


## Article

# Modeling Software Reliability with Learning and Fatigue

Tahere Yaghoobi <sup>1,\*</sup> and Man-Fai Leung <sup>2</sup> 

<sup>1</sup> Department of Computer Engineering and Information Technology, Payame Noor University, Tehran 19395-4697, Iran

<sup>2</sup> School of Computing and Information Science, Faculty of Science and Engineering, Anglia Ruskin University, Cambridge CB1 1PT, UK; man-fai.leung@aru.ac.uk

\* Correspondence: t.yaghoobi@pnu.ac.ir

**Abstract:** Software reliability growth models (SRGMs) based on the non-homogeneous Poisson process have played a significant role in predicting the number of remaining errors in software, enhancing software reliability. Software errors are commonly attributed to the mental errors of software developers, which necessitate timely detection and resolution. However, it has been observed that the human error-making mechanism is influenced by factors such as learning and fatigue. In this paper, we address the issue of integrating the fatigue factor of software testers into the learning process during debugging, leading to the development of more realistic SRGMs. The first model represents the software tester's learning phenomenon using the tangent hyperbolic function, while the second model utilizes an exponential function. An exponential decay function models fatigue. We investigate the behavior of our proposed models by comparing them with similar SRGMs, including two corresponding models in which the fatigue factor is removed. Through analysis, we assess our models' quality of fit, predictive power, and accuracy. The experimental results demonstrate that the model of tangent hyperbolic learning with fatigue outperforms the existing ones regarding fit, predictive power, or accuracy. By incorporating the fatigue factor, the models provide a more comprehensive and realistic depiction of software reliability.

**Keywords:** software reliability growth model; non-homogeneous Poisson process; learning curve; fatigue; imperfect debugging

**MSC:** 68M15



**Citation:** Yaghoobi, T.; Leung, M.-F. Modeling Software Reliability with Learning and Fatigue. *Mathematics* **2023**, *11*, 3491. <https://doi.org/10.3390/math11163491>

Academic Editor: Vassilis C. Gerogiannis

Received: 28 June 2023

Revised: 8 August 2023

Accepted: 10 August 2023

Published: 13 August 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Due to the ubiquitous use of software in our daily lives, accurately predicting the number of software errors has become crucial, particularly in critical applications. Software reliability growth models (SRGMs) based on the non-homogeneous Poisson process (NHPP) have emerged as widely adopted tools for this purpose [1] (Pham, 2006). These models allow for the numerical estimation of the remaining errors in software and provide insights into its reliability. To address the complexities of the software development process, SRGMs have evolved to incorporate various factors, including the experience, skill, and learning of software developers [2].

Research has highlighted the significant impact of fatigue on the human error-making process [3]. In particular, studies have demonstrated that fatigue can trigger attention switching in individuals, typically occurring after approximately 40 min of continuous activity. This fatigue-induced attention shift is attributed to a gradual reduction in dopamine secretion, eventually reaching a threshold that disrupts attention. Furthermore, it has been observed that other neurotransmitters cannot adequately compensate for the decline in dopamine release. To capture this phenomenon, researchers have modeled the decrease in dopamine secretion rate as an exponential decay process towards a specific limit [3].

Software debugging is the process of identifying and removing errors or defects in a computer program, which affects software reliability. Achieving perfect software debugging is often challenging and may not always be possible due to the inherent complexity of software development. The goal is to minimize bugs and deliver a high-quality product by employing best practices and continuously improving the development and debugging processes. In imperfect debugging, software testers inadvertently introduce new faults during the debugging process. Whether the debugging process is perfect or imperfect can be influenced by various human-related and non-human-related factors, such as the experience and skill of debuggers, debugging tools, program size and complexity, testing strategies, and environmental factors [4]. We believe that learning and fatigue are two human-related factors that can significantly impact software debugging, influencing the efficiency and effectiveness of the process. The reason is that developers familiar with the codebase, understand the software's logic and expected behavior, and possess domain knowledge to understand the intricacies and potential pitfalls can do more efficient debugging.

On the other hand, debugging requires sustained attention and focus, as developers need to analyze code, identify patterns, and devise solutions. Fatigue can lead to reduced concentration, making it easier to overlook critical details or commit errors during debugging. Debugging can be time-consuming and sometimes frustrating, especially when dealing with complex bugs. Fatigue may reduce a developer's patience and persistence, potentially resulting in prematurely abandoning the problem-solving process or the hasty application of inadequate fixes. In both cases, learning and fatigue can work hand in hand. New developers or those less familiar with the codebase may experience increased fatigue as they need to invest more effort in understanding the code and identifying issues. Conversely, fatigue can hinder the learning process, making it more challenging for developers to absorb new information (experience) or gain deeper insights into the software.

This research delves into a new specific aspect of imperfect debugging, i.e., the impact of tester fatigue on the debugging process. We assume these imperfections can stem from attention-switching problems caused by tester fatigue. Understanding that fatigue can lead to attention-switching problems and subsequently introduce new defects is crucial for creating more accurate representations of real-world scenarios. This research introduces two SRGMs involving human-related factors of tester learning and fatigue. The first model represents the software tester's learning phenomenon via the tangent hyperbolic ( $\tanh$ ) function, while the second model utilizes an exponential function. We investigate the behavior of our proposed models by comparing them with similar SRGMs, including the corresponding two perfect software reliability models that do not consider the effect of the fatigue factor. We estimate the models' parameters and assess their fit, predictive abilities, and accuracy using three datasets to validate them.

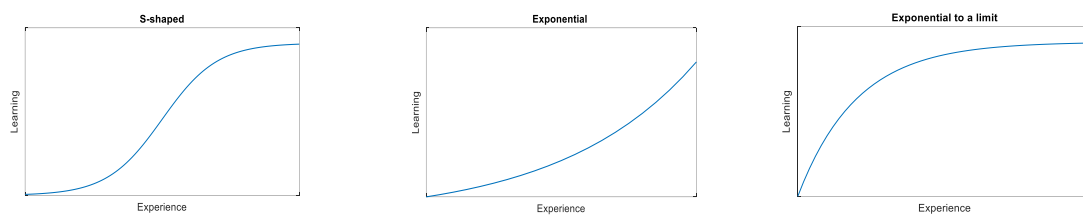
Section 2 of this paper focuses on reviewing the relevant literature and exploring previous works in the field. Section 3 introduces the mathematical formulations of our proposed models, which are based on a general framework of a family of SRGMs. Section 4 presents numerical examples to illustrate the application and performance of the models. To gain a deeper understanding of the proposed models, Section 5 conducts a sensitivity analysis, providing valuable insights into their behavior and critical parameters. Finally, Section 6 concludes this paper, summarizing the main findings and highlighting our research contributions.

