On predicting software intensity using wavelets and nonlinear regression

1st Kaoru Matsui
Tokyo Metropolitan University
Tokyo, Japan
matsui-kaoru@ed.tmu.ac.jp

2nd Xiao Xiao
Tokyo Metropolitan University
Tokyo, Japan
xiaoxiao@tmu.ac.jp

***Abstract*—In the digital age, computer systems are vital in daily life. Many use software, but it often has problems like bugs, risk system failures, financial losses, and public safety. The need for reliable software is increasingly crucial as these issues become more common. In this paper, we consider predicting the number of software faults to evaluate the reliability of software quantitatively. We propose a new prediction method that combines Wavelet Shrinkage Estimation (WSE) with nonlinear regression. The critical aspect of our proposal is that it splits data into specific data lengths, enabling its application to a small number of data points. By modeling the time series variation of wavelet coefficients using periodic functions, we achieve long-term prediction within the nonparametric framework that can predict software intensity at any time point in the future. We verify the prediction accuracy of the proposed method and existing methods using real data.**

***Index Terms*—Software Reliability, Haar Wavelet, Trigonometric Function, Long-term Prediction, Real Data Analysis**

I. Introduction

Currently, modern society is made up of computer systems, which play an important role in our daily lives. Many modern systems are based on software. However, they are often subject to defects caused by bugs and other problems that can lead to system failures, financial losses, and even risks to public safety. The need for software reliability is gaining increasing attention. Many software reliability models, which are mathematical models that consider factors affecting software reliability, have been proposed. The widely used model is the non-homogeneous Poisson process (NHPP)-based software reliability growth model. Models such as Exponential model [5], Gamma model [20] and Normal model [10] are called parametric models, which assume specific functional forms of the intensity function of the NHPP models. In other words, the intensity function in parametric models are function of time and contains unknown parameters. The method of estimating the intensity function by determining parameters is called parametric estimation. One of the advantages of parametric models is the ease of predicting the number of faults. This is because the predicted value at any future times can be calculated as the output of the function once the model is determined and the parameters are estimated. On the other hand, a disadvantage is the difficulty of model selection. Parametric models have been studied for many years, and more than 200 models have been proposed to date. However, no model has been found among them that is suitable for arbitrary data. Therefore, it is necessary to select and assume the best model for the data from among many models. In addition, such assumptions are often far from reality to create a mathematical model concisely.

In order to solve these problems of parametric models and to improve the accuracy of estimation, research on nonparametric models without assumptions have been conducted extensively. The estimation of nonparametric models is performed directly from observed data without assuming any particular functional form for the description of the intensity function. However, the major disadvantage of nonparametric models is the difficulty in predicting future debugging behavior, because the intensity function is not expressed as a function of time. To solve the above problems, Karunanithi *et al.* [6] proposed a method to predict the number of software faults using neural networks (NNs). Su *et al.* [13] proposed the NN-based approaches to achieve a dynamic weighted combinational model (DWCM) to predict the cumulative number of software faults. In recent years, researches on prediction methods using NNs have been actively conducted, and it is well known that it requires large amounts of data for training, but it is not realistic to collect large amounts of data in this field. Furthermore, models created with NNs are so complex that they are unable to understand the essence of the data and provide useful feedback to developers.

On the other hand, as methods to see through the essence of data, the kernel-based approaches have been considered. Barghout *et al.* [2] proposed an order statistics model that predicts the reliability of the program. More recently, Dohi *et al.* [3] used kernel function for reliability prediction. In addition to kernel-based approaches, wavelet-based approaches have also been considered. In order to gain an insight into the essence of the data, Xiao and Dohi [18] proposed Wavelet Shrinkage Estimation (WSE), which can estimate the intensity function with high accuracy. The conclusion of the study shows that within a nonparametric framework, WSE was found to be superior to existing methods in terms of both estimation accuracy and computational speed. Furthermore, by performing wavelet transform, we can transform the data into another dimension, which is expected to reveal hidden behavior in the data. Therefore, several studies have taken advantage of the strengths of WSE and considered prediction

2325-6664/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN-W60302.2024.00017

methods that utilize WSE. Xiao and Dohi [18] presented a one-stage look-ahead prediction which is a short-term prediction technique for WSE. Additionally, Wu *et al.* [14] proposed a multi-stage look-ahead prediction which is a long-term prediction technique expending ideas from one-stage look-ahead prediction. However, these methods forcibly treated predicted values as observed values to achieve the objective. On the other hand, Xiao [15], [16] proposed another approach based on WSE. This approach uses a basic quadratic function to predict the coefficients and predict the number of software faults at future time by composing the predicted coefficients. However, this method has two problems: first, when the number of observed data is small, the number of training data is not sufficient and not suitable for prediction; second, the quadratic function could not sufficiently capture the behavior of the coefficients. In other words, there is still room for improvement in this method in terms of both "accuracy" and "gaining an insight into the true nature of the data".

In this paper, we propose a new prediction method that combines WSE with nonlinear regression. The proposed method splits data into specific data lengths, enabling its application to a small number of data points. By assuming a trigonometric function instead of the quadratic function for wavelet coefficients, we achieve a long-term prediction within the nonparametric framework that can predict software intensity at any time point in the future. These features improve the prediction accuracy and capture the hidden behavior of the data.

