Interval Estimations of Software Reliability and Optimal Release Time
Based on Better Bootstrap Confidence Intervals

Shinji Inoue and Shigeru Yamada
Graduate School of Engineering, Tottori University, Japan
{ino,yamada}@sse.tottori-u.ac.jp

Abstract—We discuss a bootstrap method for estimating software reliability and cost-optimal software release time by using a discretized software reliability growth model. And then, we conduct interval estimations of them by five types of bootstrap confidence interval estimation methods including so-called better bootstrap confidence intervals, such as a bootstrap-t and a bias-corrected and the accelerated confidence interval (BCa) method. In our numerical examples, we confirm that our bootstrap approach yields simulation-based probability distributions of model parameters, software reliability assessment measures, and cost-optimal software release time without deriving these probability distributions analytically.


I. INTRODUCTION

A software reliability growth model [10], [11], [16] is one of the fundamental technologies for quantitative software reliability assessment. The software reliability growth model describes a software failure-occurrence or fault-detection phenomenon observed in testing and operational phases by using stochastic modeling approaches and reflecting actual debugging environment in the testing and operational phases. As interesting topics being related to the software reliability growth model, optimal software release problems [14] are widely discussed for estimating optimal software shipping time in terms of total software cost, which consists of the testing and maintenance costs. In actual software testing and operational phases, there exists a trade-off relationship between the testing cost for debugging software faults in the testing-phase and the maintenance cost for software failure occurred in the operational phase. Therefore, estimating optimal software shipping (or release) time is of interest for software development project managers.

Most of software reliability and optimal software release time estimation approaches use a point estimation method. Considering a practical situation, we encourage the software development managers to use the interval estimation method when we do not obtain a sufficient number of software reliability data (sampling data). However, the interval estimation needs to derive a probability distribution function for the parameter of interest. Especially for software reliability assessment measures and the optimal software release time, it is very difficult to derive the probability distribution functions analytically even if we use the approximation approach assuming a large number of samples. And, it is not easy to derive some useful information for the statistical inference on these measures. Under such background, Kimura and Fujicawa [9] discussed a bootstrap software reliability assessment method of an incomplete gamma function-based software reliability growth model for estimating standard errors of the model parameters. Kaneishi and Dohi [8] discussed a parametric bootstrap method for software reliability assessment based on continuous-time nonhomogeneous Poisson process (NHPP) models.

We discuss a different simulation-based software reliability assessment method of the NHPP model based on a non-parametric bootstrap method via a discretized NHPP model. The discretized NHPP model conserves the basic property of the continuous-time NHPP model and have good prediction and fitting performance for the actual data [6] because the discretized model has consistency with discrete fault count data collection activities. And, we discuss five types of bootstrap confidence intervals including so-called better bootstrap confidence interval estimation methods, such as a bootstrap-t confidence interval and a bias-corrected and the accelerated (BCa) confidence interval, for interval estimations of the software reliability assessment measures and cost-optimal software release time with the asymmetric property, bias, and skewness of the probability distributions. Finally, we show numerical examples of our bootstrap approach in this paper by using actual fault-count data, and show results of interval estimations for the software reliability assessment measures and the cost-optimal software release time based on the notion of the bootstrap confidence intervals.

II. DISCRETIZED NHPP MODEL

We briefly discuss the aspect of the discretized NHPP model [6], which is going to be used in our nonparametric bootstrap method in this paper for interval estimations of the software reliability measurement and cost-optimal software release time. Now we define a discrete counting process \( \{N_n, n = 0, 1, 2, \ldots\} \) representing the cumulative number of faults detected up to \( n \)-th testing-period. And we can say that the discrete counting process \( \{N_n, n = 0, 1, 2, \ldots\} \) follows a discrete-time NHPP [5], [15], which is the discrete analog of the continuous-time NHPP [13], if the discrete counting
have observed fault counting data procedure using the method of least-squares. Suppose we
\( \omega \) of the corresponding continuous-time exponential software reliability growth model. Solving the above inter-

discretized exponential software reliability growth model. The
can be obtained as

