Software Reliability Modeling and Evaluation under Incomplete Knowledge on Fault Distribution

Toshio Kaneishi and Tadashi DOHI
Department of Information Engineering, Graduate School of Engineering
Hiroshima University, 1-4-1 Kagamiyama, Higashi-Hiroshima, 739-8527 Japan
Email: dohi@rel.hiroshima-u.ac.jp
Telephone: (81) 82-424-7698, Fax: (81) 82-424-7025

Abstract—**In this paper we consider non-parametric estimation methods for software reliability assessment without specifying the fault distribution, where the underlying stochastic process to describe software fault-counts in the system testing is given by a non-homogeneous Poisson process. A comprehensive approach based on the kernel estimation is provided with several kernel functions and bandwidth estimations. Next, we develop interval estimation methods via the non-parametric bootstrap, and derive the confidence regions of several reliability measures such as the expected cumulative number of software faults, software intensity function, quantitative software reliability as well. The resulting data-driven methodology can give the useful probabilistic information on the software reliability prediction under the incomplete knowledge on fault distribution. In illustrative examples with a real software fault data, it is shown that the proposed methods provide useful software reliability measures under uncertainty from the view point of frequentist analysis.**

Index Terms—**software reliability, non-homogeneous Poisson processes, non-parametric estimation, kernel-based method, bootstrap, confidence region**

I. Introduction

Software reliability models (SRMs) are used to assess software reliability and to control quantitatively software testing [14], [16]. Since the quantitative software reliability is defined as the probability that software failures caused by faults do not occur for a given period of time, the probabilistic behavior of fault-detection processes in the testing phase is modeled by any stochastic point process to estimate the software reliability. During the last four decades, SRMs based on non-homogeneous Poisson processes (NHPPs) have gained much popularity for describing the stochastic behavior of the number of detected faults, because of their tractability and goodness-of-fit performance. Goel and Okumoto [8], Musa and Okumoto [17], Zhao and Xie [26], among others extensively develop representative NHPP-based SRMs. Though almost all the NHPP-based SRMs are classified into parametric models and derived from simple debugging scenarios on the software intensity functions, the lesson learned from a huge number of empirical studies reported during the last four decades suggests that the best parametric SRM does not exist, which can fit every type of software fault data. This fact implies that the utility to select the best parametric form may be somewhat limited in the software reliability prediction.

On the other hand, some authors challenge to develop the so-called *non-parametric* SRMs without specifying the software intensity function or the mean value function. Miller and Sofer [15] propose a piecewise linear function estimator with breakpoints for the mean value function, and define its slope as an estimate of the software intensity function. Gandy and Jensen [7] use the well-known *Aalen estimator* to estimate the software intensity function, though their estimator is statistically consistent but not feasible in practice, because it must be based on the multiple time-series samples of software fault-detection time data. Barghout *et al.* [1] also apply the kernel-based technique to represent the software fault-detection time distribution for a generalized order statistics-based SRM, where the likelihood cross validation and the prequential likelihood approaches are used to estimate the bandwidth. Wang *et al.* [21] try the similar kernel method for an NHPP-based SRM, where they focus on an approximate method with a local weighted log-likelihood function. Xiao and Dohi [22],[23] develop the wavelet shrinkage estimations without the detailed parametric form of the software intensity function. Recently, Dharmasena *et al.* [6] treat the non-parametric regression-based SRMs with kernel functions based on local polynomial modeling, though they focus on only the point estimation.

It should be worth noting that the above works focus on only the point estimation of the software intensity function, but that a comprehensive study to compare these non-parametric methods has not been done yet in the past literature. In this paper we consider the kernel-based technique [2],[5] and investigate the dependence of the kind of kernel functions and their estimation methods. More precisely, we consider two types of SRMs; Model 1 and Model 2. Model 1 is referred to as a kernel intensity model or an infinite failure model whose intensity function is given by a kernel function. Model 2 is referred to as a kernel density function or a finite failure model whose intensity function is given by a product of the kernel probability density and a given parameter indicating the mean initial number of software faults before the system test. For these two SRMs, we apply four representative kernel functions; triangle function, quadratic function, triweigh function and Gauss function, and two bandwidth estimation methods; least-squares cross-validation (LSCV) [2],[5] and the log-likelihood cross-validation (LLCV) [1],[9]. In addition, we introduce an approximate method called the local likelihood method in [21] with application to only Model 1 (but not

Model 2). Further we predict the future behavior of the software intensity function with a weighted kernel function proposed by [3]. Hence, our models are somewhat similar to Dharmasena *et al.* [6] in terms of the kernel-based estimation, but quite different from it from several points of view on modeling and parameter estimation.

Next we concern the interval estimation of the software intensity function and its related software reliability measures such as the mean value function and software reliability as the probability that the software does not fail after the release. Yamada and Osaki [24], Joe [10], van Pul [19], Zhao and Xie [26] give asymptotically approximate confidence intervals of model parameters for specific NHPP-based SRMs. Yin and Trivedi [25] obtain an exact confidence interval of model parameters for the exponential NHPP-based SRM. Okamura *et al.* [18] develop the variational approach to approximate the confidence interval of model parameters from the both viewpoints of frequentist and Bayesian approaches. In this way, considerable attentions have been paid to the interval estimation of model parameters. However, these approach is rather limited if one is interested in the confidence region of arbitrary software reliability measures. For instance, when we consider the confidence interval of the mean value function, it does not mean the confidence region of NHPP itself. It is worth mentioning that the probability distribution of an arbitrary estimator has to be evaluated based on the underlying software fault data. In other words, it is impossible to give the confidence regions of software reliability measures themselves, such as software intensity function, mean value function and software reliability function, by means of the analytical (approximate) techniques employed in the past work.

The statistical bootstrap is a combination of data re-sampling and replication of estimation, and enables us to estimate the probability distribution of arbitrary estimators under interest, if they exist. To our best knowledge, van Pul [20] is the first work to apply bootstrap methods to a specific SRM. They utilize a parametric bootstrapping technique to the well-known Jelinski and Moranda SRM [14], [16]. Apart from the software reliability estimation, Lei and Smith [12] apply non-parametric bootstrap methods to estimate confidence intervals for software metrics. Very recently, Kaneishi and Dohi [11] apply parametric bootstrapping to get the probability distributions of estimators for some software reliability measures, provided that the mean value functions for NHPPs are given. This idea developed in [11] can be easily applied to the non-parametric bootstrapping for NHPP-based SRMs. In this paper, we also apply the different non-parametric bootstrap methods from Lei and Smith [12] to estimate the software reliability measures, where three bootstrap methods proposed by Cowling *et al.* [4] are used for analysis under four kernel functions and two bandwidth estimation methods. We estimate 95% confidence intervals of software intensity function, the mean value function and software reliability as well as their associated probability distributions. The results are useful to predict the software reliability measures which have not been studied in the past.

TABLE I
REPRESENTATIVE PARAMETRIC SRMS.

