Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2012, Article ID 524636, 9 pages
doi:10.1155/2012/524636

Research Article

A Comparative Study of Data Transformations for Wavelet Shrinkage Estimation with Application to Software Reliability Assessment

Xiao Xiao and Tadashi Dohi

Department of Information Engineering, Graduate School of Engineering, Hiroshima University, 1-4-1 Kagamiyama, Higashi-Hiroshima 739-8527, Japan

Correspondence should be addressed to Xiao Xiao, xiaoxiao@rel.hiroshima-u.ac.jp

Received 6 January 2012; Revised 6 March 2012; Accepted 6 March 2012

Academic Editor: Chin-Yu Huang

In our previous work, we proposed wavelet shrinkage estimation (WSE) for nonhomogeneous Poisson process (NHPP)-based software reliability models (SRMs), where WSE is a data-transform-based nonparametric estimation method. Among many variance-stabilizing data transformations, the Anscombe transform and the Fisz transform were employed. We have shown that it could provide higher goodness-of-fit performance than the conventional maximum likelihood estimation (MLE) and the least squares estimation (LSE) in many cases, in spite of its non-parametric nature, through numerical experiments with real software-fault count data. With the aim of improving the estimation accuracy of WSE, in this paper we introduce other three data transformations to preprocess the software-fault count data and investigate the influence of different data transformations to the estimation accuracy of WSE through goodness-of-fit test.

1. Introduction

In the field of software reliability engineering, the quantitative assessment of software reliability has become one of the main issues of this area. Especially, people are interested in finding several *software intensity functions* from the software-fault count data observed in the software testing phases, since the software intensity function in discrete time denotes the number of software faults detected per unit time. This directly makes it possible to estimate the number of remaining software faults and the quantitative software reliability, which is defined as the probability that software system does not fail during a specified time period under a specified operational environment. Moreover, these evaluation measures can be used in the decision making such as allocation of development resources and software release scheduling. Therefore, we are interested in developing a high-accuracy estimation method for the software intensity function.

Among over hundreds of software reliability models (SRMs) [1–3], nonhomogeneous Poisson process (NHPP)-based SRMs have gained much popularity in actual software testing phases. In many cases, the NHPP-based SRM is formulated as a parametric model, where the mean value function or its difference in discrete time or derivative in continuous time, called "*software intensity function*," can be considered as a unique parameter to govern the probabilistic property. One class of parametric NHPP-based SRMs is concerned with modeling the number of software faults detected in testing phases, initiated by Goel and Okumoto [4]. Afterwards, many parametric NHPP-based SRMs were proposed in the literatures [5–8] from various points of view. However, it is well known in the software reliability engineering community that there does not exist a uniquely best parametric NHPP-based SRM which can fit every type of software-fault count data. This fact implies that nonparametric methods without assuming parametric form should be used to describe the software debugging phenomenon which is different in each testing phase.

Apart from the traditional Bayesian framework, some frequentist approaches based on non-parametric statistics were introduced to estimate the quantitative software reliability. Sofer and Miller [9] used an elementary piecewise linear estimator of the NHPP intensity function from the software-fault detection time data and proposed a smoothing technique by means of quadratic programming. Gandy and Jensen [10] applied the kernel-based estimator to estimate the NHPP intensity function in a non-parametric way. Barghout et al. [11] also proposed a kernel-based non-parametric estimation for the order statistics-based SRMs, where the likelihood cross-validation and the prequential likelihood approaches were used to estimate the bandwidth. Wang et al. [12] applied the similar kernel method to the NHPP-based SRMs, where they focused on the local likelihood method with a locally weighted log-likelihood function. By combining the non-parametric estimation with the Bayesian framework, El-Aroui and Soler [13] and Wilson and Samaniego [14] developed non-parametric Bayesian estimation methods. It should be noted that, generally, these non-parametric estimation methods require high computational cost, which, in some cases, may be almost similar to or greater than an effort on model selection in the parametric SRMs.

Another class of non-parametric estimation methods for NHPP-based SRMs is the wavelet analysis-based approach, initiated by Xiao and Dohi [15]. They proposed the wavelet shrinkage estimation (WSE), which does not require solving any optimization problem, so that the implementation of estimation algorithms is rather easy than the other non-parametric methods. They compared their method with the conventional maximum likelihood estimation (MLE) and the least squares estimation (LSE) through goodness-of-fit test. It has been shown that WSE could provide higher goodness-of-fit performance than MLE and LSE in many cases, in spite of its non-parametric nature, through numerical experiments with real software-fault count data.