This paper consists of 5 sections. Section 2 describes software reliability evaluation using wavelets, and Section 3 proposes a new software fault number prediction method based on WSE. In Section 4, we present the validation and experiments of the proposed method, and Section 5 describes the conclusions and future work.

II. Wavelet-based Software Reliability Assessment

In this section, we explain the wavelet-based software reliability assessment as a preliminary. First, we briefly explain discrete non-homogeneous Poisson process (D-NHPP) based software reliability model. We consider D-NHPP as the underlying stochastic process to describe software fault-detection behavior. Next, we describe WSE, a nonparametric estimation method for software reliability models based on D-NHPP.

A. D-NHPP-based Software Reliability Model

The probability mass function of the cumulative number of software faults $N_i\ (i = 1,\ 2,\ \ldots)$ at discrete testing time i is given by equation (1).

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

Λ_i is a mean value function with discrete time inputs and is the expected value of the cumulative number of faults at time i. In addition,

$$\Lambda_i = \sum_{k=0}^{i} \lambda_k \tag{2}$$

$\lambda_i = E[Y_i] = \Lambda_i - \Lambda_{i-1}\ (i = 0,\ 1,\ 2,\ \ldots)$ is the software intensity function with discrete time inputs and is the expected Y_i which is the number of faults at time i. Where we assume $N_0 = Y_0 = \Lambda_0 = \lambda_0 = 0$ without loss of generality.

B. Wavelet Shrinkage Estimation

We give the Equation (3) by D-NHPP model,

$$Y_i = \lambda_i + \eta_i, \quad i = 0,\ 1,\ \ldots \tag{3}$$

where Y_i is the number of faults at time i, λ_i is the expected number of faults called intensity function, and η_i is the error term (noise). The fundamental idea of WSE is to apply the Haar wavelet to remove the noise and estimate the expected value. WSE is a method to obtain the estimates of λ_i, say $\hat{\lambda}_i$ by removing the noise η_i from the observation $y_i\ (i = 0,\ 1, \ldots)$ of Y_i. Xiao and Dohi [18] proposed a data-transform-based WSE (DT-WSE) to estimate the software intensity function of the D-NHPP-based software reliability model. Roughly speaking, DT-WSE consists of 6 steps.

(a) Data transformation
(b) Wavelet transform
(c) Thresholding
(d) Inverse wavelet transform
(e) Inverse data transformation
(f) Moving Average

Details of step (a) and (e), step (b) and (d), step (c), and step (f) are shown in (1), (2), (3), and (4), respectively.

(1) Data transformation

DT-WSE [18] considers the problem of recovering the underlying software intensity function from the software-fault count data which contains Poisson noise. Since the validity of denoising by the thresholding method is guaranteed when the noise follows a normal distribution, the observed data must be preprocessed. Xiao *et al.* [19] applied various data transformations to investigate their impact on the accuracy of DT-WSE. While there are various data transformations, we use Anscombe transform [1] as an elementary attempt in this paper. The Anscombe transform is based on Equation (4) to obtain the transformed data $s_i\ (i = 1,\ 2, \ldots,\ n)$, i.e.,

$$s_i = 2\sqrt{y_i + \frac{3}{8}}, \tag{4}$$

where $y_i\ (i = 1,\ 2, \ldots,\ n)$ is the observed number of faults detected on the i-th testing date, and n is the number of total testing dates, respectively. Also, the inverse transform is given by Equation (5).

$$y_i = \left(\frac{s_i}{2}\right)^2 - \frac{3}{8}. \tag{5}$$

(2) Wavelet transform

The fundamental concept behind the wavelet approach is that a diverse set of functions can be accurately represented through a wavelet series. In other words, any given function,

$f(t)$, can be effectively approximated using wavelets in an orthonormal manner as Equation (6).

$$f(t) = \sum_{k=-\infty}^{+\infty} c_{j,k}\phi_{j,k}(t) + \sum_{j=0}^{+\infty}\sum_{k=-\infty}^{+\infty} d_{j,k}\psi_{j,k}(t). \quad (6)$$

Here, $c_{j,k}$ and $d_{j,k}$ are the scaling coefficients and the wavelet coefficients, respectively. $\phi_{j,k}(t)$ and $\psi_{j,k}(t)$ are the wavelet bases created by introducing the parameter of scaling j and the parameter of shift k to the scaling function $\phi(t)$ and the wavelet function $\psi(t)$, respectively.

$$\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k). \quad (7)$$

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k). \quad (8)$$

Moreover, DT-WSE uses the Haar scaling function

$$\phi(t) = \begin{cases} 1 \quad (0 \le t < 1) \\ 0 \quad (otherwise), \end{cases} \quad (9)$$

and the Haar wavelet function

$$\psi(t) = \begin{cases} 1 \quad (0 \le t < 1/2) \\ -1 \quad (1/2 \le t < 1) \\ 0 \quad (otherwise). \end{cases} \quad (10)$$

