
c ************************************ Calculations ************************************
c ***************************************************************************************

c compute the area beneath each population trajectory curve (sumden)

      do 50 j = 1,ipops
        sumden(1,j) = 0.0
        do 51 i = 1,trcnt - 1
          delta = trtime(i + 1) - trtime(i)
          avgden = (trden(i + 1,j) + trden(i,j))/2.0
          sumden(i + 1,j) = sumden(i,j) + (delta*avgden)
51      continue
50    continue

c loop based on populations in trajectory file
      pindex = 1
60    continue

c if detailed output is required, print header

      if (respond.eq.1) then
        write (iout,61) pindex
61      format ( 'Results for population # ',i2)
        write (iout,62) t
62      format (/,'Time',a1,'Density')
        do 63 ii = 1,trcnt
          write (iout,64) trtime(ii),t,trden(ii,pindex)
64        format(1x,f7.2,a1,f7.2)
63      continue
        write (iout,65)t,t,t,t,t,t,t,t,t,t,t,t
65      format (/'Time',a1,'Plan',a1,'Density',a1,'P{samp}',a1,'Pd1'
     +    a1,'Pd2',a1,'Pd3',a1,'Pd4',a1'Pd5',a1,'Pd6',a1,'P{trt}',a1,
     +    'ASN',a1,'Loss')
      end if

c initialize probabilities at time 0, set time to 0, set sums to 0

      do 66 i = 1,5
        do 67 j = 1,5
          prpdec(i,j) = 0.0
```

```
67      continue
        prpsamp(i) = 0.0
66    continue

      prpdec(ndel,1) = 1.0
      prpsamp(1) = 1.0
      time = begtime
      bout = 0
      sum1 = 0.
      sum2 = 0.
      sum3 = 0.
      exploss(pindex) = 0.

c loop based on time

70    continue

      bout = bout + 1

c determine which sampling plan to use

      i = plans
75    if ((time + 0.001).ge.sched(i)) then
        cplan = i
      else
        i = i-1
        go to 75
      end if

c compute the time dependent density

      call interp(trden(1,pindex),trtime,time,trcnt,1,dent)

c compute the time dependent loss function

      call interp(sumden(1,pindex),trtime,time,trcnt,1,losst)
      losssb(bout) = losst

c compute the interpolated probabilities for each decision and
c the interpolated asn

      ipoint = points(cplan)
      do 78 j = 1,ndel + 1
        call interp(pdec(1,cplan,j),mean(1,cplan),dent,ipoint,0,
     +              intpd(j))
        if (intpd(j).lt.0.0) intpd(j) = 0.0
        if (intpd(j).gt.1.0) intpd(j) = 1.0
78    continue
      call interp(asn(1,cplan),mean(1,cplan),dent,ipoint,1,intasn)

c calculate the probability of sampling in the current time period
      psamp = 0
      do 90 j = 1,ndel
        psamp = psamp + (prpsamp(j)*prpdec(ndel + 1 - j,j))
90    continue
```

```
c calculate sums

      sum1 = sum1 + (psamp*intpd(ndel + 1))
      sum2 = sum2 + (psamp*intasn)
      sum3 = sum3 + psamp
      sumptrt(bout) = sum1

c determine if it is the last sampling time

      if ((time + deltime).lt.(endtime + 0.0001)) then
        exploss(pindex) = exploss(pindex) + losst*psamp*intpd(ndel + 1)

c delay probabilities of decision 1 to ndel; prpdec(decision,delay)

        do 95 j = 1,ndel
          do 100 k = ndel,2,-1
            prpdec(j,k) = prpdec(j,k-1)
100     continue
        prpdec(j,1) = intpd(j)
95      continue

c delay probabilities of sampling; prpsamp(delay)

        do 110 j = ndel,2,-1
          prpsamp(j) = prpsamp(j - 1)
110     continue
        prpsamp(1) = psamp
        if (respond.eq.1) then
          write (iout,120) time,t,cplan,t,dent,t,psamp,t,intpd(1),t,
     +    intpd(2),t,intpd(3),t,intpd(4),t,intpd(5),t,intpd(6),t,sum1,
     +    t,intasn,t,losst
120       format(1x,f7.2,a1,i2,a1,f7.2,a1,7(f5.3,a1),f7.3,a1,f7.2,a1,
     +    f7.2)
        end if
        time = time + deltime
        go to 70
      else
        temp = 0.0
        do 125 j = 2,ndel
          intpd(1) = intpd(1) + intpd(j)
          intpd(j) = 0.0
125     continue
        temp = 0.0
        do 130 j = ndel - 1,1,-1
          do 135 i = 1,ndel - j
            temp = temp + prpsamp(j)*prpdec(i,j)
135       continue
130     continue
        exploss(pindex) = exploss(pindex) + losst*psamp + temp*losst
        if (respond.eq.1) then
          write (iout,140) time,t,cplan,t,dent,t,psamp,t,intpd(1),t,
     +    intpd(2),t,intpd(3),t,intpd(4),t,intpd(5),t,intpd(6),t,sum1,
     +    t,intasn,t,losst
140       format(1x,f7.2,a1,i2,a1,f7.2,a1,7(f5.3,a1),f7.3,a1,f7.2,a1,
     +    f7.2)
        end if
      end if