The fundamental idea of WSE is to remove the noise included in the observed software-fault count data to get a noise-free estimate of the software intensity function. It is performed through the following three-step procedure. First, the noise variance is stabilized by applying the data transformation to the data. This produces a time-series data in which the noise can be treated as Gaussian white noise. Second, the noise is removed using "*Haar-wavelet*" based denoising algorithm. Third, an inverse data transformation is applied to the denoised time-series data, obtaining the estimate of the software intensity function. Among many variance-stabilizing data transformations, the Anscombe transform [16] and the Fisz transform [17] were employed in the previous work [15]. The other well-known square root data transformations are Bartlett transform [18] and Freeman transform [19]. Both Anscombe transform and Freeman transform are actually natural extensions of the Bartlett transform.

This paper focuses on the first step of WSE and aims at identifying and emphasizing the influence that the data transformation exerts on the accuracy of WSE. The remaining part of this paper is planned as follows. In Section 2, we give a preliminary on NHPP-based software reliability modeling. Section 3 is devoted to introduce data transformations. Section 4 describes the WSE for NHPP-based SRMs in details. In Section 5, we carry out the real project data analysis and illustrate numerical examples to examine the effectiveness of the proposed methods. Finally the paper is concluded with future researches in Section 6.

2. NHPP-Based Software Reliability Modeling

Suppose that the number of software faults detected through a system test is observed at discrete time $i = 0, 1, 2, \ldots$. Let Y_i and $N_i = \sum_{k=0}^{i} Y_k$ denote the number of software faults detected at testing date i and its cumulative value, where $Y_0 = N_0 = 0$ is assumed without any loss of generality. The stochastic process $\{N_i : i = 0, 1, 2, \ldots\}$ is said to be a discrete non-homogeneous Poisson process (D-NHPP) if the probability mass function at time i is given by

$$\Pr\{N_i = m\} = \frac{\{\Lambda_i\}^m}{m!} \exp\{-\Lambda_i\}, \quad m = 0, 1, 2, \ldots, \tag{1}$$

where $\Lambda_i = E[N_i]$ is called the mean value function of a D-NHPP and means the expected cumulative number of software faults detected by testing date i. The function $\lambda_i = \Lambda_i - \Lambda_{i-1}$ $(i \geq 1)$ is called the discrete intensity function and implies the expected number of faults detected at testing date i, say $\lambda_i = E[Y_i]$.

MLE is one of the most commonly used parametric estimation method. Let θ denote the vector of parameters in the mean value function $\Lambda_i = \Lambda_i(\theta)$, and $x_i(y_i)$ denote the realization of $N_i(Y_i)$. When n software faults are detected, the log-likelihood function of a D-NHPP is given by

$$\begin{aligned} \mathcal{LLF}(\theta) = &\sum_{i=1}^{n} (x_i - x_{i-1}) \ln[\Lambda_i(\theta) - \Lambda_{i-1}(\theta)] - \Lambda_n(\theta) \\ &- \sum_{i=1}^{n} \ln[(x_i - x_{i-1})!], \end{aligned} \tag{2}$$

where $x_i = \sum_{k=0}^{i} y_k$ and $y_0 = x_0 = 0$. Then, the maximum likelihood estimate of θ, say $\widehat{\theta}$, is given by the solution of $\operatorname{argmax}_{\boldsymbol{\theta}} \mathcal{LLF}(\theta)$. Therefore, the estimate of the software intensity function $\lambda_i = \lambda_i(\theta)$ $(i = 0, 1, 2, \ldots)$ can be obtained by $\widehat{\lambda}_i = \Lambda_i(\widehat{\theta}) - \Lambda_{i-1}(\widehat{\theta})$ $(i \geq 1)$ with $\lambda_0 = 0$. In the following sections, we consider the problem of estimating the software intensity function from the noise-involved observation y_i, in a non-parametric way.

3. Variance Stabilizing Data Transformation

It is very familiar to make use of data transformations (DTs) to stabilize the variance of Poisson data. By using DT, the software-fault count data which follow the D-NHPP are approximately transformed to the Gaussian data. The most fundamental data-transform tool in statistics is the *Bartlett transform* (BT) [18]. Let η_i denote the Poisson white noise, that is,

$$Y_i = \lambda_i + \eta_i, \quad i = 1, 2 \ldots, n. \tag{3}$$

Table 1: Representative data transformaions.

	Data transformation	Inverse data transformation	Distribution of random variable after DT
BT1 [18]	$B_i = 2\sqrt{Y_i}$	$Y_i = 1/4 \times \{(B_i)^2\}$	$N(2\sqrt{\lambda_i - 1/4}, 1)$
BT2 [18]	$B_i = 2\sqrt{Y_i + 1/2}$	$Y_i = 1/4 \times \{(B_i)^2 - 2\}$	$N(2\sqrt{\lambda_i + 1/4}, 1)$
AT [16]	$S_i = 2\sqrt{Y_i + 3/8}$	$Y_i = 1/4 \times \{(S_i)^2 - 3/2\}$	$N(2\sqrt{\lambda_i + 1/8}, 1)$
FT [19]	$F_i = \sqrt{Y_i + 1} + \sqrt{Y_i}$	$Y_i = 1/4 \times \{(F_i)^2 + (F_i)^{-2} - 2\}$	$N(2\sqrt{\lambda_i}, 1)$

Table 2: Representative discrete NHPP-based SRMs.

