OPTIMAL STOPPING RULES FOR SOFTWARE TESTING

Nilgun Morali  
Department of Statistics  
Aegean University  
Refik Soyer  
Department of Management Science  
The George Washington University

DP #94-33
OPTIMAL STOPPING RULES FOR SOFTWARE TESTING

NILGUN MORALI

Department of Statistics, Aegean University, Turkey

REFIK SOYER

Department of Management Science, The George Washington University, U.S.A.

ABSTRACT

In this paper we address the problem of determining when to terminate the testing/modification process and release a piece of software during the software development. We present a Bayesian decision theoretic approach by formulating the software release problem as a sequential decision problem. By using a non-Gaussian Kalman filter type of model, proposed by Chen and Singpurwalla (1994) to track software reliability, we are able to obtain tractable expressions for inference and determine a one-stage look ahead optimal stopping rule under a specific class of loss functions.

Key words: Bayesian inference, dynamic programming, Kalman filter, preposterior analysis, software reliability.
1. INTRODUCTION AND OVERVIEW

During the development phase, software is subjected to several stages of testing to identify existing problems. At the end of each test stage corrections and modifications are made to the software with the hope of increasing its reliability. This process is called reliability growth. It is possible, however, that a particular modification, or a series of modifications, could lead to a deterioration in performance of the software. The two key statistical issues are:

(i) how to model and how to describe the changes in the performance of the software as a result of the modifications,

(ii) how to decide when to terminate the testing/modification process and to release the software.

Most of the literature in software reliability centers around the first issue, namely, that of modeling and inference for assessment of software reliability and neglects the important issue of determining when to stop the testing process and release the software. Exceptions to these are the works of Forman and Singpurwalla (1977, 1979), Okumoto and Goel (1980), Yamada, Narihisa and Osaki (1984) and Ross (1985). None these of approaches are based on a formal decision-theoretic approach. More recently work such as Dalal and Mallows (1988), and Singpurwalla (1989, 1991) propose a Bayesian decision theoretic approach to the problem. The work of Dalal and Mallows suffers from the disadvantage of an asymptotic solution whereas the work of Singpurwalla addresses a two-stage problem and suffers from the disadvantage of computational difficulties.

In this paper, we present a Bayesian decision theoretic approach by formulating the optimal release problem as a sequential decision problem. The solution of sequential decision problems of this type is usually too difficult to analyze due to the reliance on preposterior analysis [see for example van Dorp, Mazzuchi and Soyer (1994)]. Here, by using a nonGaussian Kalman filter type of model suggested by Chen and Singpurwalla (1994) to track software reliability, we are able to obtain tractable expressions for inference and determine a one-stage look ahead stopping rule under reasonable conditions and a class of loss functions.
A synopsis of our paper is as follows:

In Section 2, we formulate the problem of optimal stopping as a sequential decision problem and present the conditions for optimality of one-stage look ahead stopping rule for a general loss function. In Section 3, we present a version of the non-Gaussian Kalman filter type of model suggested by Chen and Singpurwalla (1994) for monitoring software reliability and discuss sequential inference and m-step ahead predictions for the model. In Section 4, we consider specific forms of loss function and using the model of Section 3 develop the conditions under which the one-stage look ahead rule is optimal. Finally, in Section 5, we illustrate the use of the stopping rule with an example.

2. THE SEQUENTIAL STOPPING PROBLEM

Let \( X_i, i=1, 2, \ldots \), denote the life-length of the software during the \( i \)-th stage of testing, after the \((i-1)\)st modification, and \( \theta_i \) denote the failure rate of the software during the \( i \)-th stage. A common view in software reliability modeling is that, unlike a hardware component, software does not exhibit an aging or wearout behavior. Thus, it is assumed that the failure rate of the software is constant during the \( i \)-th stage of testing and given \( \theta_i \) the failure behavior of the software is described by the exponential density

\[
p(x_i \mid \theta_i) = \theta_i \exp(-\theta_i x_i).
\]  

The special feature of the model (2.1) is that it allows the failure rate \( \theta_i \) to change from one testing stage to another as a result of corrections made to the software. At the end of each stage, following modifications made to the software, a decision must be made whether to terminate the testing/modification process. Thus after the completion of \( i \) stages of testing, the decision of whether or not to stop testing will be based on \( X^{(i)} = \{X^{(0)}, x_1, x_2, \ldots, x_i\} \), where \( X^{(0)} \) represents available information concerning failure characteristics of software prior to any testing.