The transformation from $f(t)$ to $c_{j,k}$ and $d_{j,k}$ using Equations (8) and (7) is called the *discrete Haar wavelet transform* (DHWT). In real data analysis, s_i $(i = 1, 2, \dots, n)$ is expanded instead of $f(t)$, and the coefficients are calculated as

$$c_{j,k} = \frac{c_{j+1,2k} + c_{j+1,2k+1}}{\sqrt{2}}, \quad (11)$$

and

$$d_{j,k} = \frac{c_{j+1,2k} - c_{j+1,2k+1}}{\sqrt{2}}, \quad (12)$$

where $j = J-1, J-2, \dots, 0$ and $k = 0, 1, \dots, 2^j - 1$ with $c_{J,k} = \frac{s_i}{\sqrt{n}}$. J is the highest resolution level. Information about the data at resolution j is stored in $c_{j,k}$. Also *inverse discrete Haar wavelet transform* (IDHWT) is given by Equations (13) and (14).

$$c_{j+1,2k} = \frac{c_{j,k} + d_{j,k}}{\sqrt{2}}. \quad (13)$$

$$c_{j+1,2k+1} = \frac{c_{j,k} - d_{j,k}}{\sqrt{2}}. \quad (14)$$

DT-WSE [18] utilizes the DHWT to calculate wavelet coefficients to capture small changes in the analyzed data over short time intervals. A threshold is used to differentiate between noise and the underlying intensity function in the next step.

(3) Thresholding

The thresholding method is a process applied to the wavelet coefficients $d_{j,k}$ to obtain the denoised wavelet coefficients $d'_{j,k}$. There are two thresholding rules; hard thresholding rule and soft thresholding rule. First, the function $\delta^{\mathrm{h}}_{\tau}(u)$, which performs the hard thresholding method with the input value u as the value before thresholding and the output value as the value after thresholding, is defined by

$$\delta^{\mathrm{h}}_{\tau}(u) = u 1_{|u|>\tau}(u), \quad (15)$$

where $1_{|u|>\tau}(u)$ is the indicator function and defined as

$$1_{|u|>\tau}(u) = \begin{cases} 1, & |u| > \tau \\ 0, & \text{otherwise} \end{cases}. \quad (16)$$

On the other hand, the soft threshold method $\delta^{\mathrm{s}}_{\tau}(u)$ is given by

$$\delta^{\mathrm{s}}_{\tau}(u) = \mathrm{sgn}(u)(|u| - \tau)_+, \quad (17)$$

where $\mathrm{sgn}(u)$ is the sign function and defined as

$$\mathrm{sgn}(u) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}, \quad (18)$$

and $(|u| - \tau)_+$ is defined as

$$(|u| - \tau)_+ = \max(0, |u| - \tau) = \begin{cases} |u| - \tau, & |u| - \tau > 0 \\ 0, & \text{otherwise} \end{cases}. \quad (19)$$

In thresholding processing, we have to determine the threshold values. DT-WSE uses universal threshold [4]. The universal threshold (ut) was proposed by Donoho [4] and is calculated using the data length n for Gaussian noise as follows.

$$\tau_{\mathrm{ut}} = \sqrt{2\log n}. \quad (20)$$

In Equations (13) and (14) of the IDHWT, $d'_{j,k}$ can be used instead of $d_{j,k}$ to obtain a tentative estimate, say s'_i. Furthermore, using s'_i instead of s_i in Equation (5) of the inverse data transformation, we can obtain an estimate of the software intensity function, say $\hat{\lambda}_i$.

(4) Moving Average

DT-WSE can be performed only when the number of data is a power of 2 due to the characteristics of the wavelet transform. However, fault number data which is the length of observed data n does not always satisfy this condition. Therefore, the moving average method is used to realize DT-WSE for the arbitrary number of data. We can choose the resolution level J' freely from the range of $1 \le J' \le J$, where $J = \max\left(\{j \mid 2^j \le n\}\right)$. When $n' = 2^{J}$, the number of faults data $y_l, y_{l+1}, \dots, y_{l+n'-1}$ from time l to time $l - n' + 1$ at $l = 1, 2, \dots, n - n' + 1$ is a power of 2. DT-WSE can be performed and the estimates obtained here are $\hat{\lambda}^l{}_1, \hat{\lambda}^l{}_2, \dots, \hat{\lambda}^{l'}{}_n$.The data obtained in this case is the average of the last estimates obtained at the same time, which is the final estimate.

Xiao and Dohi [17] proposed NDT-WSE. This is a method that does not preprocess the Poisson noise contained in the