SRM	Mean value function Λ_i
Geometric (GE)	$\omega\{1 - (1-p)^i\}$
Negative Binomial (NB)	$\omega\{\sum_{k=1}^{i}((r+k-1)!/(r-1)!k!)p^r(1-p)^{k-1}\}$
Discrete Weibull (DW)	$\omega\{1 - p^{i^r}\}$

Taking the BT, the random variables:

$$B_i = 2\sqrt{Y_i}, \quad i = 1, 2, \ldots, n \tag{4}$$

can be approximately regarded as Gaussian random variables with the normal distribution $N(2\sqrt{\lambda_i - 1/4}, 1)$, so that the realizations:

$$b_i = 2\sqrt{y_i}, \quad i = 1, 2, \ldots, n \tag{5}$$

can be considered as samples from $N(2\sqrt{\lambda_i - 1/4}, 1)$. That is, the transformed realizations b_i $(i = 1, 2, \ldots, n)$ by the BT are the ones from the normally distributed random variables:

$$B_i = \lambda_i' + \nu_i, \quad i = 1, 2, \ldots, n, \tag{6}$$

where $\lambda_i' = 2\sqrt{\lambda_i - 1/4}$ is the transformed software intensity function, and ν_i is the Gaussian white noise with unit variance.

Bartlett [18] also showed that

$$b_i = 2\sqrt{y_i + \frac{1}{2}}, \quad i = 1, 2, \ldots, n \tag{7}$$

is a better transformation since it provides a constant variance more closely to 1, even when the mean of Y_i is not large.

The Anscombe transform (AT) [16] is a natural extension of BT and is employed in our previous work [15]. AT is of the following form:

$$s_i = 2\sqrt{y_i + \frac{3}{8}}, \quad i = 1, 2, \ldots, n, \tag{8}$$

where s_i can be considered as observations of Gaussian random variable $S_i = 2\sqrt{Y_i + 3/8}$ with the normal distribution $N(2\sqrt{\lambda_i + 1/8}, 1)$. Freeman and Tukey [19] proposed the following square-root transform (we call it FT), which is also an extension of BT:

$$f_i = \sqrt{y_i + 1} + \sqrt{y_i}, \quad i = 1, 2, \ldots, n. \tag{9}$$

They showed that the variance of Gaussian random variable $F_i = 2\sqrt{Y_i + 1} + \sqrt{Y_i}$ is the nearest to 1 among BT, AT, and FT if the mean of Y_i is small. Recently, these variance stabilization techniques were used to LSE of the mean value function for the NHPP-based SRMs [20]. Table 1 summaries the DTs mentioned above.

As mentioned in Section 1, the first step of WSE is to apply the normalizing and variance-stabilizing DTs to the observed software-fault count data. In this paper, we employ BT, AT, and FT in the first and the third steps of WSE. Then, the target of denoising in the second step of WSE is the transformed data b_i, s_i or f_i $(i = 1, 2, \ldots, n)$. Letting b_i', s_i', and f_i' denote the denoised b_i, s_i, and f_i, respectively, the estimate of the original software intensity function λ_i can be obtained by taking the inverse DT of b_i', s_i' and f_i', as given in Table 1.

4. Wavelet Shrinkage Estimation for NHPP-Based SRM

4.1. Haar-Wavelet-Based Denoising Procedure. The Haar-wavelet-based shrinkage technique can be used as a denoising algorithm for the second step of WSE. In general, the noise removal is performed through the following three steps: (i) expanding the transformed time-series data to obtain the empirical wavelet coefficients, (ii) removing the noise included in the empirical wavelet coefficients using thresholding method, and (iii) making use of the denoised coefficients to calculate the estimate of the transformed software intensity function.

4.1.1. Haar Wavelet Transform. The Haar scaling function and the Haar wavelet function are defined as

$$\phi(i) = \begin{cases} 1 & (0 \le i < 1) \\ 0 & (\text{otherwise}), \end{cases}$$

$$\psi(i) = \begin{cases} 1 & \left(0 \le i < \frac{1}{2}\right) \\ -1 & \left(\frac{1}{2} \le i < 1\right) \\ 0 & (\text{otherwise}), \end{cases} \tag{10}$$

Table 3: Goodness-of-fit test results for different data transformations. (Threshold level: universal threshold.)