Let \( H_n \) denote a mean value function following a discretized exponential software reliability growth model. The discretized exponential software reliability growth model is derived from the following difference equation:

\[
H_{n+1} - H_n = \delta \beta (\omega - H_n),
\]

which is the discrete analog of the differential equation of the continuous-time exponential software reliability growth model [3]. In Eq. (2), \( \omega \) is the expected total number of potential faults to be detected in an infinitely long duration or the expected initial fault content, and \( \beta \) the fault detection rate per one fault. Regarding the discretization method, we use the Hirota’s bilinearization methods [4] for conserving the property of the continuous-time exponential software reliability growth model. Solving the above integrable difference equation in Eq. (2), we can obtain an exact solution \( H_n \) in Eq. (2) as

\[
\Lambda_n \equiv H_n = \omega \left[ 1 - (1 - \delta \beta)^n \right] \quad (\omega > 0, \ \beta > 0),
\]

where \( \delta \) represents the constant time-interval. As \( \delta \to 0 \), Eq. (3) converges to the exact solution of the original continuous-time exponential software reliability growth model.

The discretized exponential software reliability growth model in Eq. (3) has two parameters, \( \omega \) and \( \beta \), which have to be estimated by using actual data. The parameter estimations of \( \omega \) and \( \beta \), \( \hat{\omega} \) and \( \hat{\beta} \), can be obtained by the following procedure using the method of least-squares. Suppose we have observed fault counting data \( (n, y_n) (n = 1, 2, \ldots, N) \), where \( y_n \) represents the cumulative number of faults detected up to \( n \)-th testing-period. We can derive the following regression equation from Eq. (2):

\[
C_n = \alpha_0 + \alpha_1 D_n,
\]

where

\[
\begin{align*}
C_n &= H_{n+1} - H_n \equiv y_{n+1} - y_n, \\
D_n &= H_n \equiv y_n, \\
\alpha_0 &= \delta \omega \hat{\beta}, \\
\alpha_1 &= -\delta \beta.
\end{align*}
\]

Based on the regression analysis, we can estimate \( \hat{\alpha}_0 \) and \( \hat{\alpha}_1 \), which are the estimations of \( \alpha_0 \) and \( \alpha_1 \) in Eq. (4). Then, the parameter estimations, \( \hat{\omega} \) and \( \hat{\beta} \), can be obtained as

\[
\begin{align*}
\hat{\omega} &= -\hat{\alpha}_0/\hat{\alpha}_1, \\
\hat{\beta} &= -\hat{\alpha}_1.
\end{align*}
\]

respectively. \( C_n \) in Eq. (4) is independent of \( \delta \) because \( \delta \) is not used in calculating \( C_n \) as showing Eq. (5). Hence, we can obtain the same parameter estimates \( \hat{\omega} \) and \( \hat{\beta} \), respectively, when we choose any constant value of \( \delta \) [6].

### B. Discretized Reliability Assessment Measures

The discrete version of the expected number of remaining faults, \( M_n \), represents the expected number of undetected faults in the software system at arbitrary testing-period. Then, we have

\[
M_n \equiv E[N_\infty - N_n] = \omega - \Lambda_n = \omega (1 - \delta \beta)^n
\]

if we assume that \( N_n \) follows a discrete-time NHPP with mean value function \( H_n \) in Eq. (3). And the discrete-time software reliability function, \( R(n, h) \), is defined as the probability that a software failure does not occur in the time-interval \( (n, n + h) \) \((h = 1, 2, \ldots)\) given that the testing has been going up to the \( n \)-th testing-period. Then, we have

\[
R(n, h) \equiv \Pr\{N_{n+h} - N_n = 0 \mid N_n = x\} = \exp[- \{\Lambda_{n+h} - \Lambda_n\}] = \exp[-H_n (1 - \delta \beta)^n],
\]

### C. Cost-Optimal Software Release Time

The cost-optimal software release time means testing-termination time minimizing the total software cost formulated by a software reliability growth model. For formulating the total software cost, we define the following cost parameters:

\( c_1 \) : debugging cost per one fault in the testing-phase,

\( c_2 \) : debugging cost per one fault in the operational phase, where \( c_1 < c_2 \),

\( c_3 \) : testing cost per unit time.