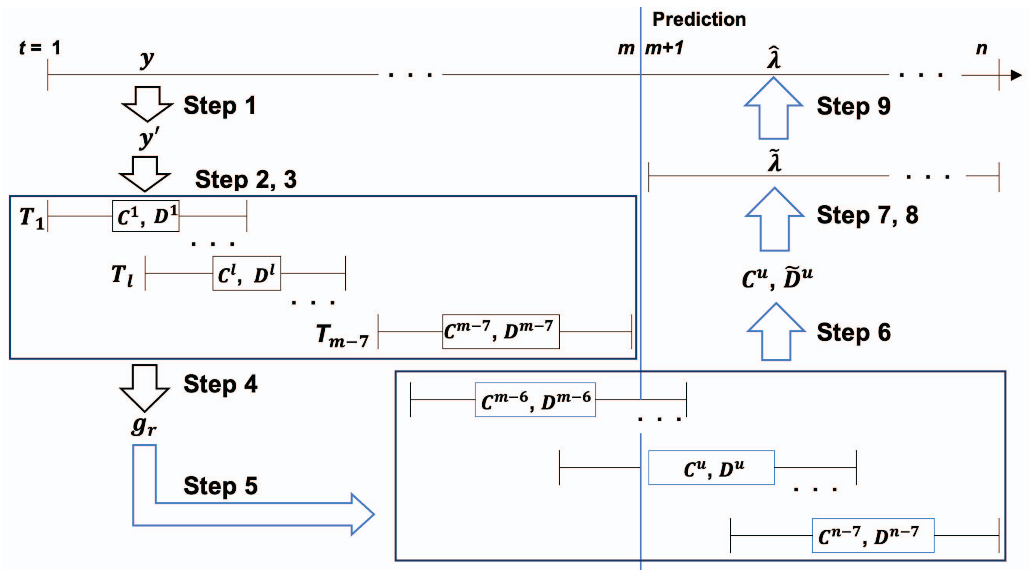


Fig. 1. Concept of our proposal

observed data but performs thresholding as the data. Specifically, NDT-WSE uses Equation (21) instead of Equation (20) in the phase of thresholding. Level-dependent threshold (ldt) was considered by Kolaczyk [7]. It is a threshold that varies with the resolution level j for Poisson noise.

$$\tau_j = 2^{-\frac{J-j+2}{2}}\{2\log(2^j) + \sqrt{4(\log(2^j))^2 + 8\log(2^j)\frac{\lambda_0}{2^j}}\}, \tag{21}$$

where λ_0 is the average of the sample data, and J is the highest resolution level. In addition, phase (1), i.e., steps (a) and (e), are not performed because the data are not preprocessed. That is, $c_{J,k} = \frac{y_i}{\sqrt{n}}$ is set when calculating the coefficients using Equations (11) and (12).

III. Our Proposal

A. The case based on DT-WSE

In this section, we describe the procedure of the proposed method using DT-WSE. Figure 1 shows the flowchart of the proposed method. Here, the training data comprises data from testing date $t = 1$ to $t = m$, and those from $t = m + 1$ to $t = n$ are the data to be predicted ($m < n, where\ m, n \in \mathbb{Z}$).

Step 1. Applying data transformation.

We apply the Anscombe transform to $\boldsymbol{y} = [y_1,\ y_2, \cdots,\ y_m]$, where y_i $(i = 1,\ 2, \ldots,\ m)$ is the number of faults on the $i-$th testing date. The data sequence $\boldsymbol{y'} = [y'_1,\ y'_2, \cdots,\ y'_m]$ are obtained.

Step 2. Applying data splitting.

We divide the data sequence $\boldsymbol{y'}$ into $m-7$ sequential data with a length of $2^{J'}$, where J' is the resolution level of DT-WSE to be determined. Since there is a trade-off between the estimation accuracy and calculation cost, the optimal resolution level must be determined via trial and error depending on the data type. In this paper, $J' = 3$ is adopted, because J'=3 provides a good balance between computational cost and accuracy. The following splitted data $\boldsymbol{T_l}$ $(l = 1,\ 2, \ldots,\ m-7)$ are obtained.

$$\begin{aligned}\boldsymbol{T_1} &= [y'_1,\ y'_2, \cdots,\ y'_8],\\ \boldsymbol{T_2} &= [y'_2,\ y'_3, \cdots,\ y'_9],\\ &\vdots\\ \boldsymbol{T_{m-7}} &= [y'_{m-7},\ y'_{m-6}, \cdots,\ y'_m].\end{aligned}$$

Step 3. Applying wavelet transform.

We perform wavelet transform to $\boldsymbol{T_l}$ to obtain the scaling coefficients $\boldsymbol{C}^l = (c^l_{j,k})$ and wavelet coefficients $\boldsymbol{D}^l = (d^l_{j,k})$, where $j = 0,\ 1, \cdots,\ 3$ and $k = 0,\ 1, \cdots,\ 7$.

Step 4. Creating time-series data.

We produce time-series data for the element of the scaling coefficients $c^l_{0,0}$ and each element of the wavelet coefficients $\boldsymbol{D}^l$. The reason why we need only $c^l_{0,0}$ and $d^l_{j,k}$ is that we can recompose the original data, using inverse wavelet

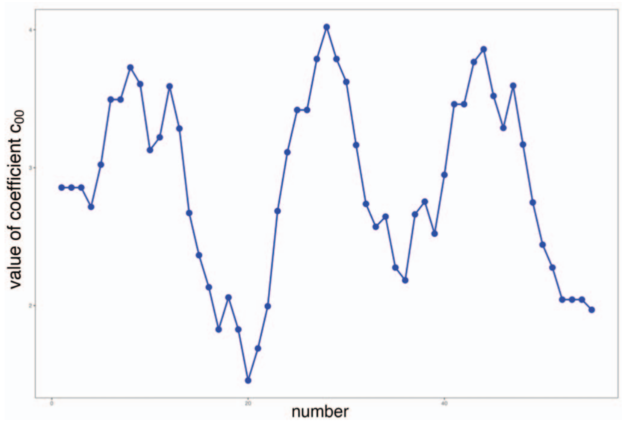


Fig. 2. Time-series data $\boldsymbol{g_1}$

transform with only them. The following time-series data $\boldsymbol{g_r}$ $(r = 1,\ 2, \ldots,\ 8)$ are obtained.

$$
\begin{aligned}
\boldsymbol{g_1} &= [c_{0,0}^{1},\ c_{0,0}^{2}, \cdots,\ c_{0,0}^{m-7}],\\
\boldsymbol{g_2} &= [d_{2,0}^{1},\ d_{2,0}^{2}, \cdots,\ d_{2,0}^{m-7}],\\
&\ \ \vdots\\
\boldsymbol{g_8} &= [d_{0,0}^{1},\ d_{0,0}^{2}, \cdots,\ d_{0,0}^{m-7}].
\end{aligned}
$$

Figure 2 is the graph of the created time series data $\boldsymbol{g_1}$ using DS1 (refer to Table I) as an example.