DS1	MSE_1	MSE_2	LL
HBT1(h, ut)	4.180	0.327	−153.759
HBT2(h, ut)	2.401	0.316	−144.850
HAT(h, ut)	2.573	0.317	−145.390
HFT(h, ut)	9.842	0.403	−383.785
HBT1(s, ut)	4.180	0.327	−153.759
HBT2(s, ut)	2.401	0.316	−144.850
HAT(s, ut)	2.573	0.317	−145.390
HFT(s, ut)	3.231	0.320	−148.023
DS2	MSE_1	MSE_2	LL
HBT1(h, ut)	12.449	1.141	−197.109
HBT2(h, ut)	11.600	1.138	−196.904
HAT(h, ut)	11.725	1.138	−196.953
HFT(h, ut)	11.847	1.138	−196.806
HBT1(s, ut)	12.450	1.141	−197.109
HBT2(s, ut)	11.600	1.138	−196.904
HAT(s, ut)	11.725	1.138	−196.953
HFT(s, ut)	11.847	1.138	−196.806
DS3	MSE_1	MSE_2	LL
HBT1(h, ut)	10.159	0.719	−143.181
HBT2(h, ut)	9.055	0.791	−192.903
HAT(h, ut)	9.286	0.793	−193.178
HFT(h, ut)	9.653	0.796	−167.025
HBT1(s, ut)	12.954	1.037	−212.321
HBT2(s, ut)	10.528	1.035	−213.978
HAT(s, ut)	10.813	1.035	−213.496
HFT(s, ut)	11.496	1.037	−213.419
DS4	MSE_1	MSE_2	LL
HBT1(h, ut)	2.065	0.304	−171.605
HBT2(h, ut)	1.781	0.304	−171.491
HAT(h, ut)	1.822	0.304	−171.504
HFT(h, ut)	1.864	0.304	−171.516
HBT1(s, ut)	2.065	0.304	−171.605
HBT2(s, ut)	1.781	0.304	−171.491
HAT(s, ut)	1.822	0.304	−171.504
HFT(s, ut)	1.864	0.304	−171.516
DS5	MSE_1	MSE_2	LL
HBT1(h, ut)	14.013	0.586	−304.198
HBT2(h, ut)	12.781	0.581	−301.073
HAT(h, ut)	12.941	0.581	−301.447
HFT(h, ut)	13.200	0.582	−301.841
HBT1(s, ut)	14.054	0.601	−312.551
HBT2(s, ut)	12.804	0.596	−309.709
HAT(s, ut)	12.968	0.597	−310.005
HFT(s, ut)	13.227	0.598	−310.480
DS6	MSE_1	MSE_2	LL
HBT1(h, ut)	12.707	0.521	−378.932
HBT2(h, ut)	11.899	0.525	−380.497
HAT(h, ut)	12.053	0.526	−380.987

Table 3: Continued.

DS6	MSE_1	MSE_2	LL
HFT(h, ut)	12.186	0.526	−381.071
HBT1(s, ut)	13.546	0.585	−423.335
HBT2(s, ut)	12.505	0.584	−421.480
HAT(s, ut)	12.670	0.584	−421.729
HFT(s, ut)	12.810	0.584	−421.866

respectively. By introducing the *scaling parameter* j and *shifting parameter* k, the Haar father wavelet and the Haar mother wavelet are defined by

$$\begin{aligned}\phi_{j,k}(i) &= 2^{-j/2}\phi\left(2^{-j}i-k\right),\\ \psi_{j,k}(i) &= 2^{-j/2}\psi\left(2^{-j}i-k\right),\end{aligned} \tag{11}$$

respectively. Then the target function, transformed software intensity function λ_i' $(i = 1, 2, \ldots, n)$, can be expressed in the following equation:

$$\lambda_i' = \sum_{k=0}^{2^{j_0}-1}\alpha_{j_0,k}\phi_{j_0,k}(i) + \sum_{j=j_0}^{\infty}\sum_{k=0}^{2^j-1}\beta_{j,k}\psi_{j,k}(i), \tag{12}$$

where

$$\alpha_{j_0,k} = \sum_{i=1}^{n}\lambda_i'\phi_{j_0,k}(i), \tag{13}$$

$$\beta_{j,k} = \sum_{i=1}^{n}\lambda_i'\psi_{j,k}(i) \tag{14}$$

are called the scaling coefficients and the wavelet coefficients, respectively, for any primary resolution level $j_0(\geq 0)$. Due to the implementability, it is reasonable to set an upper limit instead of ∞ for the resolution level j. In other words, the highest resolution level must be finite in practice. We use J to denote the highest resolution level. That is, the range of j in the second term of (12) is $j \in [j_0, J]$. The mapping from function λ_i' to coefficients $(\alpha_{j_0,k}, \beta_{j,k})$ is called the Haar wavelet transform (HWT).