Suppose that \( Z \) is the testing termination time or the software release time and \( \Lambda_{Z, LC} \) the length of the software life cycle measured from the test beginning. By using the cost parameters, we can formulate the expected total software cost \( C_Z \) as

\[
C_Z = c_1 \Lambda_Z + c_2 \left( \Lambda_{Z, LC} - \Lambda_Z \right) + c_3 Z = (c_1 - c_2) \Lambda_Z + c_2 \Lambda_{Z, LC} + c_3 Z.
\]

From Eq. (9), the cost-optimal software release time \( Z^* \) [7] is derived as

\[
Z^* = \log \left[ \frac{c_3}{(c_2 - c_1) \delta \omega \beta} \right].
\]

We should note that the discretized exponential model and software reliability assessment measures converges to the continuous-time versions as the constant time-interval tends to zero, and essentially same properties because

\[
\lim_{\delta \to 0} (1 - \delta \beta)^n = \lim_{n \to \infty} \left(1 - \frac{\beta t}{n}\right)^n = e^{-\beta t},
\]

\[
(1 - \delta \beta)^n = 1 - n\beta \delta + O(\delta^2) + \cdots,
\]

\[
e^{-\beta t} = e^{-\delta \beta n} = 1 - n\beta \delta + O(\delta^2) + \cdots,
\]

where \( t = n\delta \).
III. NONPARAMETRIC BOOTSTRAP METHOD

We apply a nonparametric bootstrap method [1] to conducting simulation-based interval estimations by using the discretized NHPP model. Our bootstrap method for software reliability measurement and estimating the optimal software release time follows the following procedure:

Step 1 Estimate $\alpha_0$ and $\alpha_1$ in Eq. (4) by following the linear regression scheme mentioned in Section II with observed data $(n, y_n)(n = 1, 2, \cdots, N)$. And we indicate $\hat{\alpha}_0$ and $\hat{\alpha}_1$ as $\hat{\alpha}_0(0)$ and $\hat{\alpha}_1(0)$, respectively.

Step 2 Calculate the residual errors, $\hat{d}_i$, at each observation point by

$$\hat{d}_i = C_i - (\hat{\alpha}_0(0) + \hat{\alpha}_1(0)D_i) \quad (i = 1, 2, \cdots, N - 1).$$

Step 3 Construct an empirical distribution function $\hat{F}$ by assuming the residual errors $\hat{d}_i$ follows the independent and identical probability distribution and by putting mass $1/(N - 1)$ at each ordered point $\{\hat{d}_1, \hat{d}_2, \cdots, \hat{d}_{N-1}\}$.

Step 4 Set the total number of iteration $B$ and let $b = 1, 2, \cdots, B$ be the iteration count.

Step 5 Generate a bootstrap sample for the residual errors, $\hat{d}_{i(b)} = \{\hat{d}_{i1(b)}, \hat{d}_{i2(b)}, \cdots, \hat{d}_{i(N-1)(b)}\}$, by sampling with replacement from $\hat{F}$.

Step 6 Generate a bootstrap sample, $\alpha_{*}^{(b)} = \{y_1, C_{11}^{*}, y_2, C_{12}^{*}, \cdots, y_{N-1}, C_{N-1}^{*}\}$, by

$$C_{bi}^{*} = \hat{\alpha}_0(0) + \hat{\alpha}_1(0)D_i + \hat{d}_{i(b)}^\ast.$$ 

Step 7 Estimate $\hat{\alpha}_0(b)$ and $\hat{\alpha}_1(b)$ by the following equations:

$$\hat{\alpha}_0^{(b)} = \frac{\hat{d}_{0(b)} - \hat{C}_{11}(b)}{\hat{C}_{11}(b)},$$

$$\hat{\alpha}_1^{(b)} = -\hat{\alpha}_0^{(b)}.$$

Step 9 Calculate a software reliability assessment measure $h(\hat{\tau}^{(b)}_*)$ and cost-optimal software release time $T^*(\hat{\tau}^{(b)}_*)$, where $\hat{\tau}^{(b)}_* = (\hat{\omega}^{(b)}, \hat{\beta}^{(b)}_*)$.

Step 10 Let $b = b + 1$ and go back to Step 5 if $b < B$.

Step 11 We have $B$ samples for $\hat{\omega}^{*}, \hat{\beta}^{*}, h(\hat{\tau}^{*})$, and $T^*(\hat{\tau}^{*})$.

Finally, we calculate the standard deviations of $\hat{\omega}$ and $\hat{\beta}$, a software reliability assessment measure $h(\hat{\tau}^{*})$, and the cost-optimal software release time $T^*(\hat{\tau}^{*})$, respectively.