In the Bayesian decision-theoretic framework the decision to whether or not stop testing must be based on maximization (minimization) of expected utility (loss). Any reasonable loss (or
utility) function should consider the tradeoff between the loss associated with extensive testing versus the loss associated with releasing an unreliable piece of software. In consideration of such tradeoff, we define the loss associated with stopping and releasing the software after the i-th testing stage as
\[
\mathcal{L}_i(X^{(0)}, \theta_{i+1}) = \sum_{j=1}^{i} \mathcal{L}_T(X_j) + \mathcal{L}_S(\theta_{i+1}),
\]  
(2.2)
where \(\mathcal{L}_T(\cdot)\) represents the loss due to testing for one stage, and \(\mathcal{L}_S(\cdot)\) relates the loss associated with stopping and releasing the software. We note that \(\mathcal{L}_0\) (the loss associated with releasing the software before any testing) is a function of \(\theta_1\) (the failure rate prior to any modification) since \(\sum_{j=1}^{0} \{ \cdot \} = 0\).

The stopping problem can be represented as a sequential decision problem as given by the \(m\)-stage decision tree in Figure 1 and can be solved using dynamic programming. Solution of the tree proceeds in the usual way by taking expectation at random nodes and minimizing the expected loss at the decision nodes. At decision node \(i\), the additional expected loss associated with the STOP and the TEST decisions are given by the terms \(E[\mathcal{L}_S(\theta_{i+1}) | X^{(0)}]\) and \(E[\mathcal{L}_T(X_{i+1}) | X^{(i)}] + L^*_i + 1\), respectively, where
\[
L^*_i = \text{MIN}\left\{ E[\mathcal{L}_S(\theta_{i+1}) | X^{(i)}], E[\mathcal{L}_T(X_{i+1}) | X^{(i)}] + L^*_i + 1 \right\}
\]  
(2.3)
for \(i=0, 1, \ldots\), and the optimal decision at decision node \(i\) then is the one associated with \(L^*_i\). In Figure 1, \(m\), the maximum number of testing stages, can be considered infinite. We note that even for the case of finite \(m\), the calculation of \(L^*_i\) in (2.3) is not trivial as it involves implicit computation of expectations and minimizations at each stage. Following van Dorp, Mazzuchi and Soyer (1994), we rewrite (2.3) as
\[
L^*_i = \text{MIN}_{\delta=0, 1, \ldots} \left\{ L^{(\delta)}_i \right\}
\]  
(2.4)
where
\[ L_i^{(\delta)} = \sum_{j=1}^{\delta} E[\mathcal{L}_T(X_{i+j}) \mid X^{(i)}] + E[\mathcal{L}_S(\theta_{i+\delta+1}) \mid X^{(i)}] \]  

(2.5)

is the additional expected loss associated with testing for \( \delta \) more stages after the \( i \)-th modification to the software.

![Decision Tree for Software Release Problem](image)

Figure 1. Decision Tree for Software Release Problem.