Since b_i, s_i or f_i $(i = 1, 2, \ldots, n)$ can be considered as the observation of λ_i', the empirical scaling coefficients $c_{j_0,k}$ and the empirical wavelet coefficients $d_{j,k}$ of λ_i' can be calculated by (13) and (14) with λ_i' replaced by b_i, s_i or f_i. The noises involved in the empirical wavelet coefficients $d_{j,k}$ can be removed by the thresholding method that we will introduce later. Finally, the estimate of λ_i' can be obtained by taking the inverse HWT with denoised empirical coefficients.

4.1.2. Thresholding. In denoising the empirical wavelet coefficients, the common choices of thresholding method are the hard thresholding:

$$\delta_\tau(u) = u1_{|u|>\tau}, \tag{15}$$

and the soft thresholding:

$$\delta_\tau(u) = \operatorname{sgn}(u)(|u| - \tau)_+, \tag{16}$$

for a fixed threshold level $\tau(> 0)$, where 1_A is the indicator function of an event A, $\operatorname{sgn}(u)$ is the sign function of u and $(u)_+ = \max(0, u)$. There are many methods to determine the threshold level τ. In this paper, we use the universal threshold [21] and the "leave-out-half" cross-validation threshold [22]:

$$\begin{aligned}\tau &= \sqrt{2\log n},\\ \tau &= \left(1-\frac{\log 2}{\log n}\right)^{-1/2}\tau\left(\frac{n}{2}\right),\end{aligned} \tag{17}$$

where n is the length of the observation s_i. Hard thresholding is a "keep" or "kill" rule, while soft thresholding is a "shrink" or "kill" rule. Both thresholding methods and both threshold levels will be employed to work on the empirical wavelet coefficients $d_{j,k}$ for denoising.

4.2. Wavelet Shrinkage Estimation for NHPP-Based SRM. Since the software-fault count data is Poisson data, the preprocessing is necessary before making use of the Haar-wavelet-based denoising procedure. Xiao and Dohi [15] combined data transformation and the standard denoising procedure to propose the wavelet shrinkage estimation (WSE) for the D-NHPP-based SRMs. They call the HWT combined with AT, the Haar-Anscombe transform (HAT). Similarly, we call the HWT combined with BT and FT, the Haar-Bartlett transform (HBT) and the Haar-Freeman transform (HFT), respectively. In the numerical study, we investigate the goodness-of-fit performance of HBT- and HFT-based WSEs and compare them with the HAT-based WSE [15].

5. Numerical Study

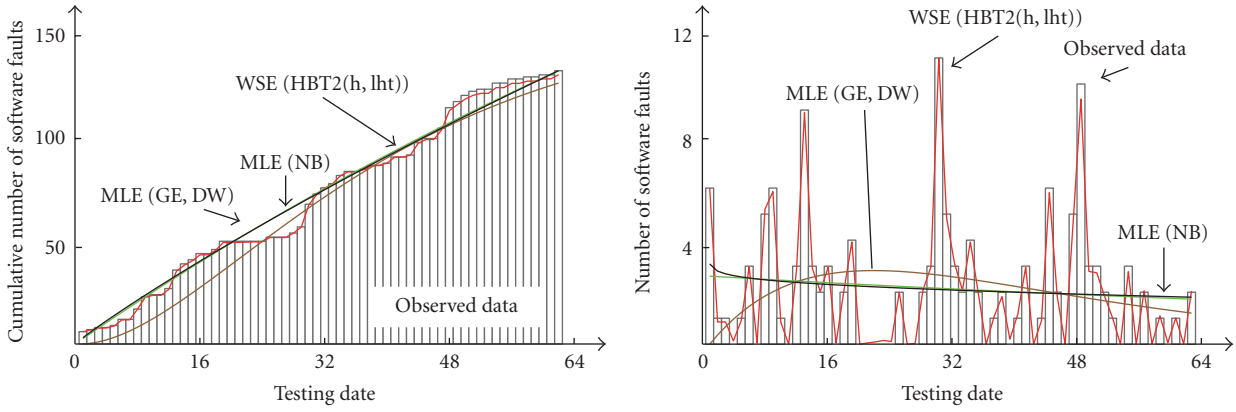
5.1. Data Sets and Measures. We use six real project data sets cited from reference [1], where they are named as J1, J3, DATA14, J5, SS1, and DATA8. These data sets are software-fault count data (group data). We rename them for convenience as DS1∼DS6 in this paper. Let (n, x_n) denote the pair of the final testing date and the total number of detected fault. Then these data sets are presented by (62, 133), (41, 351), (46, 266), (73, 367), (81, 461), and (111, 481),

Table 4: Goodness-of-fit test results for different data transformations. (Threshold level: "leave-out-half" cross-validation threshold.)