c end of time loop
```

```
c compute OC, bouts, average total sample size, losses

      oc(pindex) = 1 -sum1
      avgtotn(pindex) = sum2
      expbout(pindex) = sum3
      if (sumptrt(bout).lt.0.5) then
        lossp1(pindex) = losst
      else
        call interp(losssb,sumptrt,0.5,bout,1,lossp1(pindex))
      endif
      if (sumptrt(bout).lt.0.8) then
        lossp2(pindex) = losst
      else
        call interp(losssb,sumptrt,0.8,bout,1,lossp2(pindex))
      endif
      if (sumptrt(bout).lt.0.95) then
        lossp3(pindex) = losst
      else
        call interp(losssb,sumptrt,0.95,bout,1,lossp3(pindex))
      endif

c increment population index

      pindex = pindex + 1
      if (pindex.le.ipops) go to 60
      pindex = pindex - 1

c write summary statistics to output file and to the screen

      write (iout,200)
200   format (' Start   Interval   Last sample at   Possible bouts')
      write (iout,205) begtime,deltime,lsttime,potbouts
205   format (f6.1,2x,f6.1,7x,f6.1,12x,f6.1)
      write (*,210)
210   format (' Start   Interval   Last sample at   Possible bouts')
      write (*,215) begtime,deltime,lsttime,potbouts
215   format (f6.1,2x,f6.1,7x,f6.1,12x,f6.1)
      write (iout,220) popscale
220   format (' Population scale factor  = ',f6.3)
      write (*,225) popscale
225   format (' Population scale factor  = ',f6.3)
      write (*,230)
230   format (' Pop   Sum.den   OC      ASN        Bouts   Exp.loss   P0.5loss
     + P0.2loss   P0.05loss')
      write (iout,235) t,t,t,t,t,t,t,t
235   format (' Pop.',a1,'Cum.den.',a1,'OC',a1,'ASN',
     +a1,'Bouts',a1,'Exp.loss',a1,'P(0.5)loss',a1,'P(0.2)loss',a1,
     +'P(0.05)loss')
      do 240 j = 1,pindex
        write (iout,245) j,t,sumden(trtcnt,j),t,oc(j),t,
     +  avgtotn(j),t,expbout(j),t,exploss(j),t,lossp1(j),t,
     +  lossp2(j),t,lossp3(j)
        write (*,246) j,sumden(trtcnt,j),oc(j),avgtotn(j),expbout(j),
        exploss(j),lossp1(j),lossp2(j),lossp3(j)
245     format (1x,i2,a1,f7.2,a1,f4.2,a1,f7.2,a1,f5.2,a1,f7.2,
     +  a1,f7.2,a1,f7.2,a1,f7.2)
246     format (1x,i2,3x,f7.2,2x,f4.2,2x,f7.2,2x,f5.2,2x,f7.2,
     +  3x,f7.2,3x,f7.2,3x,f7.2)
240   continue
      write (iout,250)
```

```
250   format (' ** if P()loss  =  Cum.den, SUM(ptrt)  <  1 - P()')

260   write (*,*) 'Repeat for new population scale? (Y or N)'
      read (*,'(a)') input
      if (input.ne.'N'.and.input.ne.'n') then
        if (input.ne.'Y'.and.input.ne.'y') goto 260
      end if
      if (input .eq. 'Y' .or. input .eq. 'y') goto 46

270   write (*,*) 'Repeat for new time model? (Y or N)'
      read (*,'(a)') input
      if (input.ne.'N'.and.input.ne.'n') then
        if (input.ne.'Y'.and.input.ne.'y') goto 270
      end if
      if (input .eq. 'Y' .or. input .eq. 'y') goto 39

      close (iout)
      close (iin1)
      close (iin2)
      end

      subroutine interp (y,x,dummyx,k,trunc,yvalue)

c ***************************************************************************************
c This subroutine performs a linear interpolation of a function y  =  f(x).
c Vectors of y values (y) and x values (x) are passed along with a dummy
c x value (dummyx) for which a y value (yvalue) is sought. The dimension
c of the y and x arrays (k) must also be passed. In addition a variable
c trunc is passed that when  =  1 causes trunction of the function so that
c when dummyx  <  x(1) or dummyx  >  x(k), yvalue is constrained to be y(1) or
c y(k). When trunc is not equal to 1, yvalue is interpolated beyond y(1)
c or y(k).
c ***************************************************************************************

      real y,x,dummyx,yvalue
      integer k,trunc,j
      dimension y(k),x(k)

      if (dummyx.le.x(1)) then
        if (trunc.eq.1) then
          yvalue  =  y(1)
          goto 20
        else
          yvalue  =  y(1)-((y(2)-y(1))/(x(2)-x(1))*(x(1)-dummyx))
          goto 20
        endif
      endif

      do 10 j  =  2,k
      if (dummyx.le.x(j)) then
        yvalue  =  y(j-1)+((y(j)-y(j-1))/(x(j)-x(j-1))*(dummyx-x(j-1)))
        goto 20
      endif
10    continue

      if (trunc.eq.1) then
        yvalue  =  y(k)
        goto 20
      else
        yvalue  =  y(k)+((y(k)-y(k-1))/(x(k)-x(k-1))*(dummyx-x(k)))
        goto 20
      endif
```