## 2. Literature Review

### 2.1. Learning Curves

Learning refers to acquiring new knowledge, skills, or understanding, and a learning curve visually represents the relationship between skill level, expertise, and the time required to complete a task. Mathematically, learning can be described using various functions, each representing different improvement patterns over time. Three typical learning curves are S-shaped, exponential, and exponential growth to a limit. The S-shaped learning curve demonstrates the initial exponential growth, followed by a period of slower

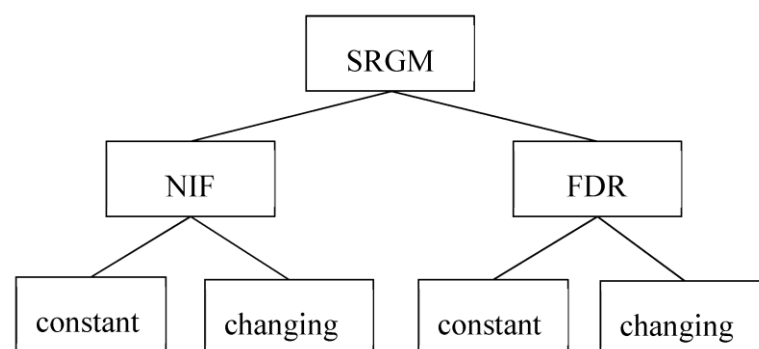
growth and ultimately approaching a maximum upper limit that is never fully reached. The logistic function commonly describes an S-shaped learning curve, also known as the sigmoid curve. The exponential learning curve illustrates a slow rate of progress at the beginning, gradually increasing over time until full proficiency is achieved. Unlike the S-shaped curve, the exponential learning curve suggests that learning can improve indefinitely without limits. The exponential growth to a limit learning curve indicates that initial attempts result in rapid skill acquisition or information retention, reaching a maximum rate and approaching a maximum upper limit. However, perfection or significant improvement in the skill may not occur with subsequent repetitions. Figure 1 represents three standard types of learning curves.



**Figure 1.** Learning curves of S-shaped, exponential, and exponential to a limit.

## 2.2. Related Works

Over the past few decades, researchers have made significant advancements in developing software reliability growth models by exploring various ideas and approaches. One notable contribution in this field is the work of Pham and Nordmann, who introduced a general framework for constructing new SRGMs [5]. This framework has served as a foundation for interpreting several existing software reliability models. Within this framework, two concepts play vital roles in the construction of an SRGM: the expected number of initial faults (NIF) present in the software at the beginning of the testing phase and the fault detection rate (FDR), which represents the rate at which failures are detected over time. In the context of software debugging, both NIF and FDR can be treated as either constant or varying in a time-dependent manner. Figure 2 categorizes this group of SRGMs based on whether the NIF and FDR are considered constant or subject to change. This figure helps to provide a clearer understanding of the different models within this family.



**Figure 2.** A classification of some SRGMs.

In models with constant NIF, it is assumed that when a fault is detected, it is immediately removed by the testers, and no new errors are introduced in the process. Consequently, the software's initial defects remain unchanged throughout the debugging phase. On the other hand, in software reliability models with changing NIF, it is acknowledged that new faults may be introduced during the testing phase. This means that the total number of defects in the software is not constant and comprises both the initial faults and the additional faults introduced during the debugging process. This assumption recognizes the possibility of testers unintentionally introducing new errors while attempting to fix existing defects.

The FDR is a significant indicator of the effectiveness of the testing phase. It is influenced by various factors, including the expertise of testers, testing techniques employed, and the selection of test cases. The FDR can remain constant or vary among faults depending on the software reliability model. In the case of a constant FDR, it is assumed that all defects in the software have an equal probability of being detected throughout the testing period. This implies that the FDR remains consistent over time, irrespective of the specific characteristics of the faults.

Conversely, in models with a time-dependent FDR, the function may exhibit increasing or decreasing trends as time progresses. This variation acknowledges the dynamic nature of the testing process, where the effectiveness of fault detection can be influenced by factors such as the testing team's expertise, the program's size, and the software's testability. By incorporating the concept of a changing FDR, software reliability models can better reflect the complexities and uncertainties inherent in real-world testing scenarios. Recognizing the dependence of the FDR on various factors enables researchers to develop more accurate models and gain deeper insights into the dynamics of software reliability assessment.

#### A. SRGMs with constant NIF and constant/changing FDR

The Goel–Okumoto model [6] is a widely referenced example of an NHPP model with constant NIF and FDR. More SRGMs with constant NIF and changing FDR have been proposed in the literature. These models consider learning phenomena, time resources, testing coverage, and environmental uncertainties. Yamada et al. [7] introduced the concept of a learning process in software testing, where testers gradually improve their skills and familiarity with the software products. They formulated an increasing FDR with a hyperbolic function to represent the learning rate of testers and proposed the delayed S-shaped model. Ohba [8] considered the learning process of testers during the testing phase and defined the FDR using a non-decreasing logistic S-shaped curve, leading to the development of the inflection S-shaped model. Yamada and Osaki [9] considered the consumption of time resources and proposed the exponential testing effort and Rayleigh testing effort models. Pham [1] introduced the imperfect fault detection (IFD) model, which incorporates a changing FDR that combines fault introduction with the phenomenon of testing coverage. This model allows for a more realistic representation of the testing process. Song et al. [10] considered the impact of testing coverage uncertainty or randomness in the operating environment. They proposed a new NHPP software reliability model with constant NIF and changing FDR regarding a testing coverage function, considering the uncertainty associated with operational environments.

#### B. SRGMs with changing NIF and constant/changing FDR

More SRGMs with time-dependent changing NIF function and constant/changing FDR have been proposed in the literature. For example, Yamada et al. [11] proposed two imperfect debugging models assuming the NIF function to be an exponential or linear function of the testing time, respectively, and FDR to be constant. Pham and Zhang [12] developed an imperfect debugging model considering an exponential function of testing time for NIF and a non-decreasing S-shaped function for FDR. Pham et al. [13] proposed an imperfect SRGM with NIF function to be linear and FDR S-shaped of the testing time. Li and Pham [14] introduced a new, changing NIF model, and FDR is expressed as a testing coverage function. In their model, they also assumed that when a software failure is detected, immediate debugging starts, and either the total number of faults is reduced by one with probability  $p$  or the total number of faults remains the same with probability  $1-p$ .