EXP	Exponential
GMA	Gamma
TRN	Truncated normal
LTR	Lognormal
TLG	Truncated logistic
LLG	Log logistic
TXA	Truncated extreme (maximum)
LXA	Logarithmic extreme (maximum)
TXI	Truncated extreme (minimum)
LXI	Logarithmic extreme (minimum)

II. SOFTWARE RELIABILITY MODELING

A. NHPP-based SRMs

Suppose that the system test of software starts at time $t = 0$. Let $\{N(t), t \geq 0\}$ be the cumulative number of software faults detected by time t and be a stochastic (non-increasing) point process. In particular it is said that $N(t)$ is a non-homogeneous Poisson process (NHPP) if the following conditions hold:

- $N(0) = 0$,
- $\{N(t), t \geq 0\}$ has independent increments,
- $\Pr\{N(t + \Delta t) - N(t) \geq 2\} = o(\Delta t)$,
- $\Pr\{N(t + \Delta t) - N(t) = 1\} = \lambda(t)\Delta t + o(\Delta t)$,

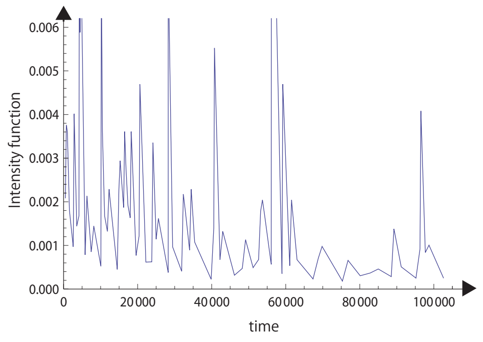
where $\lambda(t)$ is the intensity function of an NHPP, and $o(\Delta t)$ is the higher term of Δt. Then the probability mass function (p.m.f.) of the NHPP is given by

$$\Pr\{N(t) = n\} = \frac{\{\Lambda(t)\}^n}{n!} \exp\{-\Lambda(t)\}, \quad (1)$$

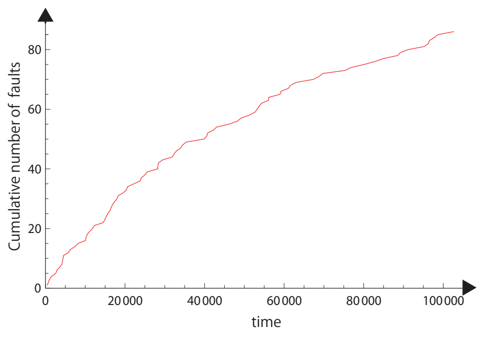
$$\Lambda(t) = \int_0^t \lambda(x)dx, \quad (2)$$

where the function $\Lambda(t) = \mathrm{E}[N(t)]$ is called the mean value function and denotes the expected cumulative number of software faults detected by time t. Hence, the intensity function is regarded as the expected number of software faults per unit time at time t.

Suppose that n software faults are detected by time t and that the fault-detection times, which are the random variables, are given by $0 < T_1 \leq T_2 \leq \ldots \leq T_n$. Given the realizations of T_i $(i = 1, 2, \ldots, n)$, t_i, we can regard the pair (i, t_i) as a realization of the underlying NHPP, $N(t)$. Without any loss of generality, we define the random variable $X_i = T_i/T_n \in (0, 1]$ with realizations $x_i = t_i/t_n \in (0, 1]$ for $i = 1,\ 2,\ \ldots,\ n$. If the parametric form of an NHPP is known with a specified mean value function, the estimation of an NHPP is reduced to an estimation of model parameters. Let $\boldsymbol{\theta}$ be the model parameters (vector) involved in the intensity function, so that we re-express the mean value and the intensity functions by $\Lambda(t; \boldsymbol{\theta})$ and $\lambda(t; \boldsymbol{\theta})$, respectively. For the scaled fault-detection time data x_i $(i = 1, 2, \ldots, n)$, the likelihood function is given



(i) Intensity function.



(ii) Mean value function.

Fig. 1. Behavior of estimates of software intensity and mean value functions with piecewise-linear interpolation estimator.

by

$$L(\boldsymbol{\theta}) = \exp\{-\Lambda(x_n;\boldsymbol{\theta})\}\prod_{i=1}^{n}\lambda(x_i;\boldsymbol{\theta}). \tag{3}$$

As described in Section I, the NHPP-based SRMs can be classified into two types; Model 1 and Model 2. In Model 1, the mean value function is arbitrary and may be unbounded, *i.e.*, $\lim_{t\to\infty}\Lambda(t;\boldsymbol{\theta}) \to \infty$ with finite $\boldsymbol{\theta}$. On the other hand, it is assumed that the mean value function in Model 2 is bounded, so $\lim_{t\to\infty}\Lambda(t;\boldsymbol{\theta}) \to \omega$, where $\omega \in \boldsymbol{\theta}$ is the mean residual number of software faults remaining in a software system before the system testing. Then it is common to assume $\Lambda(t;\boldsymbol{\theta}) = \omega F(t;\boldsymbol{\alpha})$, where $F(t;\boldsymbol{\alpha})$ is an arbitrary cumulative distribution function (c.d.f.) with the support $(0,1)$ having the probability density function (p.d.f.) $f(t;\boldsymbol{\alpha})$ and $\boldsymbol{\alpha}$ is the parameter vector, *i.e.*, $\boldsymbol{\theta} = (\omega,\boldsymbol{\alpha})$ and $\lambda(t;\boldsymbol{\theta}) = \omega f(t;\boldsymbol{\alpha})$. In Table I, we summarize the representative NHPP-based SRMs which can be categorized to Model 2, where each model corresponds to the p.d.f. of software fault-detection time.

B. Kernel-based SRMs

Next consider the case where the software intensity function $\lambda(x)$ is completely unknown. In this case, the method of maximum likelihood does not work under the incomplete knowledge on the intensity function and/or fault-detection time distribution. The most intuitive but simplest method to estimate the software intensity function is a piecewise-linear interpolation. For the n software fault-detection time data, (i, x_i) $(i = 1, 2, \cdots, n)$, define $\hat{\lambda}(x) = 1/(x_i - x_{i-1})$ $(x_{i-1} < x \le x_i; x_0 = 0)$. Miller and Sofer [15] apply the following step-function estimate with breakpoints x_i:

$$\begin{aligned}\hat{\Lambda}(x) &= i + (x - x_i)/(x_{i+1} - x_i),\\ &\quad x_i < x \le x_{i+1};\ i = 0, 1, \cdots, n-1.\end{aligned} \tag{4}$$

The resulting estimate of the mean value function in Eq.(4) is obtained by plotting n failure points and connecting them by line segments. Since only one sample path $t_i;\ i = 0,\ 1,\ 2,\ \ldots,\ n$, is available in a single software testing, it seems to be the straightforward but the most natural estimate of the cumulative number of software faults, because the mean squares error with the underlying software fault data is always zero. Figure 1 illustrates an estimate of the intensity function and its associated mean value function with real time scale $t_i = x_i t_n$ $(i = 0,\ 1,\ 2,\ \ldots, n)$. It can be seen that an estimate of the intensity function tends to fluctuate everywhere with big noise. Miller and Sofer [15] propose a smoothing technique of the intensity function by means of a quadratic programming, and concern to smooth the fluctuated estimate.