Given the above development, using the result by van Dorp, Mazzuchi and Soyer (1994) we can show that following theorem provides an optimal stopping rule:

**Theorem 1:** Let \( E[\mathcal{L}_T(X_j) \mid X^{(i)}] \) be increasing in \( j \) for \( j = i+1, \ldots \), and \( E[\mathcal{L}_S(\theta_j) \mid X^{(i)}] \) be discrete convex in \( j \) for \( j = i+1, \ldots \), then after the \( i \)-th modification to the software, the following stopping rule is optimal

\[
\text{if} \quad \begin{cases} 
L_i^{(1)} - L_i^{(0)} < 0 & \Rightarrow \text{Continue Testing} \\
L_i^{(1)} - L_i^{(0)} \geq 0 & \Rightarrow \text{Stop Testing and Release.} 
\end{cases} \quad (2.6)
\]

**Proof:** The proof follows along the same lines as in van Dorp, Mazzuchi and Soyer (1994).

Thus, under the conditions stated by the theorem, a one-stage look ahead rule is optimal. As noted by the above authors, the Theorem implies that it is optimal to stop testing when the
expected increase in loss due to testing an additional stage is greater than the expected decrease in loss due to the improvement in reliability resulting from testing an additional stage.

3. A MODEL FOR THE FAILURE RATE OF SOFTWARE

As pointed out in Section 2, the failure rate of the software is constant during the i-th stage of testing and therefore given θ_i, the life-length of the software at the i-th stage is assumed to be the exponentially distributed as in (2.1). Due to the modifications made to the software, it is expected that the failure rate at the i-th stage will be related to the failure rate at the previous stages. To reflect this, we assume that given X^{(i-1)} and θ_{i-1}

\[
\left( \frac{\theta_i}{\rho \theta_{i-1}} \mid \theta_{i-1} \right) \sim \text{Beta}(\gamma \alpha_{i-1}, (1 - \gamma) \alpha_{i-1}),
\]

(3.1)

where ρ, γ and α_{i-1} are known nonnegative quantities such that 0 < γ < 1. We note that (3.1) can be written as

\[
\theta_i = \rho \theta_{i-1} \xi_i
\]

(3.2)

with \(\xi_i \sim \text{Beta}(\gamma \alpha_{i-1}, (1 - \gamma) \alpha_{i-1})\). Equation (3.2) can be thought of as a state equation and it provides an ordering of \(\theta_i\)'s since the above implies that \(\theta_i < \rho \theta_{i-1}\) for all i. Furthermore, given X^{(i-1)} we assume that \(\theta_{i-1}\) has gamma distribution, with shape parameter \(\alpha_{i-1}\) and scale parameter \(\beta_{i-1}\) denoted as

\[
(\theta_{i-1} \mid X^{(i-1)}) \sim \text{Gamma}(\alpha_{i-1}, \beta_{i-1}).
\]

(3.3)

Given the above setup, it can be shown that, prior to the i-th stage of testing our uncertainty about the failure rate \(\theta_i\) is described by

\[
(\theta_i \mid X^{(i-1)}) \sim \text{Gamma}(\gamma \alpha_{i-1}, \frac{\beta_{i-1}}{\rho}).
\]

(3.4)
As shown by Smith and Miller (1986), once $X_i$ is observed, the posterior distribution of $\theta_i$ given $X^{(i)}$ is obtained, via the Bayes' Theorem, as

$$(\theta_i \mid X^{(i)}) \sim \text{Gamma}(\alpha_i, \beta_i),$$

(3.5)

where $\alpha_i = \gamma\alpha_{i-1} + 1$ and $\beta_i = (\beta_{i-1}/\rho) + X_i$. Once we specify the starting distribution as $(\theta_0 \mid X^{(0)}) \sim \text{Gamma}(\alpha_0, \beta_0)$, the above development and the observation model given by (2.1) provide us with a nonGaussian Kalman filter type of model. A version of this model has been used by Chen and Singpurwala (1994) for tracking reliability growth of software. Given the above development the predictive distribution of $X_i$ given $X^{(i-1)}$ is a Pareto of the form

$$p(x_i|X^{(i-1)}) = \frac{\gamma \alpha_{i-1} (\beta_{i-1}/\rho)^{\alpha_{i-1}}}{(\beta_{i-1}/\rho + x_i)^{\alpha_{i-1}+1}}.$$  

(3.6)