#### C. Other SRGMs

Many imperfect SRGMs do not fit the above framework precisely and use other approaches. For example, Chiu et al. [15] proposed a model that considers the influential factors for finding errors in software, including the autonomous errors-detected and learning factors. They proposed an FDR function including two factors representing the exponential-shaped and the S-shaped types of behaviors. Iqbal et al. [16] investigated

the impact of two learning effect factors: autonomous and acquired learning, which are gained after repeated experience/observation of the testing/debugging process by the tester/debugger in an SRGM. Wang et al. [17] proposed an imperfect software debugging model that considers a log-logistic distribution function for NIF, which can capture the increasing and decreasing characteristics of the fault introduction rate per fault. They reason imperfect software debugging models proposed in the literature generally assume a constantly or monotonically decreasing fault introduction rate per fault. These models cannot adequately describe the fault introduction process in a practical test. Wang and Wu [18] proposed a nonlinear NHPP imperfect software debugging model by considering that fault introduction is a nonlinear process. Al-Turk and Al-Mutairi [19] developed an SRGM based on one-parameter Lindley distribution, which is modified by integrating two learning effects of the autonomous error-detected factor and the learning factor. These studies highlight the ongoing efforts to refine SRGMs by considering real-world scenarios and addressing the critical aspects of the software testing and debugging processes. Huang et al. [20] developed an NHPP model considering both human factors (learning effect of the debugging process) and the nature of errors, such as varieties of errors and change points, during the testing period to extend the practicability of SRGMs. Verma et al. [21] proposed an SRGM by considering conditions of error generation, fault removal efficiency (FRE), imperfect debugging parameter, and fault reduction factor (FRF). The error generation, imperfect debugging, and FRE parameters have been assumed to be constant, while FRF is time dependent and modeled using exponential, Weibull, and delayed s-shaped distribution functions. Luo et al. [22] recently proposed a new SRGM with a changing NIF and FDR represented by an exponential decay function of testing time.

Each category of SRGMs has its own set of advantages and disadvantages. On one end of the spectrum, SRGMs with a changing NIF and FDR tend to have more parameters, as they incorporate various assumptions to yield a more realistic representation of the underlying processes. However, this realism comes at the cost of increased complexity. Complex models may require more resources, such as time and memory, to appropriately evaluate. While the abundance of parameters offers flexibility, it also leads to higher computational overhead.

In contrast, SRGMs with a constant NIF and FDR follow a simpler approach, resulting in fewer parameters and more straightforward models. A simpler model is generally easier to comprehend, interpret, and implement. Despite potentially sacrificing some level of realism, the simplicity of such models can prove advantageous, especially when computational efficiency and ease of use are significant considerations.

### 3. Development of New NHPP Software Reliability Models

This study focuses on modeling SRGMs with a constant NIF and time-dependent FDR function. This choice has two reasons: (1) To gain a deeper insight into how the new time-dependent FDR affects the model's behavior. By focusing on the FDR function, we aim to understand its implications in software reliability analysis. (2) Simplicity is another objective of this approach. Employing a constant NIF makes the resulting model more straightforward to interpret. Simpler models are often favored for their ease of implementation and comprehensibility.

The mean value function,  $m(t)$ , for the class of NHPP-SRGMs with a constant NIF and time-dependent FDR function, can be obtained by solving the following differential equation:

$$\frac{dm(t)}{dt} = r(t) \cdot [a - m(t)] \quad \text{with } m(0) = 0 \quad (1)$$

in which  $a > 0$  is the NIF, i.e., the number of defects in the software at the beginning of the test, and  $r(t)$  is a time-dependent FDR function that denotes the rate of discovering new faults in software over the testing. The SRGM defined via Equation (1) is based on the following assumptions: (1) a non-homogeneous Poisson process can describe the fault removal process; (2) the faults that remained in the software caused system failures

at random times; (3) the mean number of detected faults is proportional to the mean number of remaining faults in the system. By introducing various functions for  $r(t)$ , which can be interpreted as different assumptions made, the mathematical expression for  $m(t)$  can be derived. For example, when  $r(t) = b$ , then  $m(t) = a[1 - \exp(-bt)]$ , which is the GO model [6].

Now, we propose new models based on Equation (1) by considering the following functions for  $r(t)$ :

1. The combination of tanh learning with fatigue;
2. The combination of exponential learning with fatigue;
3. Tanh learning without fatigue;
4. Exponential learning without fatigue.

This study analyzes two learning curves: one based on the tanh function and the other based on the exponential function. The objective is to determine which curve more accurately captures the actual learning behavior in the context of this research. Unlike previous studies that have usually used an S-shaped curve for modeling  $r(t)$ , this research introduces a novel approach by adopting the  $\tanh(t)$  function, where  $t \geq 0$ , which exhibits an exponential-to-limit behavior for learning. Furthermore, this study explores the integration of this new learning curve with the fatigue phenomenon to model  $r(t)$ . The behavior of the two proposed models is also investigated when the fatigue factor is removed from the models.

In model NEW1, we assume  $r(t)$  represents a weighted combination of the tanh learning with the fatigue of the tester as follows:

$$r(t) = \alpha \cdot \tanh(st) + \beta \cdot e^{-wt} \quad (2)$$

Parameters  $s$  and  $w$  represent the learning and fatigue rates, respectively.  $\alpha$  and  $\beta$  are positive coefficients representing the weights of each factor. By substituting Equation (2) in Equation (1) and solving the resulting differential equation, the mathematical form of the mean value function of the NEW1 model is obtained as follows:

$$m(t) = a[1 - e^{\frac{\beta(e^{-wt}-1)}{w}} \cosh^{\frac{-\alpha}{s}}(st)] \quad (3)$$

This model assumes that each time a failure is observed, the failure is removed, and new faults can be introduced due to fatigue.

In model NEW2, we assume  $r(t)$  is the combination (for simplicity, average) of exponential learning and fatigue.

$$r(t) = k \cdot \cosh(st) \quad (4)$$

Parameter  $s$  represents an equal rate of learning and fatigue, and  $k$  is a weight. By substituting Equation (4) in Equation (1) and solving the resulting differential equation, the mathematical form of the mean value function of model NEW2 is obtained as follows:

$$m(t) = a \left[ 1 - e^{\frac{-k \sinh(st)}{s}} \right] \quad (5)$$

In model NEW3, only the tanh learning function without the fatigue factor is considered for  $r(t)$  as follows:

$$r(t) = k \cdot \tanh(st) \quad (6)$$

By substituting Equation (6) in Equation (1) and solving the resulting differential equation, the mathematical form of the mean value function of model NEW3 is obtained as follows:

$$m(t) = a[1 - \cosh^{\frac{-k}{s}}(st)] \quad (7)$$

In model NEW4, only the exponential learning function without the fatigue factor is considered for  $r(t)$  as follows:

$$r(t) = k \cdot e^{st} \quad (8)$$

By substituting Equation (8) in Equation (1) and solving the resulting differential equation, the mathematical form of the mean value function of model NEW4 is obtained as follows:

$$m(t) = a[1 - e^{\frac{k(1-e^{st})}{s}}] \quad (9)$$

#### 4. Numerical Examples

Our experiments specifically considered SRGMs that align with this modeling framework, featuring constant NIF and either constant or changing FDR. Table 1 summarizes the characteristics of the similar existing SRGMs and the proposed models used in this study.