In this paper we focus on alternative approach based on the kernel function. Diggle and Marron [5] prove the equivalence between the well-known kernel density estimation of a continuous p.d.f. and the kernel intensity estimation of an NHPP. Let $x_i \equiv t_i/t_n \in (0,1]$ be the normalized data set and define the arrival time sequence of the normalized NHPP by $\chi = \{x_1, \ldots, x_n\}$. Then, an estimate of the intensity function of the normalized NHPP, which is called *the kernel intensity function*, is given by

$$\hat{\lambda}(x \mid \chi) = \frac{1}{h}\sum_{i=1}^{n} K\Big(\frac{x - x_i}{h}\Big), \tag{5}$$

where the function $K(\cdot)$ is the kernel function and $h\ (> 0)$ is the bandwidth. Barghout *et al.* [1] and Wang *et al.* [21] assume the Gaussian kernel function. In this paper we use the following four kernel functions:

$$K_1(x) = (1 - |x|)I_{[-1,1]}(x) \quad \text{(Triangle)}, \tag{6}$$

$$K_2(x) = \frac{3}{4}(1 - x^2)I_{[-1,1]}(x) \quad \text{(Quadrtic)}, \tag{7}$$

$$K_3(x) = \frac{15}{16}(1 - x^2)^2 I_{[-1,1]}(x) \quad \text{(Triweigh)}, \tag{8}$$

$$K_4(x) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{x^2}{2}) \quad \text{(Gaussian)}, \tag{9}$$

where

$$I_{[-1,1]}(x) \equiv \begin{cases} 1 & \text{for } x \in [-1,1], \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

is the indicator function. Since the kernel function with an arbitrary design parameter h is replaced by the intensity function, the present model can be classified into Model 1.

On the other hand, the intensity function in Model 2 is defined by $\lambda(x;\boldsymbol{\theta}) = \omega f(x;\boldsymbol{\alpha})$. Since the function $f(x;\boldsymbol{\alpha})$

is the p.d.f., it is straightforward to apply the common kernel density estimation by

$$\hat{f}(x \mid \chi) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{11}$$

The similar but rather simplified problems are considered in Barghout *et al.* [1] with Eq.(11). In this case, another parameter ω has to be estimated in addition to the bandwidth h.

C. Non-parametric Estimation

It is well known in the kernel-based estimation that the determination of bandwidth h but not the selection of the kernel function is more sensitive to the estimation accuracy. Diggle and Marron [5] apply the least-squares cross-validation (LSCV) method to determine the bandwidth. In LSCV, the software fault-detection time data are divided into training data and validation data. By leaving out one of each i-th ($i = 1,\ 2,\ \ldots,\ n$) data from χ, we construct n training data sets consisting of $(n-1)$ software fault-detection time data. For an arbitrary time t, define the integrated least squares error of the software intensity function $\hat{\lambda}(x)$ by

$$\mathrm{ISE}(h) = \int_0^1 \left(\hat{\lambda}(x) - \lambda(x)\right)^2 dx. \tag{12}$$

By removing independent terms of h in Eq.(12), Diggle and Marron [5] show that obtaining the optimal bandwidth minimizing ISE(h) is equivalent to obtaining h minimizing the following function:

$$CV(h) = \int_0^1 \hat{\lambda}(x)^2 dx - 2\sum_{j=1}^{n} \hat{\lambda}_{h,j}(x_j), \tag{13}$$

where

$$\hat{\lambda}_{h,j}(x) = \frac{1}{h}\sum_{i=1, i\neq j}^{n} K\Big(\frac{x - x_j}{h}\Big). \tag{14}$$

On the other hand, if one is interested in the maximization on the likelihood function with the unknown mean value function, we can apply the log-likelihood cross-validation (LLCV) method [9] for the same n training data sets, and determine the optimal bandwidth h maximizing it. Define the log-likelihood function:

$$\ln L(h) = \sum_{j=1}^{n} \ln \hat{\lambda}_{h,j}(x_j) - \sum_{j=1}^{n} \hat{\Lambda}_{h,j}(x_j), \tag{15}$$

$$\hat{\Lambda}_{h,j}(x) = \int_0^x \hat{\lambda}_{h,j}(s) ds. \tag{16}$$

These two estimation methods; LSCV and LLCV, are used for Model 1.

In Model 2, both LSCV and LLCV are applicable by replacing the intensity functions in Eqs. (13) and (15) by $\hat{\lambda}(x \mid \chi) = \hat{\omega}\hat{f}(x \mid \chi)$, where the objective functions involve two decision variables (ω, h). However, it should be noted that the resulting intensity function based on LSCV in Model 2 is exactly equivalent to that in Model 1, because substituting $\hat{\lambda}(x \mid \chi) = \hat{\omega}\hat{f}(x \mid \chi)$ into Eq.(13) yields

$$CV(\omega, h) = \left(\frac{\omega}{n}\right)^2 \int_0^1 \hat{\lambda}(x)^2 dx - 2\left(\frac{\omega}{n}\right)\sum_{j=1}^{n} \hat{\lambda}_{h,j}(x_j) \tag{17}$$

and an estimate of the initial number of software faults before the system testing in Model 2 is always given by $\hat{\omega} = n$. Focusing on the difference between estimates of the intensity function in Eq.(5) and the p.d.f. in Eq.(11), Model 1 is exactly same as Model 2 in the sense of LSCV. In LLCV, it is evident to show that $\hat{\omega} \neq n$ in general and two models can be distinguished.

Apart from two conventional methods, LSCV and LLCV, Wang *et al.* [21] consider the so-called local likelihood method (LLM) for Model 1 with the Gaussian kernel function in Eq.(9). By utilizing the logarithmic transformation of the intensity function, *i.e.*, $g(x) = \log\lambda(x)$, they apply the linear approximation around the point t_0 to the function $g(x)$ as

$$g(x) \approx a + b(x - t_0). \tag{18}$$

Substituting Eq.(18) into Eq.(3), we have the following approximate log likelihood function as a function of t_0:

$$\begin{aligned}\ln L(t_0, a, b) &= -\int_0^1 \exp\{a + b(x - t_0)\}dx \\ &\quad + \sum_{i=1}^{n}\{a + b(x_i - t_0)\}.\end{aligned} \tag{19}$$

To make this approximation effective at the points far from t_0, Wang *et al.* [21] modify the approximate log likelihood function by regarding the Gaussian kernel function $K_4(\cdot)$ as weights, and give the weighted log likelihood function by

$$\begin{aligned}\ln L(t_0, a, b) = &\sum_{i=1}^{n} K\left(\frac{x_i - t_0}{h}\right)\{a + b(x_i - t_0)\} \\ &- \int_0^1 K\left(\frac{x - t_0}{h}\right)\exp\{a + b(x - t_0)\}dx.\end{aligned} \tag{20}$$