DS1	MSE_1	MSE_2	LL
HBT1(h, lht)	0.112	0.015	−59.704
HBT2(h, lht)	0.131	0.017	−60.634
HAT(h, lht)	0.114	0.014	−60.324
HFT(h, lht)	8.079	0.343	−216.320
HBT1(s, lht)	0.159	0.010	−59.581
HBT2(s, lht)	0.109	0.010	−60.195
HAT(s, lht)	0.115	0.010	−60.118
HFT(s, lht)	0.151	0.010	−59.609
DS2	MSE_1	MSE_2	LL
HBT1(h, lht)	0.477	0.102	−70.438
HBT2(h, lht)	0.225	0.072	−70.227
HAT(h, lht)	0.263	0.077	−70.663
HFT(h, lht)	0.552	0.125	−71.041
HBT1(s, lht)	0.342	0.027	−69.407
HBT2(s, lht)	0.363	0.030	−69.781
HAT(s, lht)	0.364	0.030	−69.731
HFT(s, lht)	0.351	0.028	−69.446
DS3	MSE_1	MSE_2	LL
HBT1(h, lht)	0.208	0.032	−55.829
HBT2(h, lht)	0.275	0.041	−56.767
HAT(h, lht)	0.902	0.074	−57.614
HFT(h, lht)	0.323	0.048	−56.411
HBT1(s, lht)	0.573	0.040	−55.847
HBT2(s, lht)	0.486	0.037	−56.649
HAT(s, lht)	0.496	0.037	−56.551
HFT(s, lht)	0.537	0.037	−55.922
DS4	MSE_1	MSE_2	LL
HBT1(h, lht)	0.288	0.015	−121.008
HBT2(h, lht)	0.132	0.007	−120.942
HAT(h, lht)	0.232	0.013	−121.031
HFT(h, lht)	0.281	0.014	−121.004
HBT1(s, lht)	0.097	0.005	−120.893
HBT2(s, lht)	0.046	0.003	−120.906
HAT(s, lht)	0.050	0.003	−120.907
HFT(s, lht)	0.064	0.004	−120.888
DS5	MSE_1	MSE_2	LL
HBT1(h, lht)	0.055	0.012	−120.869
HBT2(h, lht)	0.221	0.016	−121.205
HAT(h, lht)	0.228	0.017	−121.195
HFT(h, lht)	0.219	0.017	−120.890
HBT1(s, lht)	0.174	0.007	−120.805
HBT2(s, lht)	0.149	0.006	−120.913
HAT(s, lht)	0.151	0.006	−120.898
HFT(s, lht)	0.158	0.006	−120.806
DS6	MSE_1	MSE_2	LL
HBT1(h, lht)	0.778	0.025	−166.906
HBT2(h, lht)	0.558	0.017	−166.816
HAT(h, lht)	0.548	0.017	−166.767
HFT(h, lht)	0.540	0.017	−166.673

Table 4: Continued.

DS6	MSE_1	MSE_2	LL
HBT1(s, lht)	0.288	0.009	−166.432
HBT2(s, lht)	0.282	0.009	−166.532
HAT(s, lht)	0.289	0.010	−166.540
HFT(s, lht)	0.294	0.009	−166.440



(a) HBT2-based WSE(h, lht) versus MLE

(b) HBT2-based WSE(h, lht) versus MLE

Figure 1: Behavior of estimates with MLE and WSE using hard thresholding (DS1).

respectively. We employ the MSE (mean squares error) as the goodness-of-fit measures, where

$$\mathrm{MSE}_1 = \frac{\sqrt{\sum_{i=1}^{n}(\Lambda_i - x_i)^2}}{n}, \qquad \mathrm{MSE}_2 = \frac{\sqrt{\sum_{i=1}^{n}(\lambda_i - y_i)^2}}{n}. \tag{18}$$