**Table 1.** Characteristics of SRGMs used in this study.

Model	$m(t)$	$r(t)$	Comments
Goel-Okumoto (GO)	$a(1 - e^{-bt})$	$b$	Constant FDR [6]
Delayed S-shaped (DS)	$a[1 - (1 + bt)e^{-bt}]$	$\frac{b^2t}{1+bt}$	Increasing FDR with a hyperbolic function [7]
Inflection S-shaped (IS)	$\frac{a(1-e^{-bt})}{1+ce^{-bt}}$	$\frac{b}{1+ce^{-bt}}$	Increasing FDR with a two-parameter logistic function [8]
Yamada Exponential (YE)	$a[1 - e^{-ra(1-e^{-\beta t})}]$	$r \cdot \alpha \beta e^{-\beta t}$	Proportional to the exponential testing effort function [9]
Yamada Rayleigh (YR)	$a[1 - e^{-\alpha r(1-e^{-\frac{\beta t^2}{2}})}]$	$r \cdot \alpha \beta t e^{-\beta \frac{t^2}{2}}$	Proportional to the Rayleigh testing effort function [9]
IFD	$a - ae^{-bt}[1 + (b + d)t + bdt^2]$	$\frac{b^2t}{1+bt} - \frac{d}{1+dt}$	Combination of a testing coverage with a fault introduction rate function [1]
SCP	$a[1 - \frac{\beta}{\beta + bt - \ln(1+dt)}]^\alpha$	$\frac{\eta(-d + (1+dt)b)}{(1+dt)e^{-bt}}$	Testing coverage with the uncertainty of the operating environment ( $\eta$ has a generalized probability density function with two parameters $\alpha$ and $\beta$ .) [10]
NEW1	$a[1 - e^{\frac{\beta(e^{-wt}-1)}{w}} \cosh \frac{-\alpha}{s}(st)]$	$\alpha \cdot \tanh(st) + \beta \cdot e^{-wt}$	Combination of tanh learning with fatigue (Current study)
NEW2	$a[1 - e^{\frac{-ksinh(st)}{s}}]$	$k \cdot \cosh(st)$	Average of exponential learning with fatigue (Current study)
NEW3	$a[1 - \cosh \frac{-k}{s}(st)]$	$k \cdot \tanh(st)$	Tanh learning (Current study)
NEW4	$a[1 - e^{\frac{k(1-e^{st})}{s}}]$	$k \cdot e^{st}$	Exponential learning (Current study)

##### 4.1. Descriptions of the Datasets

Three datasets from different real software projects have been used to study our proposed models' fitting and predictive ability, validate our approaches, and compare them with similar ones. The first dataset (DS1) is Release 1 of the Tandem Computers Software Data Project. Over 20 weeks, 100 faults were detected [23]. This dataset is frequently used in the literature. The second dataset (DS2) was obtained from a real-time command and control system. During 25 h, 136 faults were detected [1]. The third dataset (DS3) was collected from a wireless network switching system. Over 34 weeks, 181 defects were detected [24]. Table 2 briefly describes three datasets used in this study.

**Table 2.** Summary of the selected failure data sets.

Data Set		Testing Period	Cumulative Number of Failures
DS1	Tandem Computer Software	20 weeks	100
DS2	Real-time Command and Control System	25 h	136
DS3	Wireless Network System	34 days	181

#### 4.2. Criteria for Model Comparison

We employed three criteria to compare and illustrate the models' fitting, predictive capabilities, and accuracy. These criteria were chosen to provide comprehensive evaluations of the models' performance. The three criteria used are outlined as follows.

##### Criterion 1. (A measure of fit)

The mean squared error (MSE) is a widely used criterion to assess the adequacy of a software reliability model's fit. Given a dataset consisting of pairs of observed failure times  $(t_i, m_i)$  for  $i = 1, 2, \dots, k$ , where  $k$  represents the total number of observations in the dataset, the MSE quantifies the discrepancy between the predicted values of the model and the corresponding actual data. Mathematically, the MSE is defined as follows:

$$\text{MSE} = \frac{1}{k} \sum_{i=1}^k [m_i - m(t_i)]^2 \quad (10)$$

$m_i$  denotes the cumulated number of actual software failures found until the time  $t_i$ , and  $m(t_i)$  is the model estimate for the cumulated number of failures discovered at the time  $t_i$ . A smaller value of the MSE criterion represents a minor error in fitting and therefore indicates a better model performance.

##### Criterion 2. (A measure of prediction)

The predictive ability of a software reliability growth model refers to its capability to predict future and unseen software failure data based on the observed failure data. The predictive ratio risk (PRR) is a criterion to assess the model's prediction accuracy. It quantifies the discrepancy between the model's estimations and the actual observations. The PRR is calculated as follows [25]:

$$\text{PRR} = \sum_{i=1}^k \left[ \frac{m(t_i) - m_i}{m(t_i)} \right]^2 \quad (11)$$

A smaller PRR indicates a better performance of the model.

##### Criterion 3. (A measure of accuracy)

Theil's statistic (TS) measures accuracy, assessing the deviation between the actual values and the model's predictions across all periods. It is calculated as the average deviation and is defined as follows:

$$TS = \sqrt{\frac{\sum_{i=1}^k (m(t_i) - m_i)^2}{\sum_{i=1}^k m_i^2}} \quad (12)$$

A closer TS to zero indicates better accuracy of the model.

#### 4.3. Comparisons

To compare the proposed models' fitting, predictive, and accuracy with other models, we divided the datasets into two subsets: 80% and 20%. The 80% subset was used to estimate the parameters of the models using the least-square error method. These estimated parameter values were then applied to the 80% subset to calculate the mean square error (MSE\_fit) values. The estimated parameter values were also applied to the remaining 20% of the datasets to calculate the predictive ratio risk (PRR\_predict) values. Finally, the

estimated parameter values were used for the entire period of collected failure data to calculate Theil's statistic (TS) values, which measure accuracy.

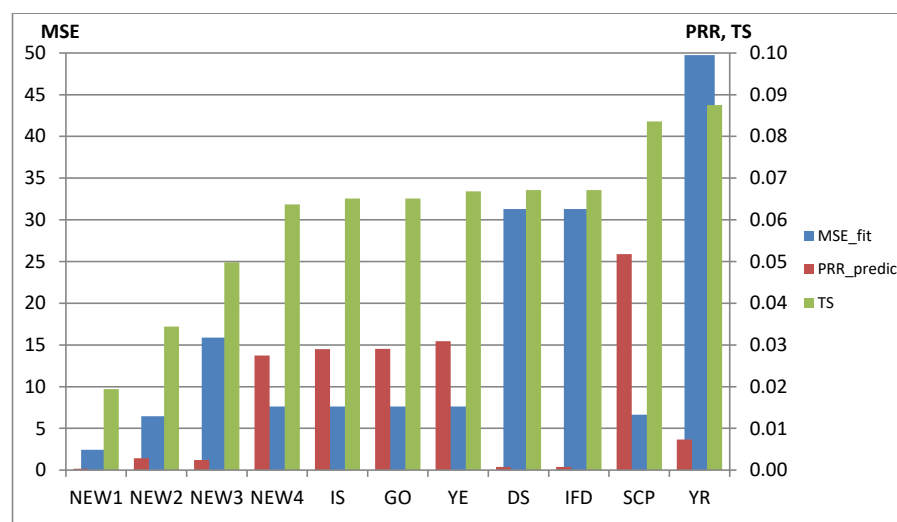
(1) DS1 (Tandem dataset).