Step 5. Applying nonlinear regression.

We perform a regression analysis of the time series data $\boldsymbol{g_r}$. In this paper, we use a nonlinear model and choose a periodic function as our first step. The periodic function is defined as

$$g(u) = a\sin(b \times u + c) + d, \tag{22}$$

where the parameters a, b, c, and d are obtained using the least squares method. The predictive coefficient sequence is calculated by substituting $u = m-6,\ m-5, \cdots$, and $n-7$ into the created regression model. Subsequently, the coefficients are recreated to generate predicted scaling coefficients $\boldsymbol{C}^u$ and predicted wavelet coefficients $\boldsymbol{D}^u$.

Step 6. Removing noise.

We perform thresholding on the predicted wavelet coefficients $\boldsymbol{D}^u$ to obtain the denoised wavelet coefficients $\tilde{\boldsymbol{D}}^u$ using universal threshold and both hard and soft thresholding methods.

Step 7. Applying inverse wavelet transform.

Using the inverse wavelet transform, we recreate $\tilde{\boldsymbol{\lambda}}^u$ after the prediction point using the predicted scaling coefficients $\boldsymbol{C}^u$ and the denoised predicted wavelet coefficients $\tilde{\boldsymbol{D}}^u$. The following $\tilde{\boldsymbol{\lambda}}^u$ $(u = m-6,\ m-5, \cdots,\ n-7)$ are obtained.

$$
\begin{aligned}
\tilde{\boldsymbol{\lambda}}^{m-6} &= [\tilde{\lambda}_{m+1}^{m-6}],\\
\tilde{\boldsymbol{\lambda}}^{m-5} &= [\tilde{\lambda}_{m+1}^{m-5},\ \tilde{\lambda}_{m+2}^{m-5}],\\
&\ \ \vdots\\
\tilde{\boldsymbol{\lambda}}^{n-7} &= [\tilde{\lambda}_{n-7}^{n-7},\ \tilde{\lambda}_{n-6}^{n-7}, \ldots,\ \tilde{\lambda}_{n}^{n-7}].
\end{aligned}
$$

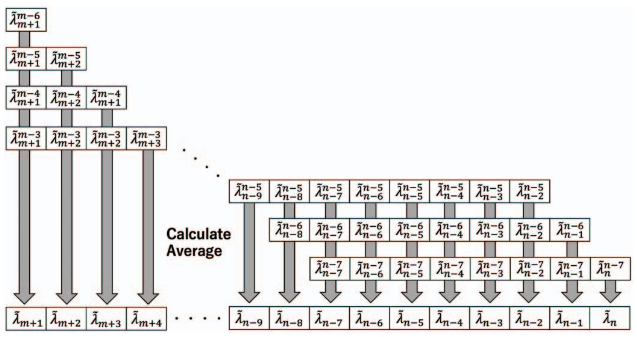


Fig. 3. Concept of moving average

Step 8. Applying moving average.

We take an average over the same testing period of the predictions obtained in Step 7 to obtain $\tilde{\boldsymbol{\lambda}} = [\tilde{\lambda}_{m+1},\ \tilde{\lambda}_{m+2}, \ldots,\ \tilde{\lambda}_{n}]$. Figure 3 shows the concept of moving average for better understanding.

Step 9. Applying inverse data transformation.

Finally, we apply the inverse Anscombe transform to $\tilde{\boldsymbol{\lambda}}$ to predict the final value $\hat{\boldsymbol{\lambda}} = [\hat{\lambda}_{m+1},\ \hat{\lambda}_{m+2}, \ldots, \hat{\lambda}_{n}]$.

B. The case based on NDT-WSE

The proposed method using NDT-WSE has two differences from that using DT-WSE. The first difference is that the data transformation of Step. 1 and inverse data transformation of Step. 9 are not performed. The second difference is that the thresholding method of Step. 6 is different. The level-dependent threshold (ldt) [7] instead of the universal threshold (ut) [4] is used. To summarize, our proposal has 4 variations as follows.

- No data transformation, level-dependent threshold, hard threshold (ndt, h)
- No data transformation, level-dependent threshold, soft threshold (ndt, s)
- Anscombe transform, universal threshold, hard threshold (dt, h)
- Anscombe transform, universal threshold, soft threshold (dt, s)

IV. Experiment

In this section, we verify and test the properties and prediction accuracy of the proposed method. Additionally, we compare the predictions obtained using the proposed and previously reported methods. The fault-count datasets are taken from [11] [12] [8] (DS1 - DS6) for experiments. Four of the six datasets are based on software fault count data presented in the paper [8]. In addition, we use the software fault count data from Retro video game emulation for macOS [11] and A web-based tool for Spriting and Pixel art [12] and

TABLE I
INFORMATION OF SOFTWARE-FAULT COUNT DATA

	Period	Total number	System
DS1	62 weeks	133	Aerospacecraft system
DS2	41 weeks	351	Spacecraft system
DS3	73 weeks	367	Spacecraft system
DS4	81 days	461	Brazilian electronic switching system
DS5	100 months	368	Retro video game emulation for macOS
DS6	59 months	80	A web-based tool for Spriting and Pixel art

denote them as DS5 and DS6. The information on the dataset is shown in Table I.