The next step is to solve the simultaneous weighted log likelihood functions:

$$\begin{aligned}&\sum_{j=1}^{n} K\left(\frac{x_j - t_0}{h}\right) \\ &\quad = \int_0^1 K\left(\frac{x - t_0}{h}\right)\exp\{a + b(x - t_0)\}dx,\end{aligned} \tag{21}$$

$$\begin{aligned}&\sum_{i=1}^{n} K\left(\frac{x_i - t_0}{h}\right)(x_i - t_0) \\ &\quad = \int_0^1 K\left(\frac{x - t_0}{h}\right)(x - t_0)\exp\{a + b(x - t_0)\}dx\end{aligned} \tag{22}$$

with respect to $t_0 \in [0, x]$ and to seek the parameters $(\hat{a}, \hat{b})$, where an estimate of the intensity function is given by $\hat{\lambda}(t_0) = \exp(\hat{a})$. By changing the estimation point t_0 sequentially for an arbitrary time point $x \in (0, 1]$, the bandwidth h is estimated so as to satisfy

$$\hat{h} = \frac{x_{l+k} - x_{l-k}}{2}, \tag{23}$$

where x_l is the closest scaled fault-detection time data to t_0, and k is the integer satisfying $\hat{h} = t_0 - x_{l-k}$ if $l + k > n$ ($\hat{h} = x_{l+k} - t_0$ if $l - k < 1$). Note again that LLM can be applied to only Model 1, but not Model 2.

D. Prediction with Weighted Kernel Functions

Based on the past observation on software fault-detection time χ, it is possible to estimate the software intensity and the mean value functions defined on $x \in [0, 1]$, However, since the non-parametric estimation methods employed here are not defined on $x \in (1, \infty)$, they cannot be applied to prediction of the future behavior of software fault-detection phenomenon. Caires and Ferreira [3] notice that an estimate of the intensity function tends to take larger value if many arrival time data are located around the estimation point of time, and propose to apply a weighted kernel function for prediction. Define

$$\hat{\lambda}_p(x|\chi) = \sum_{i=1}^{n} \frac{\hat{\lambda}(x_i) K(\frac{x - x_i}{h})}{\sum_{j=1}^{n} K(\frac{x - j}{h})}. \tag{24}$$

In the above expression, the weighted kernel function corresponds to a weighted normalized distance between the software fault-detection time and its estimate. As an effect of the weighted kernel function, it should be emphasized that the information on the intensity function estimated at the closer time to the prediction point reflects to the prediction in the future. In this prediction method, the bandwidth h influences to the range of software fault-detection time data and depends on the accuracy of prediction. In the prediction of the intensity function, we use the plug-in estimator $\hat{h}$ with LSCV and LLCV based on the past observation.

III. Non-parametric Interval Estimation

A. Bootstrapping

The kernel-based estimation in Section II can classified into a point estimation of the software intensity function. Since the estimate is based on only one sample sequence $\chi = \{x_1, \ \ldots, \ x_n\}$, it does not take the uncertainty of the corresponding estimator itself as a random variable into consideration. The statistical bootstrap is a combination of data re-sampling and replication of estimation. If the underlying c.d.f. of the software fault-detection time is completely known, the Monte Carlo simulation can be used to generate realizations of the random samples. On the other hand, even if the underlying c.d.f. is unknown, it is possible to re-sample the data with replacement. This is called the bootstrap sample. For the purpose in the software reliability assessment, we replicate m software fault-detection time data sets from the underlying data χ. Let X^*_{ki} be the random sample of x_i at k-th $(= 1, \ 2, \ \cdots, \ m)$ sampling. For the kernel-based estimation in Model 1 and Model 2, we first estimate the bandwidth h and/or ω. Second, we take the following three bootstrapping (BSP) methods to replicate the data sets.

(i) BSP Method 1: Based on an estimate of software intensity function $\hat{\lambda}(x)$ with one of four kernel functions $(K_1(\cdot) \sim K_4(\cdot))$ and one of two estimation methods (LSCV, LLCV) in each model (Model 1 or Model 2), we generate the pseudo random time sequence X^*_{ki} at k-th simulation, where the well-known *thinning algorithm* [13] is used to generate the random variates. More precisely, suppose that the intensity (but not mean value) function is bounded from the above and takes the maximum value $\bar{\lambda}$ $(\geq \lambda(x))$ for an arbitrary $x \geq 0$. Then, it can be shown that $X^*_{k1}, \ X^*_{k2}, \ \cdots \ (k = 1, 2, \cdots, m)$ follows an NHPP with the intensity function $\bar{\lambda}$, and that the i-th arrival time X_{ki} having the intensity function $\lambda(x^*_{ki})/\bar{\lambda}$ is independent of the other arrival times. The resulting bootstrap sample x^*_{ki} $(i = 1, 2, \ldots, k = 1, 2, \ldots, m)$ is called BSP 1 in this paper.

(ii) BSP Method 2: Next we consider the re-sampling based method. Given the underlying software fault-detection time data $\chi = \{x_1, \ \ldots, \ x_n\}$, we re-sample exactly n software fault-detection time data with replacement. Let N^*_k be the number of fault-detection time after removing the identical data at the k-th sampling. Then, we have the bootstrap sample x^*_{ki} $(i = 1, 2, \ldots, N^*_k, k = 1, 2, \ldots, m)$ and call this BSP 2.

(iii) BSP Method 3: The third method is almost similar to BSP 2, where the number of software fault-detection times re-sampled is given by the Poisson distributed (pseudo) random number with mean n. We call this sample BSP 3.

Based on the above three methods, the BSP estimate of the intensity function is given by

$$\hat{\lambda}^*(x \mid \chi) \quad = \quad \frac{\sum_{k=1}^{m} \hat{\lambda}^*_k(x \mid \chi)}{m}, \tag{25}$$

where

$$\hat{\lambda}^*_k(x \mid \chi) = \frac{1}{h} \sum_{i=1}^{N^*_k} K\Big(\frac{x - x^*_{ki}}{h}\Big) \tag{26}$$

and

$$\hat{\lambda}^*_k(x \mid \chi) = \frac{1}{N^*_k h} \sum_{i=1}^{N^*_k} \hat{\omega} K\Big(\frac{x - x^*_{ki}}{h}\Big) \tag{27}$$

in Model 1 and Model 2, respectively.