```
20    continue
      return
      end
```

afcm: adaptive frequency classification monitoring

Background: This program constructs stop lines for and evaluates the performance of a set of adaptive frequency classification monitoring (AFCM) sampling plans. AFCM sampling plans are designed to be used as a cascaded set to monitor a population through time. Decisions made via a single AFCM sampling plan are to sample the population again after some time interval or to intervene. Time intervals between sample bouts are specified as a multiple of some minimum time. The program computes descriptions of stop lines for each plan as well as probabilities of making various decisions, average sample number functions, and probabilities of terminating sampling one of three ways. All performance criteria are determined by simulating sampling from a specified distribution and applying the stop lines to the generated random variables. Sampling decisions for which probabilities are determined include resampling the population $1,2,\ldots,n$ minimum time intervals in the future and the probability of intervening. Sampling can be terminated by the sample path crossing the upper stop line for the SPRT, by crossing the intersection point, or by reaching the maximum sample size.

The programs work as follows:

1. Parameters used to describe the sampling plans are specified in a file read by the program. An example file is shown below. Labels read by the program and used to describe parameters occupy columns 1 to 30. Data values begin in column 31; the range of columns the data may be in are indicated in parentheses. The first two lines of the file are headers and are ignored; however, they must be present in the file. Currently only a negative binomial distribution can be used to describe sample counts and this distribution is specified by the integer 1. It is straightforward to modify the program to allow other distributions such as a Poisson or a binomial count model. Methods used in the program 'sprt' can be adapted for this purpose. In the remainder of this section text enclosed by single quotes denotes the label for a parameter. The 'growth rate' is the parameter r for the exponential model used to describe population growth. 'SPRT alpha and beta' are parameters for the sequential probability ratio test. The 'maximum sample size' is the maximum number of samples that will be taken before reaching a decision. The 'confidence interval alpha' is used to specify the confidence interval about estimated means used to compute the waiting time until the population is sampled again. The 'resample time interval' is the minimum time until the next sample will be taken. The 'resample time delays' specifies the number of resample time intervals sampling might be delayed. The maximum for this parameter is 5. For example, if the 'resample time interval' is 3 days and the 'resample time delays' is 3, plans would be constructed that would result in decisions to resample after 3, 6, and 9 days or to intervene. The 'number of sampling plans' specifies the number of plans to be constructed; this parameter must be ≤ 20. Each sampling plan is tied to s specific time and intervention threshold. The first plan is constructed for sampling at the 'sampling start time'. The second plan is to be used at the

'sampling start time' + the 'resample time interval' and the last plan is constructed for sampling at the 'sampling start time' + ('number of sampling plans' − 1)*('resample time interval'). 'TPL alpha and beta' are the parameters for Taylor's power law. When the negative binomial distribution is used, k is assumed to be a function of the mean. Intervention thresholds are interpolated from a set of values specified in the parameter data file. The 'threshold data points' specifies the number of values. The 'monte carlo iterations' specifies the number of simulated sampling bouts used to estimate performance criteria. Mean densities during the simulations are specified by the 'density start'ing value, 'stop'ping value, and 'incr'ement.

2. The program functions by first constructing stop lines for all sampling plans. These stop lines can then be displayed on the scren and are written to an output file. If the intersection of the SPRT upper stop limit on the maximum sample size line exceeds the first decision count, program execution is terminated. This situation signifies that inappropriate parameters (probably SPRT alpha and beta or the confidence interval alpha) have been used.
3. After generation of the stop lines, performance of each plan is simulated.

file for use with afcm.for

123456789 123456789 123456789	123456789 123456789	1234567890
distribution	1	(31-33)
growth rate	0.065	(31-35)
SPRT alpha and beta	0.2 0.2	(31-35,36-40)
maximum sample size	50	(31-33)
confidence interval alpha	0.30	(31-35)
resample time interval	7.0	(31-33)
resample time delays	4	(31)
sampling start time	1	(31-34)
number of sampling plans	14	(31-32)
TPL alpha and beta	4.32 1.42	(31-35,36-40)
threshold data points	4	(31-33)
threshold; time, value		
1.0 2.5		(1-5,6-10)
30.0 5.0		
90.0 7.5		
100.0 7.5		
monte carlo iteration	500	(31-36)
density start, stop,incr.	0.2 15.0 0.2	(31-37,38-44,45-50)

An example run is shown below. user inputs are underlined and comments are in italics.