Table 3 displays the optimal parameter values for each SRGM and the corresponding values obtained via the MSE\_fit, PRR\_predict, and TS criteria using the DS1 dataset.

**Table 3.** Obtained results using DS1.

Model	MSE_Fit	PRR_Predict	TS	Parameters
GO	7.6246	0.029036	0.065128	$a = 158.7887, b = 0.0624$
DS	31.296	0.00074	0.067117	$a = 103.0886, b = 0.2684$
IS	7.6247	0.02902	0.065114	$a = 158.7224, b = 0.0625, c = 0.001$
YE	7.6286	0.0309	0.066846	$a = 178.2258, r = 0.01, \alpha = 560.4812, \beta = 0.01$
YR	49.735	0.007346	0.087492	$a = 99.5568, r = 0.01, \alpha = 398.3805, \beta = 0.01$
IFD	31.299	0.00074	0.067122	$a = 103.0871, b = 0.2684, d = 0.00001$
SCP	6.6326	0.051762	0.083567	$a = 443.5951, b = 611.657, d = 7.727, \alpha = 0.1, \beta = 0.8235$
NEW1	2.4346	0.000321	0.019395	$a = 102.4245, s = 0.0001, w = 955.9065, \alpha = 225.0417, \beta = 192.0749$
NEW2	6.4589	0.0028346	0.034402	$a = 104.4743, k = 0.0954, s = 0.12998$
NEW3	15.887	0.0023959	0.049836	$a = 121.2902, k = 0.1, s = 2.0804$
NEW4	7.6233	0.027477	0.063675	$a = 149.458, k = 0.0659, s = 0.0064$

Figure 3 represents the values obtained from Table 3 in a combo chart.



**Figure 3.** Combo chart representing MSE, PRR, and TS values for SRGMs using DS1.

Based on the fitting ability (MSE\_fit), the NEW1 model, which incorporates tanh learning and fatigue, demonstrates the highest fitting level to the DS1 dataset. Regarding predictive power (PRR\_predict), the NEW1 model exhibits minimal prediction errors and outperforms other models. When considering the measure of accuracy (TS), the NEW1 model emerges as the most precise. Additionally, the other proposed models exhibit commendable performance compared to competing models.

A comparative analysis of the four proposed models shows that the NEW1 model (incorporating tanh learning with fatigue) outperforms its counterparts across all three evaluation criteria. Concerning the models' fitting ability, NEW2 (employing exponential

learning with fatigue) exhibits superior performance compared to NEW4 (utilizing exponential learning alone), followed by NEW3 (applying tanh learning). Regarding predictive power, NEW3 slightly surpasses NEW2, while NEW4 demonstrates the least favorable predictive performance. Regarding accuracy, NEW2 outperforms NEW3, followed by NEW4 as the least accurate model.

Table 4 presents the estimated number of defects projected in the proposed models.

**Table 4.** Comparison of the estimated defects by the new models using DS1.

Testing Time (Weeks)	Defects Found	Estimated Defects by the NEW1 Model	Estimated Defects by the NEW2 Model	Estimated Defects by the NEW3 Model	Estimated Defects by the NEW4 Model
1	16	20	10	8	10
2	24	22	18	19	19
3	27	27	27	28	27
4	33	32	34	37	35
5	41	39	42	45	43
6	49	47	49	52	50
7	54	54	56	59	56
8	58	62	62	65	62
9	69	69	69	70	68
10	75	75	74	75	74
11	81	81	80	80	79
12	86	86	85	84	84
13	90	90	89	87	88
14	93	93	93	90	93
15	96	96	96	93	97
16	98	98	99	96	100
17	99	99	101	98	104
18	100	100	102	101	107
19	100	101	103	103	110
20	100	101	104	104	113

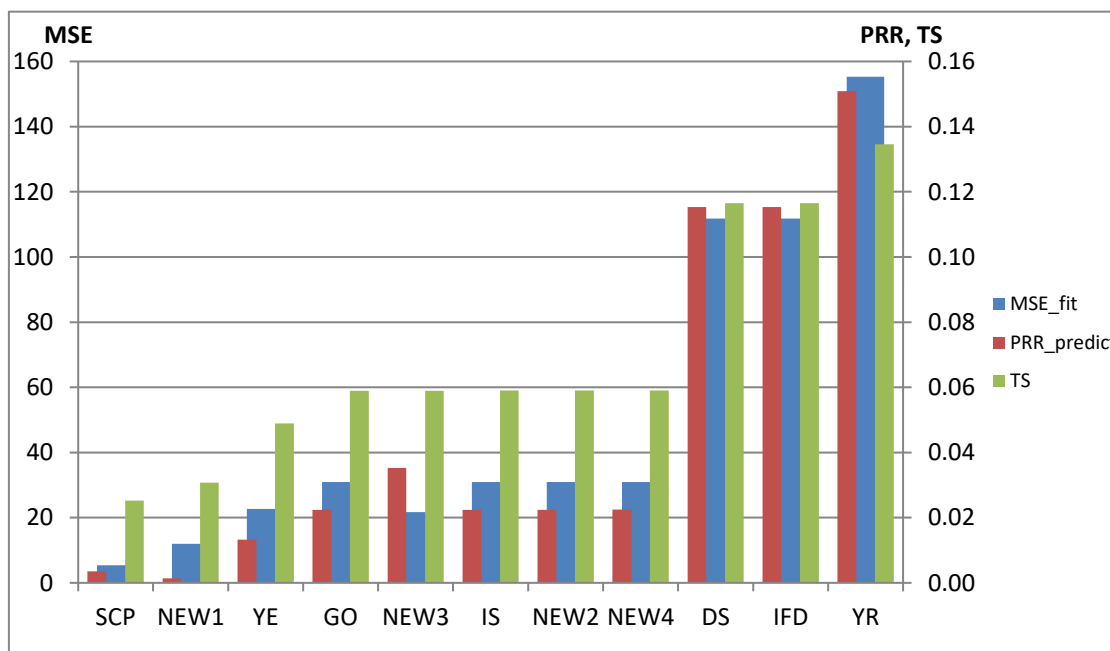
## (2) DS2 (Real-time and Command dataset)

Table 5 displays the optimal parameter values for each SRGM and the corresponding values obtained via the MSE\_fit, PRR\_predict, and TS criteria using the DS2 dataset.