As pointed out by Chen and Singpurwala (1994), the parameter $\rho$ provides information about the growth or decay of the reliability of software. For example, the values of $\rho$ less than or equal to 1 imply that the failure rates are decreasing from one stage to another whereas $\rho > 1$ implies failure rates will be increasing in $i$. We note that all the above results are conditional on the growth parameter $\rho$. As noted by the above authors, it is more reasonable to treat $\rho$ as an unknown quantity representing the behavior of failure rates over stages of testing. When $\rho$ is unknown, we define a k-point discrete distribution as the prior distribution. We specify the prior distribution by using a discretization of the beta density on $(\rho_L, \rho_U)$ since this allows for flexibility in representing prior uncertainty. The beta density is given by

$$g(\rho \mid X^{(0)}) = \frac{\Gamma(c+d)}{\Gamma(c) \Gamma(d)} \frac{(\rho - \rho_L)^{c-1}(\rho_U - \rho)^{d-1}}{(\rho_U - \rho_L)^{c+d-1}} \quad \text{for } 0 \leq \rho_L \leq \rho \leq \rho_U$$

(3.7)

where $\rho_L, \rho_U, c, d > 0$ are specified parameters. We define our distribution for $\rho$ as
\[ p(\rho | X^{(0)}) = \Pr\left\{ \rho = \rho^{L} | X^{(0)} \right\} = \int_{\rho^{L-\delta}^{-\delta}}^{\rho^{L+\delta}} g(\rho) \, d\rho \quad (3.8) \]

where \( \rho^{L} = \rho^{L} + \frac{2\ell-1}{2} \delta \) and \( \delta = \frac{\theta_{\ell} - \rho^{L}}{\kappa} \) for \( \ell = 1, \ldots, k \). Prior to any testing, we assume that, \( \theta_{0} \) and \( \rho \) are independent. After \( i \) stages of testing, given \( X_{i} = x_{i} \) is observed, the posterior distribution of \( \rho \) is obtained via the standard Bayesian machinery as

\[ p(\rho_{i} | X^{(0)}) \propto p(\rho_{i} | X^{(0-1)}) \, p(x_{i} | \rho_{i}, X^{(0-1)}) \quad i=1,2, \ldots, k, \quad (3.9) \]

where the likelihood term \( p(x_{i} | \rho_{i}, X^{(0-1)}) \) is the predictive density given by (3.6). Once the posterior distribution (3.9) is available, the unconditional posterior distribution of \( \theta_{i} \) can be obtained by averaging out (3.5) with respect to this posterior distribution.

One of the attractive features of the presented model is the existence of all the predictive moments of \( X_{i+m} \) and \( \theta_{i+m} \) given \( X^{(0)} \) analytically, for \( m > 0 \). As shown by Smith and Miller (1986), assuming \( s < \chi^{2}_{n-1} \), for \( n \geq i+1 \),

\[ E(\theta_{i+m}^{-s} | X^{(0)}) = \frac{(\beta_{i})^{s}}{\Gamma(\alpha_{i})} \prod_{n=i+1}^{i+m} (\rho)^{-s} \frac{\Gamma(\alpha_{n-1})\Gamma(\chi^{2}_{n-1} - s)}{\Gamma(\alpha_{n-1} - s)\Gamma(\chi^{2}_{n-1})} \quad (3.10) \]

and similarly

\[ E(X_{i+m}^{-s} | X^{(0)}) = \Gamma(s + 1) \, E(\theta_{i+m}^{-s} | X^{(0)}) \quad (3.11) \]

For example, from the above, the \( m \)-step ahead predictive means are obtained as

\[ E(\theta_{i+m} | X^{(0)}) = \frac{\alpha_{i}}{\beta_{i}} (\gamma \rho)^{m} \quad (3.12) \]

\[ E(X_{i+m} | X^{(0)}) = \frac{\beta_{i}}{(\alpha_{i} - 1)\rho^{m}} \prod_{n=i+1}^{i+m} \frac{\alpha_{n-1} - 1}{\Gamma(\chi^{2}_{n-1} - 1)} . \]