Datasets with the top 70%, 50%, and 30% testing date (rounded down to the nearest decimal place) are used for validation (test data), while the remaining data are used as training data. Predictive mean absolute error (PMAE) is used to evaluate the prediction accuracy, which is defined as

$$\text{PMAE} = \frac{\sum_{i=m+1}^{n} |y_i - \hat{\lambda}_i|}{n-m}, \quad (23)$$

where n is the number of total data, m is the number of training data, and y_i and $\hat{\lambda}_i$ are the observed and predicted numbers of faults on the $i-$th testing date, respectively.

The experimental environment and tools used in the experiments are as follows.

- OS: MacOS Sonoma 14.2
- CPU: M2 Max 3.5GHz 12 core
- RAM: 96.00 GB
- R: 4.3.1
- Regression function: nls function in R
- Regression algorithm: Gauss-Newton algorithm

A. Comparison with variations of the proposal method

Here, we investigate the impact of different data transformations and threshold methods on prediction accuracy. We experimented with the combinations presented in Section III-B. These combinations can be freely chosen by the user and affect the predictions. We will investigate the consequences of these differences on the accuracy of the predictions. The algorithm used in this regression, the Gaussian-Newton algorithm, has a regression accuracy that depends on the initial values. Therefore, in this experiment, each parameter from a to d was given a value from 1 to 10 with 0.5 increments, and the one with the highest regression accuracy was adopted in the regression model.

Tables II, III and IV show the PMAE of the proposed method in 70%, 50% and 30% training data, respectively. The best PMAE in each dataset is shown in red. In Table II, both (ndt, h) and (ndt, s) show good prediction accuracy for DS1, DS2, and DS4, while (dt, h) and (dt, s) show good accuracy for the other datasets. We assume that this indicates better prediction accuracy for the correct assumption due to

TABLE II
THE PMAE OF THE PROPOSED METHOD IN 70% TRAINING DATA

	(ndt, h)	(ndt, s)	(dt, h)	(dt, s)
DS1	1.51	1.51	1.65	1.65
DS2	10.72	10.72	10.83	10.83
DS3	1.65	1.65	1.39	1.39
DS4	2.47	2.47	2.54	2.54
DS5	64.94	65.17	14.31	14.31
DS6	1.01	1.01	0.95	0.95

TABLE III
THE PMAE OF THE PROPOSED METHOD IN 50% TRAINING DATA

	(ndt, h)	(ndt, s)	(dt, h)	(dt, s)
DS1	1.72	1.72	1.71	1.71
DS2	7.99	7.99	8.00	8.00
DS3	1.69	1.69	1.68	1.68
DS4	5.56	5.56	2.75	2.75
DS5	24.95	24.75	4.88	4.88
DS6	1.10	1.10	0.90	0.90

TABLE IV
THE PMAE OF THE PROPOSED METHOD IN 30% TRAINING DATA

	(ndt, h)	(ndt, s)	(dt, h)	(dt, s)
DS1	1.86	1.86	1.84	1.84
DS2	370.07	368.20	370.29	370.29
DS3	2.76	2.76	2.67	2.67
DS4	6.04	6.04	6.14	6.14
DS5	34.64	32.96	6.76	6.76
DS6	37.55	37.75	19.05	19.05

the large amount of training data. On the other hand, Table III and IV show that (dt, h) and (dt, s) show good prediction accuracy for many datasets. Figure 4 shows the time series data of the regressed coefficients (dt, 70%, DS1). The blue line shows the coefficient values and the red line shows the regression results. From Figure 4, the amount of training data and the variation in the coefficient values have a significant impact on the regression results. Since the amount of training data is small for medium-term and long-term predictions, it is considered that the regression with data transformation is more accurate because the variation in coefficient values is reduced when data transformation is performed. Tables II, III and IV show the same PMAE for the soft and hard threshold methods except for DS2 and DS5. Figure 5 shows

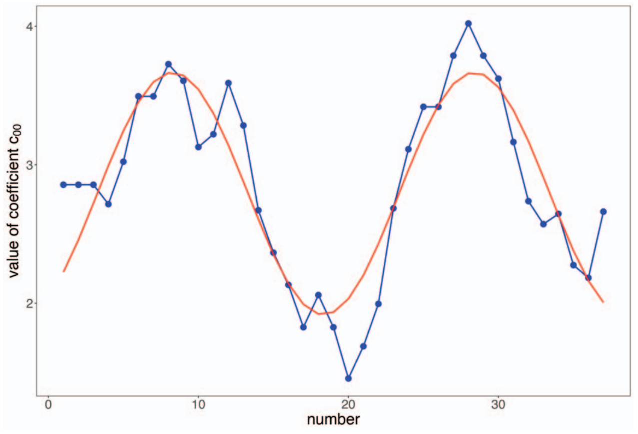


Fig. 4. Time series data $c_{0,0}$

the prediction result in DS1 and Figure 5 shows that the predictions of the soft threshold method are almost the same as those of the hard threshold method, and it can be said that the wavelet component is removed in both methods. We can see that thresholding is performed on most of the datasets. Based on these considerations, we consider that the prediction using data transformation is the most versatile, regardless of the threshold method.