B. Estimation of Probability Distributions of Estimators

Once an estimate of the software intensity function based on the BSP sample is given, we can define the ordered statistics of the intensity function by $0 = \hat{\lambda}^*_{(0)}(x) \leq \hat{\lambda}^*_{(1)}(x) \leq \hat{\lambda}^*_{(2)}(x) \leq \ldots \leq \hat{\lambda}^*_{(m)}(x)$ at time x, and regard as the complete sample from the random variable $\hat{\lambda}^*(X)$. If there exists the c.d.f.

of $\hat{\lambda}^*(X)$, the empirical distribution function $G_{km}(\lambda)$ corresponding to the sample estimates $\hat{\lambda}^*_{(k)}(x)$ $(k = 0, 1, 2, \dots, m)$ is given by

$$G_{km}(\lambda) = \begin{cases} k/m & \text{for} \quad \hat{\lambda}^*_{(k)}(x) \le \lambda < \hat{\lambda}^*_{(k+1)}(x), \\ 1 & \text{for} \quad \hat{\lambda}^*_{(m)}(x) \le \lambda. \end{cases} \tag{28}$$

It is well known that the above empirical c.d.f. approaches to the real (but unknown) c.d.f. of the estimator $\hat{\lambda}^*(X)$ as $m \to \infty$ and is strongly consistent. The confidence region of $\hat{\lambda}^*(X)$ is defined as the quantile of Eq. (28), so that the two-sided $100p\%$ confidence interval is given by $(\hat{\lambda}^*_{(i_L)}(x), \hat{\lambda}^*_{(i_U)}(x))$, where i_L and i_U are indices corresponding to $100(1-p)\%$ and $100p\%$-quantiles satisfying $G_{km}(\hat{\lambda}^*_{(i_L)}(x)) = 100(1-p)$ and $G_{km}(\hat{\lambda}^*_{(i_U)}(x)) = 100p$, respectively. From the empirical c.d.f. of $\hat{\lambda}^*(X)$, we can obtain not only the mean $\mathrm{E}[\hat{\lambda}^*(X)] = \hat{\lambda}^*(x)$ and its higher moments, but also the confidence interval with significant level $100p\%$. The similar approach can be taken to the other reliability measures. For instance, if one is interested in the cumulative number of faults by time x, then it is possible to obtain the ordered estimates of the mean value function $0 = \hat{\Lambda}^*_{(0)}(x) \le \hat{\Lambda}^*_{(1)}(x) \le \hat{\Lambda}^*_{(2)}(x) \le \dots \le \hat{\Lambda}^*_{(m)}(x)$ and its associated empirical distribution, which is the c.d.f. of an estimator of the mean value function $\hat{\Lambda}^*(X)$ (but not the NHPP $\{N(x), x \ge 0\}$).

This idea can be applied to an interval estimation of the quantitative software reliability. The software reliability function is defined as the probability that the software system does not fail during the operational period τ after releasing at time $x = 1$. For an arbitrary release time x, define the software reliability function by

$$R(\tau|x) = e^{-\int_t^{\tau+x} \lambda(s)ds} = e^{\Lambda(x) - \Lambda(\tau+x)}. \tag{29}$$

Since the prediction of the intensity function is given by $\hat{\lambda}_p(x + \tau|\chi)$ in Eq.(24), we obtain the point estimates of the mean value function $\hat{\Lambda}_p(x + \tau|\chi)$ and the reliability function $\hat{R}_p(\tau|x)$ from Eq.(29). Next, based on an arbitrary BSP method, we have the ordered statistics of the reliability function $\hat{R}_{p,(1)}(\tau|x) \le \hat{R}_{p,(2)}(\tau|x) \le \dots \le \hat{R}_{p,(m)}(\tau|x)$ and get the corresponding empirical c.d.f. of an estimator of the reliability function $\hat{R}_p(\tau|X)$.

IV. Numerical Illustrations

A. Data Set

We analyze four data sets on the software fault-detection time data, which is observed in real software development projects. Here we just introduce the results for one of them for brevity. The underlying data set consists of 86 fault-detection time data and is the well-known reference data recorded in AT&T [14], [16]. We use four kernel functions and three (one) parameter estimation methods in Model 1 (Model 2) for the point estimation, and three BSP methods for the interval estimation. As competitors of non-parametric NHPP-based SRMs, we try to investigate the goodness-of-fit performance on ten parametric NHPP-based SRMs in Table I, and calculate the maximum log likelihood (MLL) and the mean squares error (MSE) between the estimated mean value function and the fault data given by

$$\mathrm{MSE} = \frac{\sqrt{\sum_{i=1}^{n} \{i - \hat{\Lambda}(x_i)\}^2}}{n}. \tag{30}$$

TABLE II
Comparison of Goodness-of-fit performance with parametric NHPP-based SRMs.

Model	MLL	MSE
EXP	-686.538	0.219
GMA	-686.348	0.172
TRN	-687.012	0.295
LTR	-685.840	0.127
TLG	-686.799	0.253
LLG	-686.125	0.139
TXA	-686.724	0.244
LXA	-685.854	0.129
TXI	-687.205	0.334
LXI	-686.293	0.168

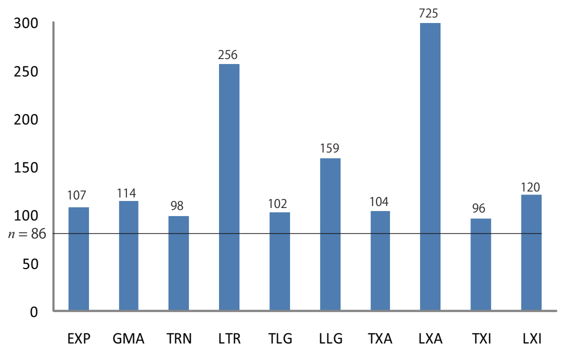


Fig. 2. Comparison of estimates of the initial number of software faults using 10 parametric NHPP-based SRMs.

B. Goodness-of-Fit Test

Table II presents MLL and MSE for ten parametric NHPP-based SRMs, where LTR provides the largest MLL and the smallest MSE among them. It is noted that the number of free parameters in the parametric NHPP-based SRMs may be different from each other (2 or 3), the comparison through MLL and MSE should be adjusted with the information criteria such as Akaike information criterion (AIC) and Bayesian information criterion (BIC). In Fig. 2, we plot estimates of the initial number of software faults using 10 parametric NHPP-based SRMs. For the number of software faults detected in the system testing $n = 86$, LTR overestimates the number of residual faults, because the maximum likelihood estimate of ω tends to approach to the number of underlying fault count data. In other words, it can be concluded that except LTR and LXA, almost parametric NHPP-based SRMs show the similar

TABLE III
COMPARISON OF GOODNESS-OF-FIT PERFORMANCES WITH NON-PARAMETRIC NHPP-BASED SRMS (MODEL 1).

Kernel	Estimation	MLL	MSE
K_1	LLM	-679.990	0.176
	LSCV	-687.451	0.755
	LLCV	-675.813	0.197
K_2	LLM	-682.939	0.182
	LSCV	-687.752	0.737
	LLCV	-680.503	0.216
K_3	LLM	-681.665	0.202
	LSCV	-687.821	0.754
	LLCV	-677.493	0.190
K_4	LLM	-684.114	0.113
	LSCV	-687.876	0.784
	LLCV	-562.353	0.061

TABLE IV
COMPARISON OF GOODNESS-OF-FIT PERFORMANCES WITH NON-PARAMETRIC NHPP-BASED SRMS (MODEL 2).

Kernel	Estimation	MLL	MSE
K_1	LLCV	-675.803	0.110
K_2	LLCV	-680.487	0.119
K_3	LLCV	-677.485	0.108
K_4	LLCV	-562.386	0.141

goodness-of-fit.