The availability of these moments enables us to obtain optimal stopping rules in Section 4. We note that the moments in (3.12) are conditional on the parameter \( \rho \). For example, the updating of
the parameter \( \beta_i \) is done conditional on \( \rho \). The unconditional \( m \)-step ahead predictive means can be obtained by averaging out (3.12) with respect to the posterior distribution of \( \rho \) given by (3.9), that is,

\[
E(\theta_{i+m} \mid X^{(i)}) = E_{\rho} \left( E(\theta_{i+m} \mid X^{(i)}, \rho) \right)
\]

(3.13)

\[
E(X_{i+m} \mid X^{(i)}) = E_{\rho} \left( E(X_{i+m} \mid X^{(i)}, \rho) \right).
\]

4. OPTIMALITY OF THE ONE-STAGE LOOK AHEAD RULE

It is reasonable to assume that \( L_T \) is an increasing function of \( X_j \) in (2.5). A simple but a reasonable candidate for the loss function is

\[
L_T(X_j) = k_T X_j,
\]

(4.1)

where \( k_T > 0 \) can be interpreted as the testing cost per unit execution time. In specifying \( L_S \), we note that an unreliable piece of software will yield a higher loss, and thus \( L_S \) should be an increasing function of the software failure rate at the time of release. In view of this we assume that

\[
L_S(\theta_{i+1}) = k_S \theta_{i+1},
\]

(4.2)

where \( k_S > 0 \).

For the Theorem 1 of Section 2 to be applicable, we need to show that \( E(X_j \mid X^{(i)}) \) is increasing in \( j = i+1, \ldots \), and \( E(\theta_j \mid X^{(i)}) \) is discrete convex in \( j = i+1, \ldots \). Using the model presented in Section 3, we can easily identify the conditions under which both of these requirements are satisfied.
Case 1: $\rho = 1$

If $\rho = 1$ then (3.1) implies that $\{\theta_i\}$ is a decreasing sequence in $i$ since $\xi_i$ takes values in $(0, 1)$. As we have previously noted, this is the case of reliability growth. It follows then from (3.10) that for $j \geq (i+1)$

$$E(X_j \mid X^{(0)}) = \frac{\beta_i}{(\alpha_i - 1)} \prod_{n=i+1}^{j} \frac{(\alpha_n - 1)}{(\gamma \alpha_{n-1} - 1)},$$

(4.3)

and since the term $\frac{(\alpha_{n-1} - 1)}{(\gamma \alpha_{n-1} - 1)} > 1$ in the product, $E(X_j \mid X^{(0)})$ is increasing in $j$ for $j = i+1, \ldots$ with the restriction that $\gamma \alpha_{n-1} > 1$ for $n \geq i + 1$.

To show that $E(\theta_j \mid X^{(i)})$ is discrete convex in $j \geq i + 1$, we can write from (3.10) that the first difference is given by

$$[E(\theta_j \mid X^{(0)}) - E(\theta_{j+1} \mid X^{(0)})] = \frac{\alpha_i}{\beta_i} \gamma^j (1 - \gamma)$$

(4.4)

for $j \geq i+1$. We note that the term on the right hand side is decreasing in $j$ since $0 < \gamma < 1$ and therefore $E(\theta_j \mid X^{(i)})$ is discrete convex in $j \geq i+1$. In this particular case, this assumption implies that software reliability improvement diminishes during the testing/modification process. This is expected in software testing, due to the fact that the more major software bugs will be discovered and fixed during the earlier stages of the process. Since both conditions are satisfied, Theorem 1 is applicable in this case and the one-stage look ahead rule is optimal.