```
> afcm
Enter name of file with parameters
afcm.in
Parameter file = afcm.in
Is this correct (Y or N)?
y
Enter name of file for output
afcm.out
```

Input parameters are displayed and confirmed

distribution	1
growth rate	0.065
SPRT alpha and beta	0.2 0.2
maximum sample size	50
confidence interval alpha	0.30
resample time interval	7.0
resample time delays	4
sampling start time	1
number of sampling plans	14
TPL alpha and beta	4.32 1.42
threshold data points	4
threshold; time, value	
monte carlo iteration	500
density start, stop, incr.	0.2 15.0 0.2

Is this correct?
y
Stop lines computed for all plans
Display results for each plan (Y or N)?
n
Performance criteria are then simulated; density and sampling plan being used are displayed on the screen. A portion of the output file is shown below. Parameters usd to construct the sampling plans are written to the output file; these are not shown.

Results for plan 1

Time = 1.0 Threshold = 2.50
SPRT H0 = 1.59 intercept = 27.44 slope = 1.98

intersection = 87 n at intersection = 30

Waiting time	Decision count	Threshold
7.0	87	3.10
14.0	65	3.71
21.0	47	4.31
28.0	33	4.91

Results for plan 2

Time = 8.0 Threshold = 3.10
SPRT H0 = 1.97 intercept = 30.03 slope = 2.46

intersection = 104 n at intersection = 30

Waiting time	Decision count	Threshold
7.0	104	3.71
14.0	76	4.31
21.0	54	4.91
28.0	36	5.25

Results for plan 14

Time = 92.0 Threshold = 7.50
SPRT H0 = 4.76 intercept = 43.41 slope = 5.93

intersection = 216 n at intersection = 5.93

Waiting time	Decision count	Threshold
7.0	216	7.50
14.0	135	7.50
21.0	84	7.50
28.0	52	7.50

14 4 *number of plans and delay intervals; used by 'cascade'*
40 1.0 *number of data points and starting time for use; used by cascade*
performance criteria for the first plan:

density	*pd1*	*pd2*	*pd3*	*pd4*	*pd5*	*asn*	*pseq1*	*psq2*	*pterminal*
0.20	0.998	0.002	0.000	0.000	0.000	50.00	0.000	0.000	1.000
0.40	0.936	0.062	0.002	0.000	0.000	50.00	0.000	0.000	1.000
0.60	0.630	0.304	0.058	0.008	0.000	50.00	0.000	0.000	1.000
0.80	0.306	0.422	0.240	0.028	0.004	49.91	0.002	0.002	0.996
1.00	0.106	0.378	0.374	0.140	0.002	50.00	0.000	0.000	1.000
1.20	0.040	0.160	0.440	0.296	0.064	49.59	0.004	0.040	0.956
1.40	0.010	0.088	0.316	0.422	0.164	48.66	0.018	0.122	0.860
1.60	0.006	0.034	0.178	0.446	0.336	46.66	0.042	0.266	0.692
1.80	0.000	0.012	0.112	0.318	0.558	43.78	0.088	0.428	0.484
2.00	0.000	0.006	0.040	0.246	0.708	41.47	0.130	0.522	0.348
2.20	0.000	0.000	0.014	0.140	0.846	37.54	0.204	0.604	0.192
2.40	0.000	0.000	0.010	0.070	0.920	34.21	0.302	0.584	0.114
2.60	0.000	0.000	0.008	0.032	0.960	31.79	0.376	0.570	0.054
2.80	0.000	0.000	0.002	0.020	0.978	27.69	0.516	0.456	0.028
3.00	0.000	0.000	0.000	0.014	0.986	26.10	0.560	0.424	0.016
3.20	0.000	0.000	0.000	0.008	0.992	24.31	0.626	0.366	0.008
3.40	0.000	0.000	0.000	0.000	1.000	21.73	0.706	0.290	0.004

.
.
.

40 8.0 *results for the second plan*

0.20	1.000	0.000	0.000	0.000	0.000	50.00	0.000	0.000	1.000
0.40	0.952	0.048	0.000	0.000	0.000	50.00	0.000	0.000	1.000
0.60	0.736	0.246	0.018	0.000	0.000	50.00	0.000	0.000	1.000
0.80	0.420	0.460	0.114	0.006	0.000	50.00	0.000	0.000	1.000
1.00	0.148	0.466	0.334	0.052	0.000	50.00	0.000	0.000	1.000
1.20	0.058	0.298	0.466	0.172	0.006	49.98	0.000	0.004	0.996
1.40	0.014	0.202	0.448	0.294	0.042	49.77	0.002	0.032	0.966
1.60	0.004	0.100	0.312	0.454	0.130	49.05	0.010	0.090	0.900

```
      program afcm

c     Program written in Fortran 77: IBM version compiled using Microsoft
c     FORTRAN compiler v5.1.
c     This program requires a uniform {0,1} random number generator. This
```

```
c     version makes use of the RANDOM subroutine provided in Microsoft
c     Fortran. If the FORTRAN compiler used does not provide a similar
c     function, one must be provided and small sections of code in some
c     subroutines must be changed.
c     Program written by Jan Nyrop and Wopke van der Werf
c     Department of Entomology, NYSAES,
c     Cornell University, Geneva, NY 14456
c     Last modification date: 2 February, 1993

c ************************************************************************************
c ********************************* Variable descriptions ****************************
c ************************************************************************************

c asn, average sample size
c avg, average of h0 and h1
c cialpha, alpha used to construct confidence limits
c cid(i), critical initial density; density at time that will not
c exceed a threshold at time t +i*sint
c counter, integer counter
c dec(i), number of sample bouts that end with dec(i),
c i = 1 is the longest waiting time, i = ndelay + 1 = intervene
c deccnt(nplans,i), count of organisms that dictate specific time delays
c (sint*i) to the next sample bout
c deltime(i), time delays (i*sint) between sample bouts
c density, mean density used in simulations
c device, output device
c distr, distribution of sample counts
c dstart,dstop,dincr, starting and stopping density,density increment
c for simulations
c fthresh(nplan,ndelay), future threshold values used to calculate cid
c ftime, future time; used to compute the future threshold
c when determining deccnt
c growth, function for calculating growth rate
c h0(nplans),h1(nplans), hypotheses for SPRT for each sampling plan
c i, integer index
c ii, integer index
c intsct(nplans), total count of organisms where upper stop
c line for SPRT = count corresponding to decision to wait 1 time interval
c intsctn(nplans), sample size that corresponds to intsct on SPRT stop line
c j, integer index
c k, ngb k
c lowi(nplans),highi(nplans),slope(nplans), low and high
c intercept, slope for SPRT
c mci, monte carlo iterations
c ndelays, number of time delays (sint) sampling may be delayed; maximum of 5
c nmax, maximum sample size
c nplans, number of plans to be developed; maximum of 20
c nthresh, number of values threshold is to be interpolated
c from pdec(ndelay + 1), probability of a decision indexed by mean density
c pseq1, probability of reaching a decision by crossing the SPRT stop line
c pseq2, probability of reaching a decision by crossing intsct before nmax
c pterm, probability of reaching a decision after taking
c the maximum number of samples
c r, growth rate for exponential model
c samples, number of samples taken during a monte carlo iteration
c sem, estimated standard error of the mean
c seq1, number of sampling decisions made by crossing upper SPRT limit
c seq2, number of sampling decisions reached by crossing
c intsct before the maximum sample size is reached
c sint, sample time interval; minimum time delay between samples
```