Tables III and IV present MLL and MSE for the non-parametric NHPP-based SRMs in Model 1 and Model 2, respectively. In Model 1 it is seen that the Gaussian kernel $K_4(\cdot)$ with LLCV provides the largest MLL and smallest MSE. Comparing with the results on the parametric NHPP-based SRMs in Table II, the difference are quite remarkable though Model 1 involves only one free parameter h. In Table IV, we find that Model 2 also provides the better goodness-of-fit performance than the parametric NHPP-based SRMs when the Gaussian kernel is assumed. In this case, the kernel function $K_3(\cdot)$ with LLCV takes a bit smaller MSE than $K_4(\cdot)$, but this difference is not so remarkable. In general, it is worth noting that Model 1 is superior to Model 2 under the same kernel function and the same parameter estimation, because the degree of freedom in Model 2 with parameters (ω, h) is less than that in Model 1, and it has a restriction in the form of $\lambda(x) = \omega f(x)$. Figure 3 is the plot of estimates of the initial number of software faults using non-parametric NHPP-based SRMs in Model 2. For $n = 86$, it is seen that the estimation results on the initial number of faults are more stable than the results in Fig. 2 and do not depend on the kind of kernel functions.

In Fig.4, we show the estimation results of both software intensity and mean value functions in Model 1 with different parameter estimation methods, where the Gaussian kernel function is assumed. The LLCV with largest MLLL and smallest MSE gives a fluctuated estimate of the intensity function with big noise, which is similar to that in the piecewise-linear interpolation [15] in Fig. 1. On the other hand, the estimates by means of LSCV and LLM are much smoother. When we look at the mean value function, LLCV and LLM rather fit the cumulative number of software faults, and this observation is consistent to the results in Table III. Since MLL and MSE are the criteria to measure the probabilistic distance and the vertical distance between SRMs and the underlying data, respectively, they do not essentially take account of the functional smoothness. In Fig. 5, we depict estimates of both software intensity and mean value functions in Model 2,

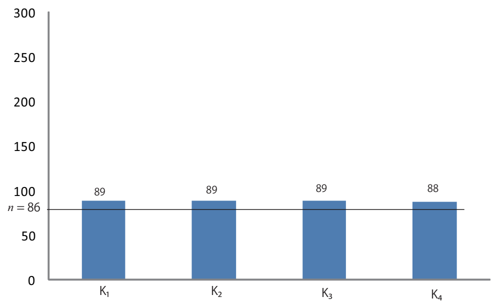
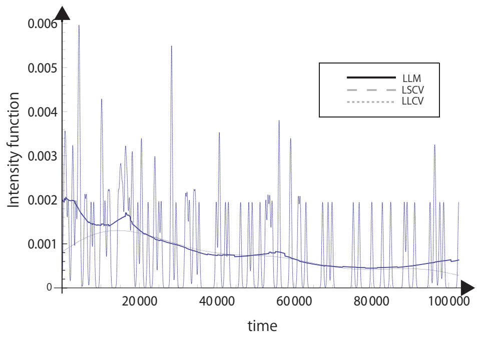
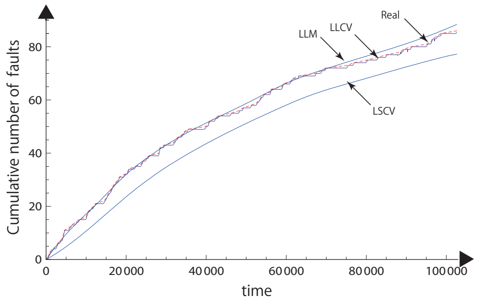


Fig. 3. Comparison of estimates of the initial number of software faults using non-parametric NHPP-based SRMs (Model 2).

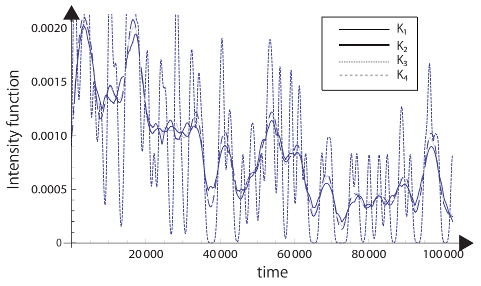


(i) Intensity function.

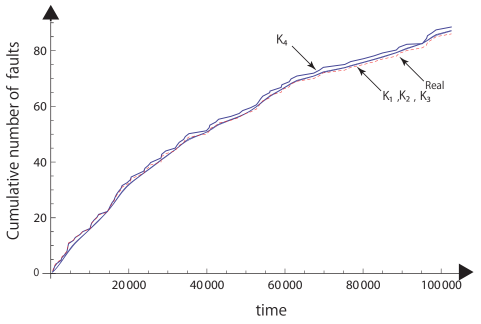


(ii) Mean value function.

Fig. 4. Behavior of estimates of software intensity and mean value functions in Model 1 with K_4.



(i) Intensity function.



(ii) Mean value function.

Fig. 5. Behavior of estimates of software intensity and mean value functions in Model 2.

TABLE V
COMPARISON OF PREDICTIVE MEAN SQUARES ERROR.

Kernel	Estimation	PMS
K_1	Models 1 & 2 (LSCV)	1017.75
	Model 1 (LLCV)	722.34
	Model 2 (LLCV)	454.90
K_2	Models 1 & 2 (LSCV)	927.08
	Model 1 (LLCV)	1784.26
	Model 2 (LLCV)	1311.97
K_3	Models 1 & 2 (LSCV)	1212.63
	Model 1 (LLCV)	1188.13
	Model 2 (LLCV)	741.70
K_4	Models 1 & 2 (LSCV)	788.89
	Model 1 (LLCV)	4478.73
	Model 2 (LLCV)	3144.91

where only the Gaussian kernel function tends to provide the similar fluctuated estimate of the intensity function to Fig. 1 as well. In this case, it can be checked that the Gaussian kernel function somewhat overestimates the cumulative number of faults, comparing with $K_1(\cdot)$, $K_2(\cdot)$ and $K_3(\cdot)$.

C. Predictive Performance

Next we investigate the predictive performance of the kernel-based methods. As mentioned in Section II, we predict the future software intensity function with a weighted kernel function in Eq.(24). Especially we concern the one-stage look-ahead prediction of the fault-detection time and estimate the next fault-detection time from the observation point [1]. More specifically, we estimate the $(k+1)$-st fault-detection time $\hat{x}_{k+1}$ from k-th $(k = 1, 2, \ldots, n)$ data point so as to satisfy

$$\int_{x_k}^{\hat{x}_{k+1}} \hat{\lambda}_p(x \mid \chi)dx = 1. \qquad (31)$$

The predictive performance is measured by the predictive mean squares error given by

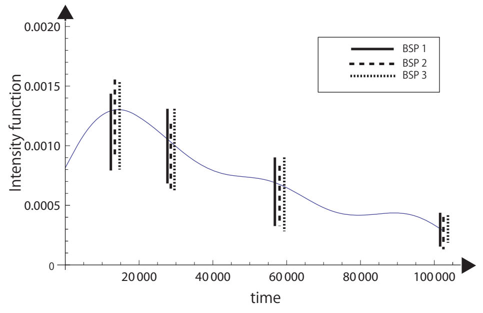
$$\text{PMS} = \frac{\sqrt{\sum_{j=1}^{l}(\hat{x}_{k+j} - x_{k-1+j}])^2}}{l}, \qquad (32)$$

where k is the number of detected faults at the observation time x_k and l is the number of predictions. In Table V we compare the predictive mean squares errors with different kernel functions and estimation methods. We do the one-stage look ahead prediction of the fault-detection time from the 75% point of the whole data. It is seen that Model 2 with K_1 and LLCV gives the smallest PMS and that the predictive performance is different from the goodness-of-fit performance. This is a well-known result in the software reliability engineering. Finally, we recommend the kernel functions $K_1(\cdot)$ and $K_4(\cdot)$ with LLCV for both Model 1 and Model 2.