Case 2: $\rho > 1$

When $\rho > 1$ (3.1) implies that failure rates may be increasing with $i$. Again using (3.12), we can write for $j \geq i + 1$

$$E(X_j \mid X^{(0)}) = \frac{\beta_i}{(\alpha_i - 1)} \prod_{n=i+1}^{j} \frac{1}{\rho} \frac{(\alpha_{n-1} - 1)}{(\gamma \alpha_{n-1} - 1)}.$$  

(4.5)

We note that if the term
\[
\frac{1}{\rho} \frac{(\alpha_{n-1} - 1)}{(\gamma \alpha_{n-1} - 1)} = \frac{(\alpha_{n-1} - 1)}{(\rho \gamma \alpha_{n-1} - \rho)} > 1
\]

in the product, then \(E(X_j \mid X^{(i)})\) is increasing in \(j\) for \(j \geq i + 1\), with the restriction that \(\gamma \alpha_{n-1} > 1\) for \(n \geq i + 1\). We note that the condition \(\gamma \rho < 1\) is sufficient for this ratio to be greater than unity in all cases and guaranteeing that \(E(X_j \mid X^{(i)})\) is increasing in \(j\) for \(j \geq i + 1\).

To find out under what conditions \(E(\theta_j \mid X^{(i)})\) is discrete convex in \(j\) for \(j \geq i + 1\), we can write from (3.10) that the first difference is

\[
[E(\theta_j \mid X^{(i)}) - E(\theta_{j+1} \mid X^{(i)})] = \frac{\alpha_i}{\beta_i} (\rho \gamma)^j (1 - \rho \gamma).
\] (4.6)

It can be easily seen that the right hand side will be decreasing in \(j\) if \(\gamma \rho < 1\) as in the previous case. Thus, \(\gamma \rho < 1\) is sufficient for the Theorem 1 to be applicable in the case of \(\rho > 1\) and when this condition is satisfied the one stage look ahead rule is still optimal. From (3.1) we note that \(E(\frac{\theta_i}{\hat{\theta}_{i-1}} \mid \theta_{i-1}) = \gamma \rho\) and the condition \(\gamma \rho < 1\) implies that we expect the failure rate to be decreasing from one stage to another. Thus, the one-stage look ahead rule is optimal if the failure rates are stochastically decreasing or equivalently if it is expected that the reliability will be improving.

It is important to note that for the predictive means in (4.5) to exist we need to have \(\gamma \alpha_{n-1} > 1\) for \(n \geq i + 1\). As \(n\) gets large it can be shown that \(\alpha_n\) goes to the steady state value \(\alpha = \frac{1}{1 - \gamma}\) and this implies that for the above condition to be satisfied we need \(\frac{\gamma}{1 - \gamma} > 1\) or equivalently \(\gamma > 0.5\).

As we have previously noted it is reasonable to treat \(\rho\) as unknown and describe uncertainty about \(\rho\) probabilistically as in (3.8). By making inference on \(\rho\) at any stage of testing we can also assess the optimality of the one stage look ahead rule by using probability statements. If the optimality of the rule is assessed with high probability then we only need to compare \(L^{(0)}_q\) with \(L^{(1)}_q\) to determine whether to stop or not as required by Theorem 1.

Using the results of Section 3 and the specific form of loss functions given by (4.1) and (4.2), conditional on \(\rho\), we can obtain the additional expected loss associated with testing for \(\delta\) more stages after the \(i\)-th modification to the software as
\[ L_i^{(0)} = \left\{ \sum_{j=1}^{i} k_T \frac{\beta_i}{(\alpha_i - 1)\rho^j} \prod_{n=i+1}^{i+j} \frac{(\alpha_{n-1} - 1)}{(\gamma\alpha_n - 1)} \right\} + \left\{ k_S \frac{\alpha_i}{\beta_i} (\gamma \rho)^{i+1} \right\}. \] (4.7)

It follows from (4.7) that

\[ L_i^{(0)} = \left\{ k_S \frac{\alpha_i}{\beta_i} (\gamma \rho) \right\} \]

(4.8)

\[ L_i^{(1)} = \left\{ k_T \frac{\beta_i}{(\gamma\alpha_i - 1)} + k_S \frac{\alpha_i}{\beta_i} (\gamma \rho)^2 \right\}. \]

By using the posterior distribution of \( \rho \) given by (3.9), we can average out (4.7) and obtain \( L_i^{(0)} \) unconditional on \( \rho \).