```
c sprta, sprtb, alpha and beta for sprt
c start, starting time
c terminal, number of sampling decisions made with the maximum
c number of samples
c thresh(nthresh), threshold values corresponding to tthresh
c time(nplans), time each sampling plan is first used
c total, total of x over samples in a monte carlo iteration
c tpla, tplb, a and b for Taylor's power law
c tsamps, total number of samples taken during simulation for a mean
c tthresh(nthresh), times corresponding to threshold values
c ucl, upper confidence level (based on cialpha) for xbar
c usl, upper stop limit for SPRT
c uslnmax, upper stop limit for SPRT at nmax
c x, random variable
c xbar, estimated mean density
c znormal, function that computes a standard normal deviate for alpha

      integer distr,nmax,start,nplans,nthresh,j,ndelays,i,
     +uslnmax,counter,ii,intsct(20),intsctn(20),deccnt(20,5),
     +mci,device,dec(6),tsamps,seq1,seq2,terminal,samples,n,
     +dindex,points
      real r,sprta,sprtb,cialpha,sint,tpla,tplb,growth,time(20),
     +h0(20),h1(20),lowi(20),highi(20),slope(20),k,avg,sem,ucl,
     +xbar,ftime,znormal,dstart,dstop,dincr,density,x,total,usl
      real tthresh(100),thresh(100),cid(5),fthresh(20,5),
     +deltime(5),pdec(6),asn,pseq1,pseq2,pterm
      character*30 label(14)
      character*12 file1,file2
      character*1 input
      character*45 frmt
      character*2 rep

      frmt  = ' (1x,f7.2,2x,   (f5.3,2x),f7.2,3(2x,f5.3))'

1     write ( (a)') ' Enter name of file with parameters'
      read (*,'(a)') file1
      write (*,2) file1
2     format (' Parameter file  =  ',a12)
3     write (*,*) 'Is this correct? (Y or N)'
      read (*,'(a)') input
      if (input.ne.'N'.and.input.ne.'n') then
        if (input.ne.'Y'.and.input.ne.'y') goto 3
      end if
      if (input .eq. 'N' .or. input .eq. 'n') goto 1
      write (*,'(a)') ' Enter name of file for output'
      read (*,'(a)') file2
      open (unit = 1,file  =  file1,status  = 'old')
      open (unit = 2,file  =  file2,status  = 'unknown')

      read (1,10)
10    format (/)
      read (1,11) label(1),distr
11    format (a30,i3)
      read (1,15) label(2),r
15    format (a30,f6.3)
      read (1,20) label(3),sprta,sprtb
20    format (a30,f5.3,1x,f5.3)
      read (1,25) label(4),nmax
25    format (a30,i4)
      read (1,30) label(5),cialpha
```

```
30    format (a30,f5.3)
      read (1,35) label(6),sint
35    format (a30,f6.1)
      read (1,36) label(7),ndelays
36    format (a30,i1)
      read (1,40) label(8),start
40    format (a30,i3)
      read (1,45) label(9),nplans
45    format (a30,i2)
      read (1,50) label(10),tpla,tplb
50    format (a30,f5.3,1x,f5.3)
      read (1,55) label(11),nthresh
55    format (a30,i3)
      read (1,60) label(12)
60    format (a30)
      do 70 j  =  1,nthresh
        read (1,65) tthresh(j),thresh(j)
65      format (f5.1,1x,f5.1)
70    continue
      read (1,80) label(13),mci
80    format (a30,i6)
      read (1,85) label(14),dstart,dstop,dincr
85    format (a30,f7.2,f7.2,f6.3)

      write (2,100) label(1),distr
      write (*,100) label(1),distr
100   format (' ',a30,i3)
      write (2,105) label(2),r
      write (*,105) label(2),r
105   format (1 ',a30,f6.3)
      write (2,120) label(3),sprta,sprtb
      write (*,120) label(3),sprta,sprtb
120   format (' ',a30,f5.3,1x,f5.3)
      write (2,125) label(4),nmax
      write (*,125) label(4),nmax
125   format (' ',a30,i4)
      write (2,130) label(5),calpha
      write (*,130) label(5),cialpha
130   format (' ',a30,f5.3)
      write (2,135) label(6),sint
      write (*,135) label(6),sint
135   format (' ',a30,f6.1)
      write (2,136) label(7),ndelays
      write (*,136) label(7),ndelays
136   format (' ',a30,i1)
      write (2,140) label(8),start
      write (*,140) label(8),start
140   format (' ',a30,i3)
      write (2,145) label(9),nplans
      write (*,145) label(9),nplans
145   format (' ',a30,i2)
      write (2,150) label(10),tpla,tplb
      write (*,150) label(10),tpla,tplb
150   format (' ',a30,f5.3,1x,f5.3)
      write (2,155) label911),nthresh
      write (*,155) label(11),nthresh
155   format (' ',a30,i3)
      write (2,160) label(13),mci
      write (*,160) label(13),mci
160   format (' ',a30,i6)
```