**Table 5.** Obtained results using DS2.

Model	MSE_Fit	PRR_Predict	TS	Parameters
GO	30.905	0.02243	0.0589	$a = 128.9073, b = 0.156$
DS	111.77	0.11536	0.1165	$a = 116.1565, b = 0.418$
IS	30.909	0.022436	0.0590	$a = 128.9051, b = 0.156, c = 0.00017$
YE	22.711	0.01326	0.0489	$a = 183.9655, r = 0.09, \alpha = 14.9442, \beta = 0.09$
YR	155.36	0.15089	0.1346	$a = 116.9754, r = 0.025, \alpha = 145.7865, \beta = 0.025$
IFD	111.77	0.11537	0.1165	$a = 116.1561, b = 0.418, d = 0.00001$
SCP	5.4306	0.003572	0.025238	$a = 433.4581, b = 1000, d = 2.5449, \alpha = 4.9218, \beta = 0.38695$
NEW1	11.986	0.0013361	0.0307	$a = 143.9516, s = 15.59059, w = 873.0036, \alpha = 0.1, \beta = 138.7975$
NEW2	30.905	0.022431	0.0590	$a = 128.9058, k = 0.156, s = 0.001$
NEW3	21.675	0.035237	0.05892	$a = 123.4075, k = 0.169, s = 47.3651$
NEW4	30.941	0.022493	0.0590	$a = 128.8742, k = 0.156, s = 0.00015$

Figure 4 represents the values obtained from Table 5 in a combo chart.



**Figure 4.** Combo chart representing MSE, PRR, and TS values for SRGMs using DS2.

Based on the fitting ability (MSE\_fit), the SCP model demonstrates the best fit, followed by the NEW1 model for the DS2 dataset. Regarding predictive power (PRR\_predict), the NEW1 model exhibits the lowest prediction errors, while the SCP model ranks second. Regarding accuracy (TS), the NEW1 model is the second-most accurate model after SCP. The other proposed models exhibit satisfactory performance and outperform the DS, IFD, and YR models.

A comparative analysis of the four proposed models shows that the NEW1 model (incorporating tanh learning with fatigue) outperforms its counterparts across all three evaluation criteria. NEW2 (employing exponential learning with fatigue) and NEW4 (utilizing exponential learning alone) exhibit considerable similarity in their performance across all three criteria. Meanwhile, the NEW3 model, which applies tanh learning, showcases distinct characteristics compared to NEW2 and NEW4. It notably excels in fitting ability; however, its predictive power falls behind that of NEW2 and NEW4, suggesting that NEW3 might be less accurate in making future predictions. Nevertheless, in terms of accuracy, NEW3 performs similarly to NEW2 and NEW4, implying that all three models yield comparable levels of correctness in their predictions.

Table 6 presents the estimated number of defects for the proposed models.

### (3) DS3 (Wireless network system dataset)

Table 7 displays the optimal parameter values for each SRGM and the corresponding values obtained via the MSE\_fit, PRR\_predict, and TS criteria using the DS3 dataset.

Based on the fitting ability (MSE\_fit), the NEW1 model best fits the DS3 dataset. Regarding predictive power (PRR\_predict), the NEW1 model exhibits the lowest prediction errors. Regarding accuracy (TS), the NEW1 model is the most accurate. The other proposed models also exhibit satisfactory performance among their competitors.

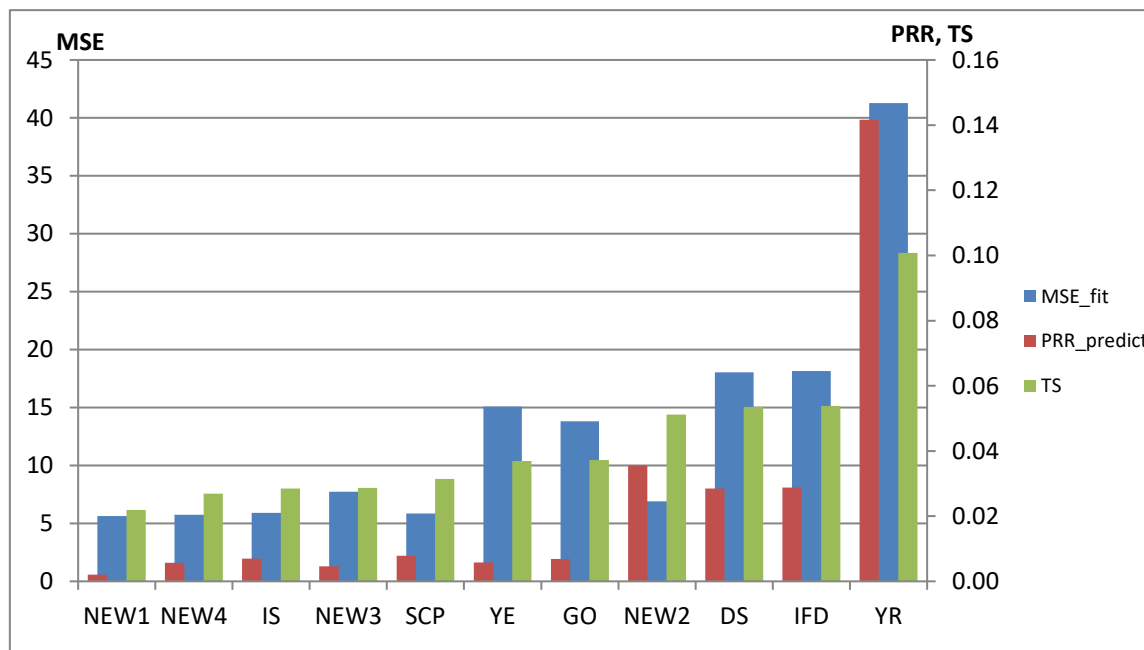
**Table 6.** Comparison of the estimated defects by the new models using DS2.

Testing Time (Hours)	Defects Found	Estimated Defects by the NEW1 Model	Estimated Defects by the NEW2 Model	Estimated Defects by the NEW3 Model	Estimated Defects by the NEW4 Model
1	27	32	19	19	19
2	43	43	35	35	35
3	54	53	48	49	48
4	64	61	60	61	60
5	75	69	70	70	70
6	82	76	78	79	78
7	84	83	86	86	86
8	89	89	92	91	92
9	92	94	97	96	97
10	93	99	102	101	102
11	97	103	106	104	106
12	104	107	109	107	109
13	106	110	112	110	112
14	111	114	114	112	114
15	116	116	117	114	117
16	122	119	118	123	118
17	122	121	120	123	120
18	127	124	121	123	121
19	128	126	122	123	122
20	129	127	123	123	123
21	131	129	124	123	124
22	132	130	125	123	125
23	134	132	125	123	125
24	135	133	126	123	126
25	136	134	126	123	126