Additionally, we calculate LL (Log Likelihood), which is defined as

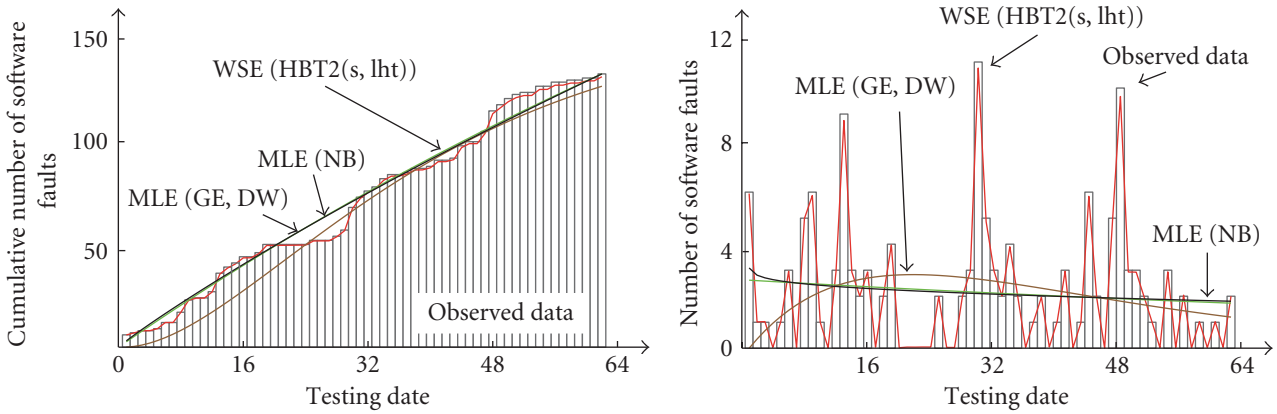
$$\mathrm{LL} = \sum_{i=1}^{n}(x_i - x_{i-1})\ln\left[\hat{\Lambda}_i - \hat{\Lambda}_{i-1}\right] - \hat{\Lambda}_n(\theta) - \sum_{i=1}^{n}\ln[(x_i - x_{i-1})!], \tag{19}$$

where $\hat{\Lambda}_i = \sum_{k=1}^{i}\hat{\lambda}_k$ $(i = 1, 2, \ldots, n)$, and $\hat{\lambda}_i$ is the WSE estimate of the software intensity function λ_i.

5.2. Goodness-of-Fit Test. A total of 16 wavelet-based estimation methods are examined in this paper since the WSE is applied with four thresholding techniques: hard thresholding (h) versus soft thresholding (s); universal threshold (ut) versus "leave-out-half," cross-validation threshold (lht). Let HBT$(\cdot,\cdot)$, HAT$(\cdot,\cdot)$ and HFT$(\cdot,\cdot)$ denote the WSEs based on Haar-Bartlett Transform, Haar-Anscombe Transform and Haar-Freeman Transform, respectively. HBT1$(\cdot,\cdot)$, and HBT2$(\cdot,\cdot)$ correspond to the transforms in (5) and (7), respectively. Additionally, the result of HAT-based WSE was introduced in [15], but they only showed the results of HAT-based WSE with hard thresholding. Here, we present comprehensively all the results of them for a further discussion.

We present the goodness-of-fit results based on different threshold levels in Tables 3 and 4, respectively. HBT2 provides smaller MSE and larger MLL than the others when "ut" is used, regardless of the thresholding method employed. It is worth mentioning that "ut" provides the same estimates with hard or soft thresholding in three data sets (DS1, DS2, and DS4). This is due to the relatively large value of "ut", since when threshold is set to be 0, the output of thresholding $\delta_\tau(d_{j,k})$ is the wavelet coefficient $d_{j,k}$ itself. In our numerical experiments, "ut" is relatively large in these 3 software-fault count data sets, which result in $\delta_{\mathrm{ut}}(d_{j,k}) = 0$ whichever hard or soft thresholding is applied. HBT2 also looks better than the others when software thresholding is applied with "lht." However, when "lht" is combined with hard thresholding, HBT1 (HAT; HFT) possesses the best results in DS3 and DS5 (DS1; DS6), respectively. Since "lht" is considered as a much more proper threshold level than "ut" in analyzing of software-fault count data, we suggest that HBT2 should be selected as an appropriate DT for the WSE.

5.3. Comparison with MLE. Our concern in this section is the comparison of WSE with MLE. We estimate the software intensity function by using three parametric D-NHPP-based SRMs listed in Table 2, where the MLE is applied to estimate the model parameters for comparison. HBT2-based WSE is selected as a representative one among the 16 wavelet-based estimation methods. Figures 1 and 2 depict the estimation behavior of cumulative number of software faults and its increment per testing date (individual number of software



(a) HBT2-based WSE(s, lht) versus MLE

(b) HBT2-based WSE(s, lht) versus MLE

Figure 2: Behavior of estimates with MLE and WSE using soft thresholding (DS1).

faults detected per testing date) with DS1. The observed data is plotted in bar graph in both figures. Looking at (i) and (iii) in these figures, it is clear that the parametric D-NHPP-based SRMs with maximum likelihood estimator can fit the real data. However, since the parametric D-NHPP-based SRMs assume the software intensity function as smooth function, they can estimate only the average tendency of the individual number of software faults detected at each testing date, but cannot follow the microscopic fluctuated behavior in (ii) and (iv) of Figures 1 and 2. In other words, the estimation accuracy based on the cumulative number of faults is embedded in "cumulative effects" in (i) and (iii). The experimental results performed here give the potential applicability of the wavelet-based estimation methods with different thresholding schemes. Our methods employed here do not need the expensive computational cost comparing with the MLE (within less than one second to get an estimate). This is a powerful advantage in applying the D-NHPP-based SRMs, in addition to the fact that practitioners do not request much time and effort to implement the wavelet-based estimation algorithms.