```
      write (2,165) label(14),dstart,dstop,dincr
      write (*,165) label(14),dstart,dstop,dincr
165   format (' ',a30,f7.2,f7.2,f6.3)

180   write (*,*) 'Is this correct? (Y or N)'
      read (*,'(a)') input
      if (input.ne.'N'.and.input.ne.'n') then
        if (input.ne.'Y'.and.input.ne.'y') goto 180
      end if
      if (input .eq. 'N' .or. input .eq. 'n') goto 5000

      n = ndelays + 1
      write (rep,190) n
190   format (i2)
      frmt(13:14) = rep

c ************************ Stop lines ************************

c ****** time loop ******

      do 200 j = 1,nplans
        time(j) = float(start) + float(j-1)*sint

c ****** determine threshold ******

        call interp (thresh,tthresh,time(j),nthresh,2,h1(j))

c ******* compute sprt stop lines *******

        h0(j) = h1(j)/growth(r,sint)
        if (distr.eq.1) then
          avg = (h1(j) + h0(j))/2.0
          k = avg**2/(tpla*avg**tplb - avg)
        endif
        call sprtlim (distr,h1(j),h0(j),sprta,sprtb,k,lowi(j),
     +                highi(j),slope(j))

c ****** compute fixed sample size stop line ******

c     compute each critical initial density (cid)

        do 220 i = 1,ndelays
          deltime(i) = float(i)*sint
          ftime = time(j) + deltime(i)
          call interp (thresh,tthresh,ftime,nthresh,2,fthresh(j,i))
          cid(i) = fthresh(j,i)/growth(r,deltime(i))
220     continue

c     find means with upper confidence limit = cid

        uslnmax = nint( highi(j) + slope(j)*float(nmax))
        do 230 i = 1,ulsnmax
          xbar = float(i)/float(nmax)
          sem = sqrt(trpla*xbar**tplb/nmax)
          counter = 0
          ucl = xbar + znormal(cialpha)*sem
```

```
            if (ucl*float(nmax).gt.uslnmax) then
              write (*,*) 'UCL  >  max sprt upper stop limit'
              write (2,*) 'UCL  >  max sprt upper stop limit'
              goto 5000
            endif
            do 240 ii  =  1,ndelays
              if (ucl.lt.cid(ii)) then
                deccnt(j,ii)  =  i
                counter  =  counter + 1
              endif
240         continue
          if (counter.eq.0) goto 250
230       continue
250       continue

c ****** integrate sprt and fixed sample size stop line ******

          intsct(j)  =  deccnt(j,1)
          intsctn(j)  =  nint((float(intsct(j)) - highi(j))/slope(j))

c ****** output stop lines for each sampling plan ******

            device  =  2
            call display1 (j,device,time,h1,h0,highi,slope,intsct,intsctn,
     +        deltime,deccnt,fthresh,ndelays)

200   continue

      write (*,*) 'Stop lines computed for all plans'
      write (*,*) 'Display results for each plan (Y or N)?'
360   read (*,'(a)') input
      if (input.ne.'N'.and.input.ne.'n') then
        if (input.ne.'Y'.and.input.ne.'y') goto 360
      end if
      if (input .eq. 'Y' .or. input .eq. 'y') then
        device  =  0
        do 370 j  =  1,nplans
          call display1 (j,device,time,h1,h0,highi,slope,intsct,
     +      intsctn,deltime,deccnt,fthresh,ndelays)
          pause 'Enter to continue'
370     continue
      endif

c ****** write number of plans and delays to output file ******

      write (2,380) nplans,ndelays
380   format (1x,i2,2x,i2)

c ****************** Pdec and ASN functions ******************

c ****** determine number of data points for each function ******

      points  =  anint((dstop - dstart)/dincr) + 1

c ****** loop for sampling plans ******
```

```
      do 400 j = 1,nplans
        write (*,401) j,nplans
401     format (' ','sampling from plan ',i2,' of ',i2,' total')
        write (2,402) points, time(j)
402     format (1x,i3,2x,f6.1)


c ****** loop for density ******


        do 405 dindex = 1,points
          density = dstart + dincr*float(dindex - 1)
          k = density**2/(tpla*density**tplb - density)
          do 410 i = 1,6
            pdec(i) = 0.0
            dec(i) = 0
410       continue
          tsamps = 0
          seq1 = 0
          seq2 = 0
          terminal = 0