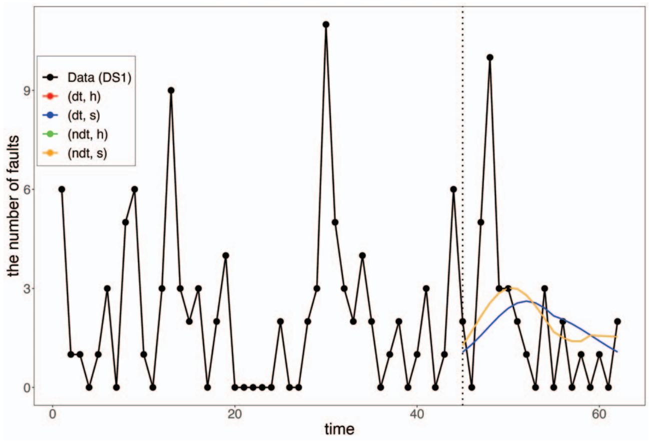
B. Comparison with the conventional methods

Here, we compare the prediction accuracy of the proposed method with that of the conventional method. The comparison is made among the three methods: the prediction using the parametric model [9], the one-stage look-ahead method proposed by Wu et al [14]. and the conventional method proposed by Xia et [15], [16]. The three methods are named as the following.

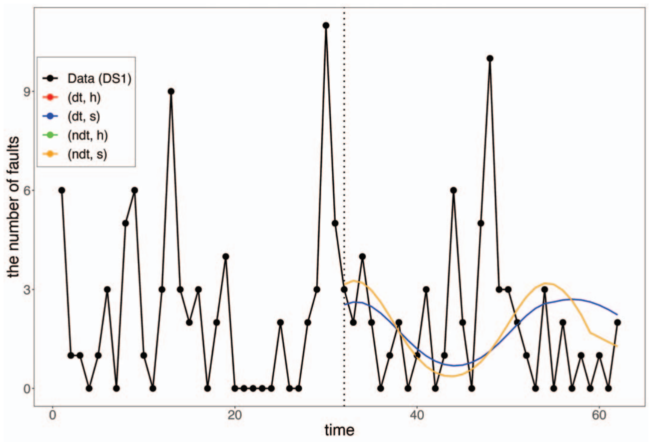
- Method A: Parametric prediction (SRATS [9])
- Method B: Multi-stage look-ahead prediction (Wu [14])
- Method C: Quadratic-based prediction (Xiao [15], [16])

Since many variations exist among the conventional methods, the most accurate combination of each method was used for comparison.

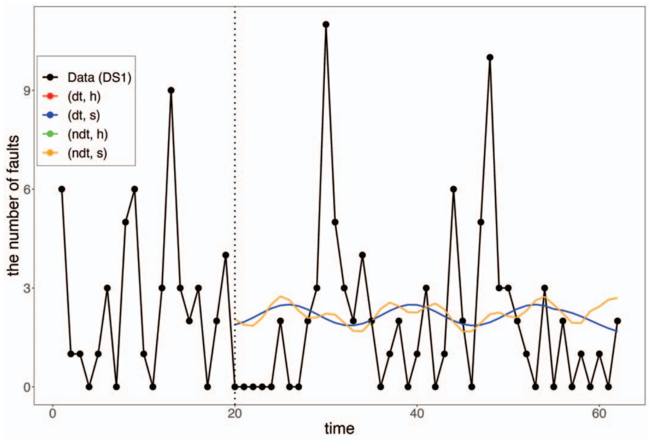
Table VI shows the PMAE of the proposed method, Method A (multi-stage look-ahead prediction [14]), and Method B (quadratic function-based prediction [16]). The symbol "-" in Table VI means Method B could not be executed due to an insufficient number of data. The proposed method shows a large PMAE for most of the datasets. Also, the predictions tend to be large even under the condition of correct regression. This can be considered to be because the threshold value is large or the predicted coefficients are too large to perform the thresholding process. Tables V, VI, and VII show that the proposed method does not show better PMAE than other methods for most datasets and prediction periods. However, Method A is a parametric method and requires the selection of the most accurate model among many models. In addition,



(a) 70% training data



(b) 50% training data



(c) 30% training data

Fig. 5. The result of proposal method in DS1

TABLE V
The PMAE in 70% training data
(vs Method A, Method B and Method C)

	DS1	DS2	DS3	DS4	DS5	DS6
Method A	1.60	1.29	1.14	2.31	1.18	0.80
Method B	1.71	2.36	1.23	1.52	1.23	0.84
Method C	1.62	48.82	1.13	2.41	7.57	1.17
Proposal Method	1.51	10.72	1.39	2.47	14.31	0.95

TABLE VI
The PMAE in 50% training data
(vs Method A, Method B and Method C)

	DS1	DS2	DS3	DS4	DS5	DS6
Method A	1.51	6.71	1.66	2.14	1.22	0.87
Method B	1.73	3.31	1.76	2.90	1.24	1.58
Method C	1.94	-	4.63	2.44	1.68	28.20
Proposal Method	1.72	8.00	1.68	2.75	4.88	0.90

TABLE VII
The PMAE in 30% training data
(vs Method A, Method B and Method C)