**Table 7.** The obtained results using DS3.

Model	MSE_Fit	PRR_Predict	TS	Parameters
GO	13.823	0.0068311	0.0372	$a = 5724.2965, b = 0.001$
DS	18.05	0.028493	0.0535	$a = 201.7278, b = 0.0977$
IS	5.912	0.0070125	0.028491	$a = 208.1097, b = 0.1, c = 4.097$
YE	15.072	0.005775	0.0369	$a = 2989.2663, r = 0.1523, \alpha = 84.755, \beta = 0.00015$
YR	41.288	0.14166	0.1008	$a = 156.15498, r = 0.2652, \alpha = 18396.9465, \beta = 0.0000015$
IFD	18.163	0.028824	0.0538	$a = 201.4796, b = 0.098, d = 0.0001$
SCP	5.8568	0.0078277	0.031415	$a = 964.07144, b = 18.663, d = 0.3086, \alpha = 1.268, \beta = 1622.9581$
NEW1	5.6262	0.0020131	0.0219	$a = 242.957, s = 0.017, w = 0.017, \alpha = 0.1057, \beta = 0.017$
NEW2	6.8976	0.035603	0.0512	$a = 166.1322, k = 0.0322, s = 0.1017$
NEW3	7.7425	0.0046374	0.028717	$a = 685.335, k = 0.01, s = 0.4079$
NEW4	5.754	0.0056439	0.0269	$a = 187.6476, k = 0.0237, s = 0.063$

Figure 5 represents the values obtained from Table 7 in a combo chart.



**Figure 5.** Combo chart representing MSE, PRR, and TS values for SRGMs using DS3.

In a comparative analysis of the four proposed models, compelling evidence emerges, clearly showcasing the superiority of the NEW1 model (integrating tanh learning with fatigue) over its counterparts across all three evaluation criteria. Conversely, NEW2 (employing exponential learning with fatigue) exhibits the least favorable performance among all models, showcasing inferior results across all three criteria. Furthermore, NEW4 (utilizing exponential learning exclusively) demonstrates advantages over NEW3 (applying tanh learning) regarding fitting ability and accuracy. However, it falls short compared to NEW3 regarding predictive power, implying that NEW3 possesses a better capability to make accurate future predictions.

Table 8 presents the estimated number of defects for the proposed models.

**Table 8.** Comparison of the estimated defects by the proposed models using DS3.

Testing Time (Days)	Defects Found	Estimated Defects by the NEW1 Model	Estimated Defects by the NEW2 Model	Estimated Defects by the NEW3 Model	Estimated Defects by the NEW4 Model
1	5	4	5	1	5
2	6	9	10	5	9
3	13	14	16	10	14
4	22	19	21	16	19
5	24	24	26	23	24
6	29	29	31	29	30
7	34	35	36	35	35
8	40	41	41	42	41
9	46	47	47	48	47
10	53	53	53	55	53
11	63	59	58	61	59

Table 8. *Cont.*

Testing Time (Days)	Defects Found	Estimated Defects by the NEW1 Model	Estimated Defects by the NEW2 Model	Estimated Defects by the NEW3 Model	Estimated Defects by the NEW4 Model
12	70	65	64	67	65
13	71	72	70	73	71
14	74	78	77	79	78
15	78	84	83	85	84
16	90	90	90	91	90
17	98	97	96	97	97
18	105	103	103	103	103
19	110	109	109	109	109
20	117	115	116	115	115
21	123	121	122	120	121
22	128	127	128	126	127
23	130	132	133	131	133
24	136	138	139	137	138
25	141	143	144	142	143
26	148	149	148	148	148
27	156	154	152	153	153
28	164	159	155	158	157
29	166	163	158	164	161
30	169	168	160	169	165
31	170	172	162	174	168
32	176	177	163	179	172
33	180	181	164	184	174
34	181	185	165	189	177

#### 4.4. Threats to the Validity

In this section, we address potential limitations to the generalizability of our findings. These limitations primarily concern the applicability of our models in industrial settings. Although our experiments utilized three real datasets to demonstrate the performance of the proposed models, it is essential to acknowledge that the results may vary across specific applications. The reason is that software reliability models rely on the failure dataset; thus, no single model is suitable for every application. Furthermore, the choice of criteria and models used in the experiments is another issue that may impact the outcomes. We selected three comparison criteria and seven competitor models based on previous software reliability studies that align with our approach. We recommend using additional criteria and expanding the set of candidate models for evaluation and comparison to select the most suitable software reliability model for a specific application. Expanding the evaluation's scope can give a more comprehensive understanding of the models' performance.

#### 5. Sensitivity Analysis

A scientific model can be likened to a black box that takes inputs and produces corresponding outputs. In the case of a mathematical model, sensitivity analysis is employed to assess the impact of changes in input values on the model's outputs. Sensitivity analysis serves various purposes, including prioritizing model inputs to identify the critical drivers of model behavior. It also provides insights into the stability of inputs. Sensitivity plots

visualize how the model's output changes when the inputs are modified within predetermined small ranges. This information is valuable for managers, decision-makers, or analysts as it offers insights into the problem. In one-way sensitivity analysis, inputs are varied individually around a selected value of interest, and the variations can be minor. By systematically adjusting the parameter values, we gained insights into the model's response to parameter changes and identified the parameters significantly impacting the model's behavior.

To assess the sensitivity and stability of the NEW1 model, we conducted a one-way sensitivity analysis by modifying a single parameter while keeping all other parameters fixed. This analysis aimed to identify which model parameters are sensitive to changes and which are more stable. Specifically, we examined how variations in the estimated parameter values obtained from Tables 3, 5 and 7, ranging from  $-40\%$  to  $+40\%$  at  $20\%$  intervals, affect the estimated mean value function of the NEW1 model.

In Figures 6–8, we present the results of a sensitivity analysis performed on all five parameters of the NEW1 model, utilizing DS1-DS3 datasets. These figures display the mean value function,  $m(t)$ , for the NEW1 model. Within each figure, we vary one parameter value, as represented in the corresponding plots, while keeping the remaining parameters fixed, following the details in Tables 3, 5 and 7. These figures provide insights into the impact of parameter variations on the cumulative number of expected faults. It is evident from Figures 6–8 that among all parameters of the NEW1 model, the predicted number of initial defects, represented by the parameter “a”, plays a critical role in driving the behavior of the proposed model. Parameter changes “a” result in noticeable variations in the model's output for all datasets.