IV. BOOTSTRAP CONFIDENCE INTERVALS

We discuss the following five types of bootstrap confidence intervals including so-called better bootstrap confidence intervals [2], [12]. Let $\theta$ be parameter of interest.

A basic bootstrap confidence interval is developed by using the quantile of the distribution of $\hat{\theta} - \theta$, where $\hat{\theta}$ is the bootstrap statistic. We can approximate the $\alpha$ and $(1 - \alpha)$ quantile denoting $v_\alpha$ and $v_{1-\alpha}$, respectively, of the distribution of $\hat{\theta} - \theta$ by $\hat{\theta}^{*}_{[B]} - \hat{\theta}$ and $\hat{\theta}^{*}_{[B(1-\alpha)]} - \hat{\theta}$. Then,

$$1 - 2\alpha = \Pr \left\{ v_\alpha \leq \hat{\theta} - \theta \leq v_{1-\alpha} \right\} = \Pr \left\{ \hat{\theta} - v_{1-\alpha} \leq \theta \leq \hat{\theta} - v_\alpha \right\} = \Pr \left\{ 2\hat{\theta} - \hat{\theta}^{*}_{[B(1-\alpha)]} \leq \theta \leq 2\hat{\theta} - \hat{\theta}^{*}_{[B]} \right\}.$$

Thus, the $100(1 - 2\alpha)$% basic bootstrap confidence interval is given by

$$[2\hat{\theta} - \hat{\theta}^{*}_{[B]}, 2\hat{\theta} - \hat{\theta}^{*}_{[B(1-\alpha)]}].$$

A standard normal bootstrap confidence interval is derived by assuming that the distribution of $\hat{\theta} - \theta$ can be approximated by the distribution of $\hat{\theta}^{*} - \theta$ and $\hat{\theta}^{*} - \theta \sim N(0, SD[\theta]^{2})$. That is,

$$1 - 2\alpha = \Pr \left\{ z_\alpha \leq \frac{\hat{\theta}^{*} - \hat{\theta}}{SD[\theta]} \leq z_{1-\alpha} \right\},$$

where $SD[\theta]$ represents the standard deviation of the parameter of interest. Thus, we have the $100(1 - 2\alpha)$% standard normal bootstrap confidence interval as

$$[\hat{\theta} - z_{1-\alpha}SD[\theta], \hat{\theta} - z_{\alpha}SD[\theta]],$$

where $z_\alpha = \Phi^{-1}(1 - 2\alpha)$. For example, $z_{0.025} = 1.96$.

A percentile bootstrap confidence interval is calculated from the empirical cumulative distribution function consisted by the bootstrap iteration value: $\hat{\theta}^{*}_{[1]}, \hat{\theta}^{*}_{[2]}, \cdots, \hat{\theta}^{*}_{[B]}$. The $100(1 - 2\alpha)$% percentile bootstrap confidence interval is calculated by

$$[\hat{\theta}^{*}_{[B]}, \hat{\theta}^{*}_{[B(1-\alpha)]}],$$

where $\hat{\theta}^{*}_{[B]}$ represents the $\alpha$ quantile of the empirical cumulative distribution.

A bootstrap-$t$ confidence interval enables us to take the effect of the variance of $\theta$ by using $T = (\hat{\theta} - \theta)/\hat{\sigma}$, where $\hat{\sigma}^2$ is the variance of $\hat{\theta}$. Letting $u_\alpha$ and $u_{1-\alpha}$ are the $\alpha$ and $(1 - \alpha)$ quantile of $T$, we have

$$1 - 2\alpha = \Pr \left\{ u_\alpha \leq \frac{\hat{\theta} - \theta}{\hat{\sigma}} \leq u_{1-\alpha} \right\} = \Pr \left\{ \hat{\sigma} - \hat{\sigma}u_{1-\alpha} \leq \theta \leq \hat{\sigma} - \hat{\sigma}u_\alpha \right\}.$$
the $100(1-2\alpha)\%$ bootstrap-t confidence interval is derived as

$$
\hat{\theta} - \hat{\sigma}^* T_{[B(1-\alpha)]}, \hat{\theta} - \hat{\sigma}^* T_{[B\alpha]}
$$

The parameter of interest. And $\alpha$ is the acceleration constant derived as

$$
\alpha = \frac{1}{6} \sum_{i=1}^{n} \left( \hat{\theta}(i) - \hat{\theta}(1) \right)^3
$$

where $\hat{\theta}(i)$ is a jackknife iteration value, which is estimated by using data removed the $i$th data, and $\hat{\theta}(\cdot) = \sum_{i=1}^{n} \hat{\theta}(i)/n$.