5. AN ILLUSTRATION OF STOPPING RULES

Assume that a particular piece of software will go through at most 10 stages of testing/modification and after each stage, following the modifications made to the software, a decision is made on whether or not to terminate the testing process.

Part 1: \( \rho \) is known. For illustrative purposes, assume that, prior to any testing, uncertainty about software's performance is expressed by the model given in (3.1) with \( \rho=1 \), \( \gamma=0.8 \) and the prior parameters \( \alpha_0 = \beta_0 = 2 \). Thus, our model implies that the failure rates will be decreasing with additional stages of testing. This assumption will be dropped in the second part of our analysis. We also assume that \( k_T = 1 \) and \( k_S = 100,000 \) in (4.1) and (4.2) implying a relatively high loss of releasing an unreliable piece of software. We note that, as shown in Section 4, when \( \rho = 1 \) the one-stage look ahead rule is optimal.

Based on the prior information alone \( L_0^{(0)} \) can be obtained using (4.7). These values are presented in the first row of Table 1 for \( \delta = 0, 1, \ldots, 10 \). Using the stopping rule (2.6),
\( L_0^{(1)} - L_0^{(0)} < 0 \) and thus the optimal decision is to initiate testing. Furthermore, using (2.4), \( L_0 = L_0^{(10)} \) which implies that testing is expected to terminate after 10 stages.

Suppose that software has failed after 3 units of time during the first testing stage. After the completion of the first stage uncertainty about the software failure rate is revised using the results of Section 3 and the expected additional loss is calculated using (4.7). This is given in the second row of Table 1. Again as \( L_1^{(1)} - L_1^{(0)} < 0 \), the optimal decision is to continue testing. Since \( L_1^* = L_1^{(9)} \), the testing is still expected to be completed after the tenth stage.

### Table 1

Expected Additional Loss Estimates after Each Stage.

<table>
<thead>
<tr>
<th>STAGE j</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_0 )</td>
<td>80,000</td>
<td>64,300</td>
<td>51,200</td>
<td>40,900</td>
<td>32,800</td>
<td>26,200</td>
<td>21,000</td>
<td>16,800</td>
<td>13,500</td>
<td>10,800</td>
<td>8,700</td>
</tr>
<tr>
<td>( L_1^{(1)} )</td>
<td>-</td>
<td>41,600</td>
<td>33,300</td>
<td>26,600</td>
<td>21,300</td>
<td>17,100</td>
<td>13,700</td>
<td>10,900</td>
<td>8,800</td>
<td>7,100</td>
<td>5,700</td>
</tr>
<tr>
<td>( L_2^{(2)} )</td>
<td>-</td>
<td>-</td>
<td>7,100</td>
<td>5,700</td>
<td>4,600</td>
<td>3,700</td>
<td>3,000</td>
<td>2,500</td>
<td>2,100</td>
<td>1,800</td>
<td>1,700</td>
</tr>
<tr>
<td>( L_3^{(3)} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1,800</td>
<td>1,600</td>
<td>1,400</td>
<td>1,300</td>
<td>1,200</td>
<td>1,300</td>
<td>1,400</td>
<td>1,700</td>
</tr>
<tr>
<td>( L_4^{(4)} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1,300</td>
<td>1,200</td>
<td>1,100</td>
<td>1,000</td>
<td>1,100</td>
<td>1,300</td>
<td>1,500</td>
</tr>
<tr>
<td>( L_5^{(5)} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>930</td>
<td>900</td>
<td>910</td>
<td>990</td>
<td>1,200</td>
<td>1,500</td>
</tr>
<tr>
<td>( L_6^{(6)} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>950</td>
<td>910</td>
<td>900</td>
<td>990</td>
<td>1,200</td>
</tr>
<tr>
<td>( L_7^{(7)} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>980</td>
<td>930</td>
<td>910</td>
<td>980</td>
</tr>
<tr>
<td>( L_8^{(8)} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>810*</td>
<td>820</td>
<td>860</td>
</tr>
</tbody>
</table>