6. Concluding Remarks

In this paper, we have applied the wavelet-based techniques to estimate the software intensity function. Four data transformations were employed to preprocess the software-fault count data. Throughout the numerical evaluation, we could conclude that the wavelet-based estimation methods with Bartlett transform $2\sqrt{Y_i}$ have much more potential applicability than the other data transformations to the software reliability assessment practice because practitioners are not requested to carry out troublesome procedures on model selection and to take care of computational efficiency such as judgment of convergence and selecting initial guess of parameters in the general purpose of optimization algorithms. Note that, the result obtained here does not mean that the other data transformations are not good because the performance evaluation was executed only through goodness-of-fit test. Although the prediction ability of the proposed methods is out of focus of this paper at the present, the predictive performance should be considered and compared in the future.

References

[1] M. R. Lyu, Ed., *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, NY, USA, 1996.

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

[3] H. Pham, *Software Reliability*, Springer, Singapore, Singapore, 2000.

[4] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Transactions on Reliability*, vol. R-28, no. 3, pp. 206–211, 1979.

[5] A. L. Goel, "Software reliability models: assumptions, limitations and applicability," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 12, pp. 1411–1423, 1985.

[6] X. Xiao and T. Dohi, "Estimating software reliability using extreme value distribution," in *Proceedings of the International Conference on Advances in Software Engineering and Its Applications (ASEA '11)*, vol. CCIS 257, pp. 399–406, Springer, 2011.

[7] X. Xiao, H. Okamura, and T. Dohi, "NHPP-based software reliability models using equilibrium distribution," *IEICE Transactions of the Fundamentals A*, vol. E95-A, no. 5, pp. 894–902, 2012.

[8] S. Yamada, M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," *IEEE Transactions on Reliability*, vol. R-32, no. 5, pp. 475–484, 1983.

[9] A. Sofer and D. R. Miller, "A non-parametric software reliability growth model," *IEEE Transactions on Reliability*, vol. R-40, no. 3, pp. 329–337, 1991.

[10] A. Gandy and U. Jensen, "A non-parametric approach to software reliability," *Applied Stochastic Models in Business and Industry*, vol. 20, no. 1, pp. 3–15, 2004.

[11] M. Barghout, B. Littlewood, and A. Abdel-Ghaly, "A non-parametric order statistics software reliability model," *Software*

Testing Verification and Reliability, vol. 8, no. 3, pp. 113–132, 1998.

[12] Z. Wang, J. Wang, and X. Liang, "Non-parametric estimation for NHPP software reliability models," *Journal of Applied Statistics*, vol. 34, no. 1, pp. 107–119, 2007.

[13] M. A. El-Aroui and J. L. Soler, "A bayes nonparametric framework for software-reliability analysis," *IEEE Transactions on Reliability*, vol. 45, no. 4, pp. 652–660, 1996.

[14] S. P. Wilson and F. J. Samaniego, "Nonparametric analysis of the order-statistic model in software reliability," *IEEE Transactions on Software Engineering*, vol. 33, no. 3, pp. 198–208, 2007.

[15] X. Xiao and T. Dohi, "Wavelet-based approach for estimating software reliability," in *Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE '09)*, pp. 11–20, IEEE CS Press, November 2009.

[16] F. J. Anscombe, "The transformation of Poisson, binomial and negative binomial data," *Biometrika*, vol. 35, no. 3-4, pp. 246–254, 1948.

[17] M. Fisz, "The limiting distribution of a function of two independent random variables and its statistical application," *Colloquium Mathematicum*, vol. 3, pp. 138–146, 1955.

[18] M. S. Bartlett, "The square root transformation in the analysis of variance," *Journal of the Royal Statistical Society*, vol. 3, no. 1, pp. 68–78, 1936.

[19] M. F. Freeman and J. W. Tukey, "Transformations related to the angular and the square root," *The Annals of Mathematical Statistics*, vol. 21, no. 4, pp. 607–611, 1950.

[20] H. Ishii, T. Dohi, and H. Okamura, "Software reliability prediction based on least squares estimation," *Quality Technology and Quantitative Management Journal*. In press.

[21] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[22] G. P. Nason, "Wavelet shrinkage using cross-validation," *Journal of the Royal Statistical Society B*, vol. 58, no. 2, pp. 463–479, 1996.