c ****** loop for Carlo iterations ******


            do 420 i = 1,mci
              samples = 0
              total = 0.0
c generate random variable
425           call ngb(density,k,x)
              total = total + x
              samples = samples + 1
c # of samples < intersection number
              if (samples.lt.intsctn(j)) then
                usl = highi(j) + slope(j)*float(samples)
                if (total.gt.usl) then
                  dec(ndelays + 1) = dec(ndelays + 1) + 1
                  tsamps = tsamps + samples
                seq1 = seq1 + 1
                  togo 420
                else
                  goto 425
                endif
c # of sample >= intersection number and < maximum samples
              elseif (samples.lt.nmax) then
                if (total.gt.intsct(j)) then
                  dec(ndelays + 1) = dec(ndelays + 1) + 1
                  tsamps = tsamps + samples
                  seq2 = seq2 + 1
                  goto 420
                else
                  goto 425
                endif
c # of samples = maximum samples
              elseif (samples.eq.nmax) then
                do 430 ii = ndelays,1,- 1
                  if (total.lt.deccnt(j,ii)) then
                    dec(ndelays - ii + 1) = dec(ndelays - ii + 1) + 1
                    tsamps = tsamps + samples
                    terminal = terminal + 1
                    goto 420
```

```
                endif
430             continue
                dec(ndelays + 1) = dec(ndelays + 1) + 1
                tsamps = tsamps + samples
                terminal = terminal + 1
              endif
420           continue

c ****** end of loop for Monte Carlo iterations ******

c ****** compute statistics and write results ******

            do 450 i = 1,ndelays + 1
              pdec(i) = float(dec(i))/float(mci)
450         continue
            asn = float(tsamps)/float(mci)
            pseq1 = float(seq1)/float(mci)
            pseq2 = float(seq2)/float(mci)
            pterm = float(terminal)/float(mci)
            write (2,frmt) density,(pdec(i), i = 1,ndelays + 1),asn,
     +                   pseq1,pseq2,pterm
            write (*,frmt) density,(pdec(i), i = 1,ndelays + 1),asn,
     +                   pseq1,pseq2,pterm
405     continue

c ****** end of loop for density ******

400   continue

c ****** end loop for sampling plans ******

5000  continue
      close (1)
      close (2)
      stop
      end

c ***************************************************************************************
c ****************************** Functions and Subroutines ******************************
c ***************************************************************************************

      function growth(r,t)
      real r,t
      growth = exp(r*t)
      return
      end

      subroutine sprtlim (distr,h1,h0,alpha,beta,k,lowi,highi,
     +slope)

c ***************************************************************************************
c This subroutine computes stop limit parameters for an SPRT
c Parameters passed are; distr - distribution, 1 = ngb, 2 = Poisson,
c 3 = binomial; h1 and h0 null and alternate hypotheses
c ***************************************************************************************
      real h1,h0,alpha,beta,lowi,highi,slope,k,a,lna,b,lnb,q1,q0,
     +bvar1,pnb0,pnb1,qnb0,qnb1,vnb1,vnb2
      integer distr
```

```
c variables used for all distributions

      a = (1-beta)/alpha
      lna = log(a)
      b = beta/(1-alpha)
      lnb = log(b)

c variables used for binomial and negative binomial

      if (distr.eq.3) then
        q1 = 1-h1
        a0 = 1-h0
        bvar1 = log((h1*q0)/(h0*q1))
      end if
      if (distr.eq.1) then
        pnb0 = h0/k
        pnb1 = h1/k
        qnb0 = 1 + pnb0
        qnb1 = 1 + pnb1
        vnb1 = (pnb1*qnb0)/pnb0*qnb1)
        vnb2 = qnb0/qnb1
      end if

c calculate stop limit parameters

      if (distr.eq.3) then
        lowi = lnb/bvar1
        highi = lna/bvar1
        slope = log(q0/q1)/bvar1
      else if (distr.eq.2) then
        lowi = lnb/log(h1/h0)
        highi = lna/log(h1/h0)
        slope = (h1 - h0)/log(h1/h0)
      else
        lowi = lnb/log(vnb1)
        highi = lna/log(vnb1)
        slope = k*(log(qnb1/qnb0)/log(vnb1))
      end if
      return
      end


      subroutine display1 (j,device,time,h1,h0,highi,slope,intsct,
     +intsctn,deltime,deccnt,fthresh,ndelays)
      integer device,j,intsctn(20),intsct(20),deccnt(20,10),i,ndelays
      real time(20),h1(20),h0(20),highi(20),slope(20),deltime(10),
     +fthresh(20,10)
      write (device,10) j
10    format (' Results for plan ',i3)
      write (device,20) time(j),h1(j)
20    format (' Time = ',f6.1,' Threshold = ',f6.2)
      write (device,30) h0(j),highi(j),slope(j)
30    format (' SPRT H0 = ',f6.2,' intercept = ',f7.2,
     +' slope = ',f6.2)
      write (device,40) intsct(j),intsctn(j)
40    format (' intersection = ',i7,' n at intersection = ',i4)
      write (device,50)
50    format (' Waiting time  Decision count  threshold')
      do 55 i = 1,ndelays
        write (device,60) deltime(i),deccnt(j,i),fthresh(j,i)
```