The observed life-lengths of the software at the subsequent testing stages are given in Table 2 below and the corresponding expected additional losses, \( L_i^{(i-1)} \), are given in Table 1. Note that the optimal decision rule is to stop testing the first time \( L_4^{(1)} > L_4^{(0)} \). We see from Table 1, that the optimal decision after testing stages 2 through 7 is to continue testing. It is not until the 8th stage that \( L_8^{(1)} > L_8^{(0)} \) implying that the optimal decision is to stop testing. In Figure 2, we illustrate the posterior means of the failure rates, \( \theta_i \)'s after each stage of testing. The downward trend in \( \theta_i \)'s reflects the improvement in the performance of the software as implied by the model. The sequential nature of the inference and decision making during the testing/modification process is shown in Figure 3 where we present the plots of expected additional loss after 0, 3, 6, and 8
TABLE 2
Actual Life-length of Software Tested in Stages \( j = 1, \ldots, 8. \)

<table>
<thead>
<tr>
<th>TESTING STAGE ( j )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_j )</td>
<td>3</td>
<td>30</td>
<td>113</td>
<td>81</td>
<td>9</td>
<td>2</td>
<td>91</td>
<td>112</td>
</tr>
</tbody>
</table>

stages of testing. These curves are obtained from the rows 1, 4, 7, and 9 of Table 1. The plot shows how the expected duration of the testing changes with additional observations. For example, apriori, testing is expected to terminate after the 10th stage. After three stages of testing, the expected termination is revised to the 7th stage and after six stages of testing it is revised to the 8th stage.

**Part 2: \( \rho \) is unknown.** We now assume that \( \rho \) is unknown in (3.1) and we describe our uncertainty about \( \rho \) apriori via the discrete beta density given by (3.7). Again for illustrative purposes, we specify a 100-point prior with \( \rho_L = 1, \rho_U = 2, c = 1.25 \) and \( d = 5 \) in (3.7). Such a choice reflects the prior belief of expected improvement in software's performance. As in part 1 of our analysis, we choose \( \gamma = 0.8, \alpha_0 = \beta_0 = 2 \) and assume that \( k_T = 1 \) and \( k_S = 100,000 \). We note that such a choice of prior parameters does not guarantee that the one-stage look ahead rule is optimal since \( \gamma \rho > 1 \) for \( \rho > 1.25 \).

In Figure 4 we present the plots of expected additional loss after 0, 2, 3, and 5 stages of testing. In this particular case it is optimal to stop after 5 stages of testing. We can see from the plot that the one-stage look ahead rule is optimal in all the four cases. As in part 1 of the analysis this plot shows how the expected duration of the testing changes with additional observations. For example, apriori we expect to terminate the testing after two stages. After the 2nd stage, we revise the expected termination to stage 7 and after the 3rd stage we revise it to stage 5 which was the optimal stage to terminate testing.

Figure 5 shows the prior distribution and the posterior distributions of \( \rho \) after 2, 3, and 5 stages of testing. We can see from the figure that the posterior mass shifts more to the left (closer
to 1) with additional observations implying an expected improvement in software. We can compute \( \Pr\{\rho > 1.25 \mid X^{(i)}\} \), the probability that \( \rho > 1.25 \) after \( i \)-stages of testing, for any stage \( i \). For example, with the particular values of prior parameters, we obtain \( \Pr\{\rho \leq 1.25 \mid X^{(5)}\} = 0.915 \) reinforcing the optimality of the one-stage look ahead rule. This can also be observed from the additional loss plot associated with Stage 5 in Figure 4.

![Figure 2: Posterior Means of \( \theta_i \)'s.](image)

![Figure 3: Additional Loss of Testing after Stages 0, 3, 6, and 8.](image)
Figure 4: Additional Loss of Testing after Stages 0, 2, 3, and 5.

Figure 5: Prior and Posterior Distributions of $\rho$ after Testing Stages 0, 2, 3, and 5.
REFERENCES