	DS1	DS2	DS3	DS4	DS5	DS6
Method A	1.84	4.72	1.71	3.66	1.31	1.41
Method B	1.83	5.20	1.79	2.66	1.28	1.38
Method C	-	-	-	148.70	39.11	-
Proposal Method	1.86	368.20	2.67	6.04	6.76	19.05

Tables V, VI, and VII show that Method C does not yield predictions for all datasets and prediction periods. This indicates that Method C is not a good prediction method. Looking at the methods other than Method A and Method B, Table VI confirms that the PMAE of the proposed method is smaller than that of the conventional methods in four of six datasets. Moreover, Method B uses predictions, so the predictions are different at the same time for different prediction periods. For these reasons, we believe that the proposed method is superior to the other methods. Additionally, the execution time of our proposed method is about 30 seconds, which is faster than Method B on some datasets.

V. Conclusion and Future Work

This paper proposed a new prediction method to overcome the disadvantages of nonparametric methods using WSE to predict the number of software faults. Our proposal is a direct prediction method, while the existing method [14] predicts an arbitrary time using the predicted value as the observed value. In addition, we are able to capture the hidden behavior of the data. Experimental results show that the proposed method is comparable to the existing method [14] in terms of accuracy. From the perspective of calculating cost, the proposed method is more efficient than the existing method [14] in some datasets.

Many types of WSE and regression models are available, and the optimal model for each dataset may differ. Therefore, it is necessary to investigate the impact of these factors on prediction accuracy in the future. Additionally, comparison with the kernal-based approaches [2], [3] and NN-based approach [6], [13] will also be a future challenge.

References

[1] F. J. Anscombe, “The transformation of Poisson, binomial and negative-binomial data,” *Biometrika*, vol. 35, no. 3–4, pp. 246–254, 1948.
[2] M. Barghout, B. Littlewood, and A. Abdel-Ghaly, “A non-parametric order statistics software reliability model,” *Software Testing Verification and Reliability*, vol. 8, pp. 113–132, 1998.
[3] T. Dohi, J. Zheng, and H. Okamura, “Data-driven software reliability evaluation under incomplete knowledge on fault count distribution,” *Quality Engineering*, vol. 32, 2020.
[4] D. L. Donoho and J. M. Johnstone, “Ideal spatial adaptation by wavelet shrinkage,” *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.
[5] A. L. Goel and K. Okumoto, “Time-dependent error-detection rate model for software reliability and other performance measures,” *IEEE transactions on Reliability*, vol. 28, no. 3, pp. 206–211, 1979.
[6] N. Karunanithi, D. Whitley, and Y. K. Malaiya, “Using neural networks in reliability prediction,” *IEEE Software*, vol. 9, no. 4, pp. 53–59, 1992.
[7] E. D. Kolaczyk, “Nonparametric estimation of gammaray burst intensities using haar wavelets,” *The Astrophysical Journal*, vol. 483, no. 1, pp. 654–659, 1997.
[8] M. R. Lyu, Ed., *Handbook of Software Reliability Engineering*. New York, USA: Computing McGraw-Hill, 1996.
[9] H. Okamura and T. Dohi, “SRATS: Software reliability assessment tool on spreadsheet (experience report),” *Proceedings of IEEE 24th International Symposium on Software Reliability Engineering* (ISSRE 2013), pp. 100–107, 2013.
[10] H. Okamura, T. Dohi, and S. Osaki, “Software reliability growth models with normal failure time distributions,” *Reliability Engineering & System Safety*, vol. 116, pp. 135–141, 2013.
[11] OpenEmu, URL: https://github.com/OpenEmu/OpenEmu(Access date: 2023-12-11).
[12] Piskel, URL: https://github.com/piskelapp/piskel(Access date: 2023-12-11).
[13] Y. S. Su and C. Y. Huang, “Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models,” *Journal of Systems and Software*, vol. 80, 2007.
[14] J. Wu, T. Dohi, and H. Okamura, “W-SRAT: Wavelet-based software reliability assessment tool,” *Proceedings of the 21st International Conference on Software Quality, Reliability and Security (QRS2021)*, pp. 564–573, 2021.
[15] X. Xiao, “Haar wavelet regression model for nhpp-based software reliability assessment,” *Proceedings of the 9th International Conference on Mathematical Methods in Reliability*, pp. pp. 56–63, 2015.
[16] ——, “Software intensity function prediction by haar wavelet regression,” *Supplemental Proceedings of the 15th International Conference on Software Quality, Reliability & Security (QRS2015)*, pp. 182–183, 2015.
[17] X. Xiao and T. Dohi, “Estimating software intensity function based on translation-invariant Poisson smoothing approach,” *IEEE Transactions on Reliability*, vol. 62, no. 4, pp. 930–945, 2013.
[18] ——, “Wavelet shrinkage estimation for non-homogeneous Poisson process based software reliability models,” *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 211–225, 2013.
[19] X. Xiao, D. Tada, and H. Yamamoto, “On the role of unbiased inverse variance-stabilizing transformation in wavelet shrinkage estimation for nhpp-based software reliability assessment,” *International Journal of Industrial Engineering : Theory Applications and Practice*, vol. 26, 2019.
[20] S. Yamada, M. Ohba, and S. Osaki, “S-shaped reliability growth modeling for software error detection,” *IEEE Transactions on reliability*, vol. 32, no. 5, pp. 475–484, 1983.