V. NUMERICAL EXAMPLES

We show numerical examples for our bootstrapping software reliability assessment method based on the discretized exponential software reliability growth model. We apply the following data: $(n, y_n) = (1, 2, \cdots, 25; y_{25} = 136)$ [6] and we set the total number of iteration $B = 2000$. Actually, it is known that we need to set the number of iteration $B = 1000 \sim 2000$ for obtaining a bootstrap distribution of a parameter of interest.

We first obtain $\hat{\omega}_{0(0)} = 15.8586$ and $\hat{\omega}_{1(0)} = -0.1133109$ by the linear regression scheme from the actual data. Following the procedure of our nonparametric bootstrapping method, we have 2000 bootstrap samples $(\hat{z}_{(1)}^{25}, \hat{z}_{(2)}^{25}, \cdots, \hat{z}_{(2000)}^{25})$. Needless to say, we obtain bootstrap samples for $\omega$ and $\beta$ as $(\hat{\tau}_{(1)}^{25}, \hat{\tau}_{(2)}^{25}, \cdots, \hat{\tau}_{(2000)}^{25})$. Then, we have $\hat{\omega}_{(b)}$ and $\hat{\beta}_{(b)}$ ($b = 1, 2, \cdots, 2000$) by following the procedure.

Table I shows the means and standard deviations of the estimates of $\hat{\omega}_{0}, \hat{\omega}_{1}, \hat{\omega}, \hat{\beta}, \hat{M}_{25}$, and $\hat{R}(25, 1)$, respectively, with results of point estimations for them. And Table II shows the results of interval estimations for the software reliability assessment measures. From Tables I and II, we can see that we assess software reliability pessimistically because

![Fig. 1. Bootstrap distribution of cost-optimal software release time, $\hat{T}^*$](image-url)
the means of the bootstrap samples are estimated pessimistically compared with the results of the point estimations, and that inappropriate confidence intervals on $M_{25}$ are obtained in the basic, standard normal, and bootstrap-t confidence intervals. Regarding the cost-optimal software release time, Tables III and IV show the quantities of the bootstrap distributions and the results of the interval estimations based on the bootstrap confidence intervals. For an example of the bootstrap distributions, Figure 1 shows the bootstrap distribution of the cost-optimal software release time in a case $c_1 = 1$, $c_2 = 2$, and $c_3 = 0.01$. We suppose that $c_1 = 1$ to consider the relative software costs in this paper. From Table III, we can say that means and standard deviations of the bootstrap distributions are getting large as the debugging cost per one faults in the operational phase takes large values. And from Table IV, we have the same consideration in Table III. From these results, it might be better to use the BCA method as the bootstrap confidence interval because the BCA method considers the bias and skewness of the bootstrap distribution.

VI. Conclusion

We discussed interval estimation methods for software reliability and cost-optimal software release time based on the bootstrap confidence intervals including so-called better bootstrap confidence intervals, such as a bootstrap-t confidence interval and a bias-corrected and the accelerated (BCa) confidence interval, which enable us to obtain bootstrap confidence interval considering with the asymmetric property, bias, and skewness of the probability distribution. And we confirmed that our bootstrap approach yielded a simulation-based probability distribution of the software reliability assessment measures and the cost-optimal software release time without deriving these probability distributions analytically. Our interval estimation methods for the optimal software release time in this paper is very useful for planning software release time and quantifying the risk of mismatched estimation. Further, our nonparametric bootstrap method via a discretized software reliability growth model does not need to give a underlying probability distribution function for the sampling data when we generate a bootstrap sample. In our further studies, we are going to apply our bootstrap method and the bootstrap confidence interval estimation methods to estimating cost-reliability-optimal software release time [14] and other practical software project management issues.

ACKNOWLEDGMENT

This research was partially supported by the Grant-in-Aid for Scientific Research (C), Grant No. 25330081 and 25350445, from the Ministry of Education, Culture, Sports, Science and Technology of Japan and The Telecommunications Advancement Foundation.

REFERENCES