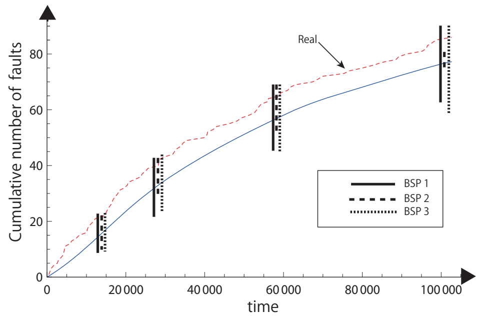
D. Interval Estimation

Next, we estimate the confidence interval of estimates of several reliability measures by applying three BSP methods. Throughout this paper, we fix the significance level as $100p = 95\%$. We generate $m = 10000$ samples based on the three BSP methods, and obtain $m = 10000$ estimates of software intensity function and mean value function. Based on these estimates, we obtain the corresponding empirical distributions of estimators, and calculate the 95% confidence interval. In Figs. 6-8, we plot the 95% confidence intervals of software intensity function and mean value function at the observation point, 25%, 50%, 75% and 100% of the whole data, where Model 1 and Model 2 with LSCV, Model 1 with LLCV and Model 2 with LLCV are examined for the Gaussian kernel function $K_4(\cdot)$. Looking at Fig. 6, it is seen that the cumulative number of detected faults is rather underestimated with LSCV. Also, in Fig. 8, we find that Model 2 with LLCV tends to overestimate the real cumulative number of faults. As Model 1 with LLCV provides the best goodness-of-fit performance in Table III, we can check that the point estimate in Eq.(25) based on the BSP sample shows the very nice goodness-of-fit result on the mean value function.

On the confidence interval, it is seen that the two-sided 95% confidence intervals of software intensity function in Model 1 with LSCV are almost similar among three BSP methods. On the other hand, the interval length of the mean value function for BSP 2 is shorter than those for BSP 1 and BSP 3 in the latter system testing phase. From the results including other models in Figs. 6 and 8, it can be seen that the confidence intervals based on BSP 2 method provide narrow bands comparing with the other BSP methods. Although we omit the discussion for brevity, the comparison with parametric

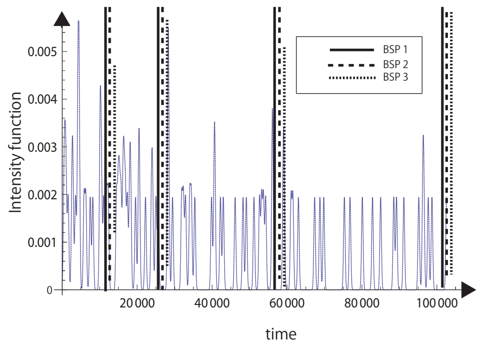


(i) Intensity function.

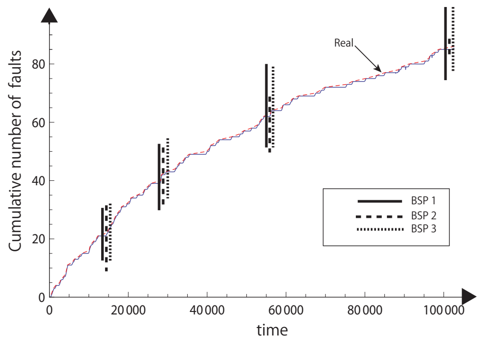


(ii) Mean value function.

Fig. 6. Confidence intervals of software intensity and mean value functions in Models 1 & 2 with LSCV.



(i) Intensity function.



(ii) Mean value function.

Fig. 7. Confidence intervals of software intensity and mean value functions in Model 1 with LLCV.

bootstrapping [11] tells us that the non-parametric confidence region of software intensity function tends to fluctuate in time more than the parametric one, but the interval estimation of the other reliability measures are quite stable in both methods. In Table VI, we derive the two-sided 95% confidence intervals of software reliability function, where the operational period is given by $\tau = 581.11$. When we focus on the Gaussian kernel function, it tends to estimate the confidence interval of quantitative software reliability as smaller value than the other kernel functions for an arbitrary BSP method, except with LSCV. That is, the estimation method based on LSCV always provides the relatively higher lower limit $\hat{\lambda}^*_{(i_L)}(x)$ than the other methods in the case with the Gaussian kernel. Based on the predictive performance in Table V, it can be checked that Model 1 (LLCV) with $K_1(\cdot)$ and Model 2 (LLCV) with $K_1(\cdot)/K_3(\cdot)$ give the similar two-sided confidence regions. The lesson learned from this example is that the two-sided 95% confidence interval of quantitative software reliability function has wide bands and depends on the kind of BSP method, bandwidth estimation method and the kernel function.

V. Conclusions

In this paper, we have developed a comprehensive software reliability assessment method under incomplete knowledge on software fault-detection time distribution, and investigated the goodness-of-fit performance and the predictive performance under several combinations of estimation techniques. The proposed kernel-based approach has been utilized to the interval estimation of some significant software reliability measures such as the software intensity function, mean value function and the software reliability function.

References

[1] M. Barghout, B. Littlewood, and A. Abdel-Ghaly. A non-parametric order statistics software reliability model. *Software Testing, Verification and Reliability*, 8(3):113–132, 1998.

[2] M. M. Brooks and S. J. Marron. Asymptotic optimality of the least squares cross-validation bandwidth for kernel estimates of intensity functions. *Stochastic Processes and Their Applications*, 38:157–165, 1991.

[3] S. Caires and J. A. Ferreira. On the non-parametric prediction of conditionally stationary sequences. *Statistical Inference Stochastic Processes*, 8(2):151–184, 2005.

[4] A. Cowling, P. Hall, and M. J. Phillips. Bootstrap confidence regions for the intensity of a Poisson point process. *Journal of American Statistical Association*, 91(436):1516–1524, 1996.

[5] P. Diggle and J. S. Marron. Equivalence of smoothing parameter selectors in density and intensity estimation. *Journal of the American Statistical Association*, 83(403):793–800, 1988.