```
60      format (' ',f6.1,8x,i5,11x,f7.2)
55    continue
      return
      end

      subroutine interp (y,x,dummyx,k,trunc,yvalue)

c ***************************************************************************************
c This subroutine performs a linear interpolation of a function y = f(x).
c Vectors of y values (y) and x values (x) are passed along with a dummy
c x value (dummyx) for which a y value (yvalue) is sought. The dimension
c of the y and x arrays (k) must also be passed. In addition a variable
c trunc is passed that when = 1 causes truncation of the function so that
c when dummyx < x(1) or dummyx > x(k), yvalue is constrained to be
c y(1) or y(k). When trunc is not equal to 1, yvalue is interpolated
c beyond y(1) or y(k).
c ***************************************************************************************

      real y,x,dummyx,yvalue
      integer k,trunc,j
      dimension y(k),x(k)

      if (dummyx.le.x(1)) then
        if (trunc.eq.1) then
          yvalue = y(1)
          goto 20
        else
          yvalue = y(1)-((y(2)-y(1))/(x(2)-x(1))*(x(1)-dummyx))
          goto 20
        endif
      endif

      do 10 j = 2,k
        if (dummyx.le.x(j)) then
          yvalue = y(j-1) + ((y(j)-y(j-1))/(x(j)-x(j-1))*(dummyx-x(j-1)))
          goto 20
        endif
10    continue

      if (trunc.eq.1) then
        yvalue = y(k)
        goto 20
      else
        yvalue = y(k)+((y(k)-y(k-1))/(x(k)-x(k-1))*(dummyx-x(k)))
        goto 20
      endif

20    continue
      return
      end

*****************************************************************************************
*     The function ZNORMAL returns the x-value at which the cumulative                  *
*     normal probability density function reaches the y-value p.                        *
*     The algorithm was taken from Abramowitch & Stegun (1972):                         *
*     Handbook of mathematical functions, 9th ed. Dover, New York,                      *
*     1046 pp. p. 933, equation 26.2.23                                                 *
*     error < 4.5E-4                                                                    *
*                                                                                       *
*     Wopke van der Werf, 4/30/92                                                       *
*****************************************************************************************
```

```
      function znormal(a)
      real c0,c1,c2,d1,d2,d3,cdf,p,t,x,a
      cdf  =  1 - a
      c0  =  2.515517
      c1  =  0.802853
      c2  =  0.010328
      d1  =  1.432788
      d2  =  0.189269
      d3  =  0.001308

      if ( cdf .le. 0. ) then
        znormal  =  -5.
        return
      else if ( ( cdf .gt. 0. ) .and. (cdf . lt . 0.5 ) ) then
        p  =  cdf
        t  =  sqrt( log( p**(-2.) ) )
        x  =  t - ( c0 + c1 * t + c2 * t**2. ) / ( 1 + d1 * t + d2 * t**2.
     $                                              + d3 * t**3. )
        znormal  =  -x
        return
      else if ( ( cdf .ge. 0.5 ) .and. ( cdf .lt. 1. ) ) then
        p  =  1 - cdf
        t  =  sqrt( log( p**(-2.) ) )
        x  =  t - ( c0 + c1 * t + c2 * t**2. ) / ( 1 + d1 * t + d2 * t**2.
     $                                              + d3 * t**3. )
        znormal  =  x
        return
      else if ( cdf .ge. 1. ) then
        znormal  =  5.
        return
      endif

      return
      end

      subroutine ngb(mean,k,x)

c ***************************************************************************************
c This subroutine generates negative binomial distributed random
c variables using a rejection method. It requires uniform {0,1}
c random variates be generated as the variable r. If the Fortran
c compiler does not support a 'random' function, one must be provided.
c ***************************************************************************************
      real x,mean,k,px,ppx,pxsum,r
      call random(r)
      x  =  0.0
      px  =  1/((1+mean/k)**k)
      ppx  =  px
      pxsum  =  px
      if (r. .le. px) goto 4
5     x  =  x+1
      px  =  ((k+x-1.)/x)*(mean/(mean+k))*ppx
      ppx  =  px
      pxsum  =  pxsum+px
      if (r .le. pxsum) goto 4
      if (x .gt. 200.) goto 4
      goto 5
4     continue
      return
      end
```

REFERENCES

1. Binns, M. R. and Nyrop, J. P., Sampling insect populations for the purpose of IPM decision making, *Annu. Rev. Entomol.*, 37, 427, 1992.
2. Nyrop, J. P., Sequential classification of prey-predator ratios with application to European red mite (Acari: Tetranychidae) and *Typlodromus pyri* (Acari: Phytoseiidae) in New York apple orchards, *J. Econ. Entomol.*, 80, 14, 1988.
3. Nyrop, J. P. and Binns, M. R., Algorithms for computing operating characteristic and average sample number functions for sequential sampling plans based on binomial count models and revised plans for European red mite (Acari: Tetranychidae) on apple, *J. Econ. Entomol.*, 85, 1253, 1992.
4. Croft, B. A. and Nelson, E. E., An index to predict efficient interaction of *Typhlodromus occidentalis* in control of *Tetranychus mecdanieli* in Southern California apple trees, *J. Econ. Enomol.*, 64, 845, 1972.
5. Cochrar., W. G., *Sampling Techniques*, 3rd ed., John Wiley & Sons, New York, 1977, chap. 6.
6. Taylor, L. R., Aggregation, variance, and the mean, *Nature (London)*, 189, 732, 1961.
7. Wald, A., *Sequential Analysis*, John Wiley & Sons, New York, 1947.
8. Wilson, L. T., Estimating the abundance and impact of arthropod natural enemies in IPM systems, in *Biological Control in Agricultural IPM Systems*, Hoy, M. A. and Herzog, D. C., Eds., Academic Press, New York, 1985, pp. 303.