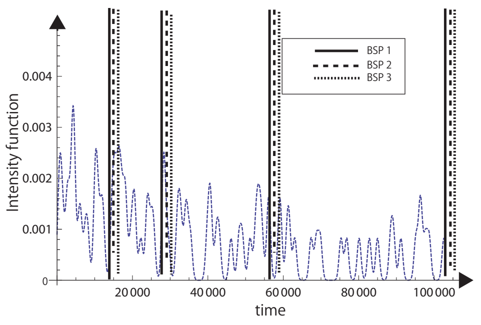
[6] L. S. Dharmasena, P. Zeephongsekul and C. L. Jayasinghe. Software reliability growth models based on local polynomial modeling with kernel smoothing. In *Proceedings of 22nd International Symposium on Software Reliability Engineering (ISSRE-2011)*, pages 220–229. IEEE CPS, 2011.

[7] A. Gandy and U. Jensen. A non-parametric approach to software reliability. *Applied Stochastic Models in Business and Industry*, 20(3):3–15, 2004.

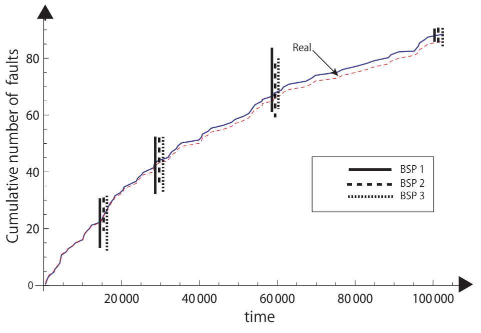
[8] A. L. Goel and K. Okumoto. Time-dependent error-detection rate model for software reliability and other performance measuress. *IEEE Transactions on Reliability*, R-28(3):206–211, 1979.

TABLE VI
CONFIDENCE INTERVALS OF SOFTWARE RELIABILITY FUNCTION.

Kernel	Estimation	BSP 1	BSP 2	BSP 3
K_1	Models 1 & 2 (LSCV)	[7.30E-1,9.11E-1]	[6.97E-1,8.83E-1]	[7.11E-1,9.03E-1]
	Model 1 (LLCV)	[4.63E-1,8.62E-1]	[6.40E-1,8.62E-1]	[6.40E-1,8.62E-1]
	Model 2 (LLCV)	[4.57E-1,8.89E-1]	[6.30E-1,8.57E-1]	[6.05E-1,8.79E-1]
K_2	Models 1 & 2 (LSCV)	[7.29E-1,9.15E-1]	[6.99E-1,8.88E-1]	[7.12E-1,9.07E-1]
	Model 1 (LLCV)	[4.95E-1,8.94E-1]	[7.15E-1,8.94E-1]	[7.15E-1,8.94E-1]
	Model 2 (LLCV)	[4.75E-1,8.99E-1]	[7.15E-1,8.94E-1]	[6.43E-1,8.97E-1]
K_3	Models 1 & 2 (LSCV)	[7.32E-1,9.14E-1]	[7.00E-1,8.86E-1]	[7.12E-1,9.08E-1]
	Model 1 (LLCV)	[4.55E-1,8.70E-1]	[6.58E-1,8.70E-1]	[6.58E-1,8.70E-1]
	Model 2 (LLCV)	[4.43E-1,8.77E-1]	[6.49E-1,8.66E-1]	[6.24E-1,8.87E-1]
K_4	Models 1 & 2 (LSCV)	[7.33E-1,9.08E-1]	[7.02E-1,8.83E-1]	[7.17E-1,9.03E-1]
	Model 1 (LLCV)	[1.13E-1,3.23E-1]	[3.37E-2,3.23E-1]	[3.37E-2,3.23E-1]
	Model 2 (LLCV)	[4.32E-3,1.72E-1]	[1.70E-3,1.19E-1]	[9.66E-4,1.72E-1]



(i) Intensity function.



(ii) Mean value function.

Fig. 8. Confidence intervals of software intensity and mean value functions in Model 2 with LLCV.

[9] Y. Guan. A composite likelihood cross-validation approach in selecting bandwwidth for the estimation of the pair correlation function. *Scandinavian Journal of Statistics*, 34(2):336–346, 2007.

[10] H. Joe. Statistical inference for general-order-statistics and nonhomogeneous Poisson process software reliability model. *IEEE Transactions on Software Engineering*, SE-15(2):1485–14902, 1989.

[11] T. Kaneishi and T. Dohi. Parametric bootstrapping for assessing software reliability measures. In *Proceedings of 17th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC-2011)*, pages 1–9, IEEE CPS, 2011.

[12] S. Lei and M. R. Smith. Evaluation of several nonparametric bootstrap methods to estimate confidence intervals for software metrics. *IEEE Transactions on Software Engineering*, 29(11):996–1004, 2003.

[13] P. A. W. Lewis and G. S. Shedler. Simulation of nonhomogeneous poisson processes by thinning. *Naval Research Logistics Qualterly*, 26(3):403–413, 1979.

[14] M. R. Lyu. *Handbook of Software Reliability Engineering*. McGraw-Hill, New York, 1996.

[15] D. R. Miller and A. Sofer. A non-parametric software reliability growth model. *IEEE Transactions on Reliability*, R-40(3):329–337, 1991.

[16] J. D. Musa, A. Iannino, and K. Okumoto. *Software Reliability, Measurement, Prediction, Application*. McGraw-Hill, New York, 1987.

[17] J. D. Musa and K. Okumoto. A logarithmic poisson execution time model for software reliability measurement. In *Proceedings of 7th International Conference on Software Engineering (ICSE-1984)*, pages 230–238. ACM/IEEE CPS, 1984.

[18] H. Okamura, M. Grottke, T. Dohi, and K. S. Trivedi. Variational bayesian approach for interval estimation of NHPP-based software reliability models. In *Proceedings of 2007 International Conference on Dependable Systems and Networks (DSN-2007)*, pages 698–707. IEEE CPS, 2007.

[19] M. C. van Pul. Asymptotic properties of a class of statistical models in software reliability. *Scandinavian Journal of Statistics*, 19(3):235–253, 1992.

[20] M. C. van Pul. Simulations on the Jelinski-Moranda model of software reliability; application of some parametric bootstrap methods. *Statistics and Computing*, 2(3):121–136, 1992.

[21] Z. Wang, J. Wang, and X. Liang. Non-parametric estimation for NHPP software reliability models. *Journal of Applied Statistics*, 34(1):107–119, 2007.

[22] X. Xiao and T. Dohi. Wavelet-based approach for estimating software reliability. In *Proceedings of 20th International Symposium on Software Reliability Engineering (ISSRE-2009)*, pages 11–20. IEEE CPS, 2009.

[23] X. Xiao and T. Dohi. Estimating software intensity function via multiscale analysis and its application to reliability assessment. In *Proceedings of 17th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC-2011)*, pages 10–19. IEEE CPS, 2011.

[24] S. Yamada and S. Osaki. Software reliability growth modeling: models and applications. *IEEE Transactions on Software Engineering*, SE-11(12):1431–1437, 1985.

[25] L. Yin and K. S. Trivedi. Confidence interval estimation of nhpp-based software reliability models. In *Proceedings of 10th International Symposium on Software Reliability Engineering (ISSRE-1999)*, pages 6–11. IEEE CPS, 1999.

[26] M. Zhao and M. Xie. On maximum likelihood estimation for a general non-homogeneous Poisson process. *Scandinavian Journal of Statistics*, 23(4):597–607, 1996.