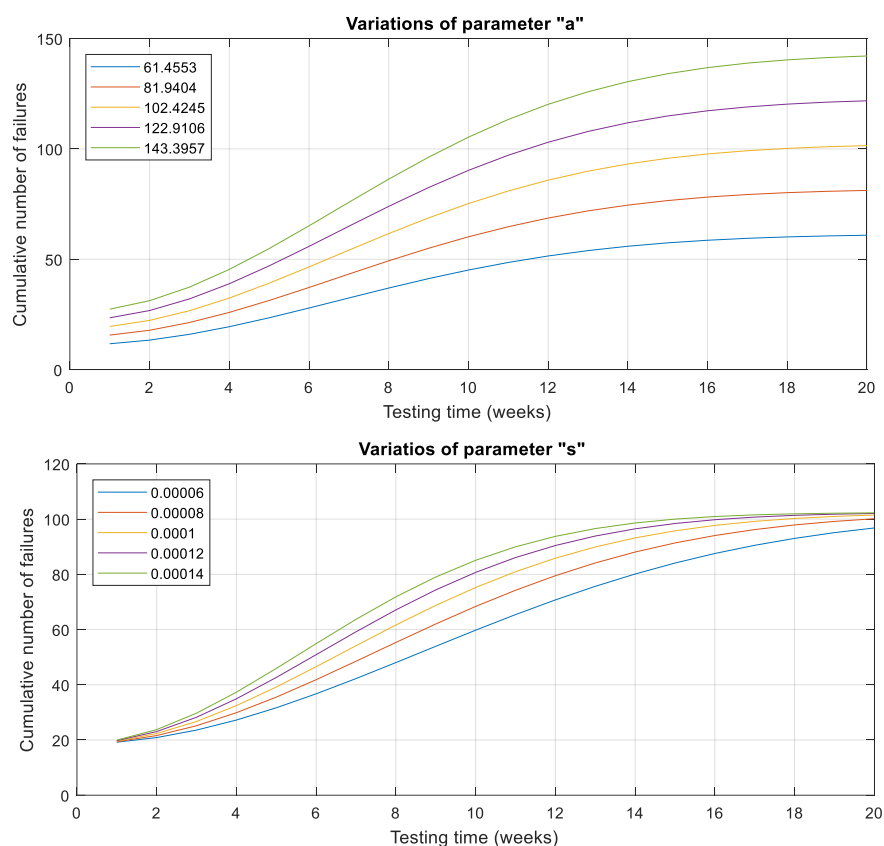


Figure 6. Cont.

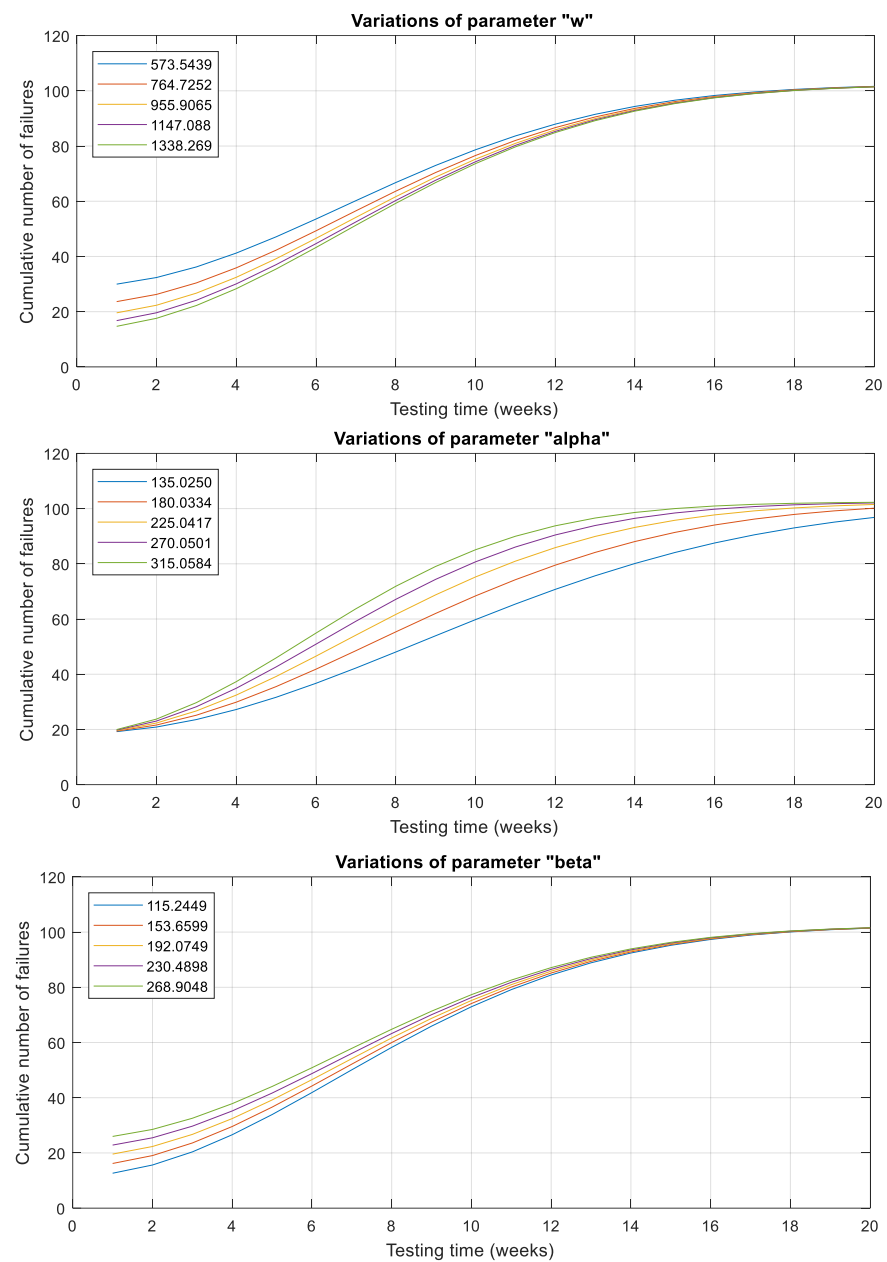


Figure 6. Sensitivity analysis plots for the parameters of the NEW1 model using DS1.

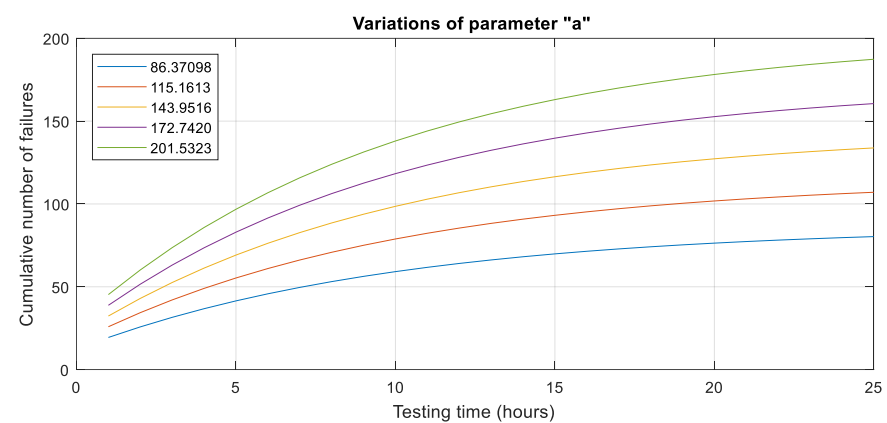
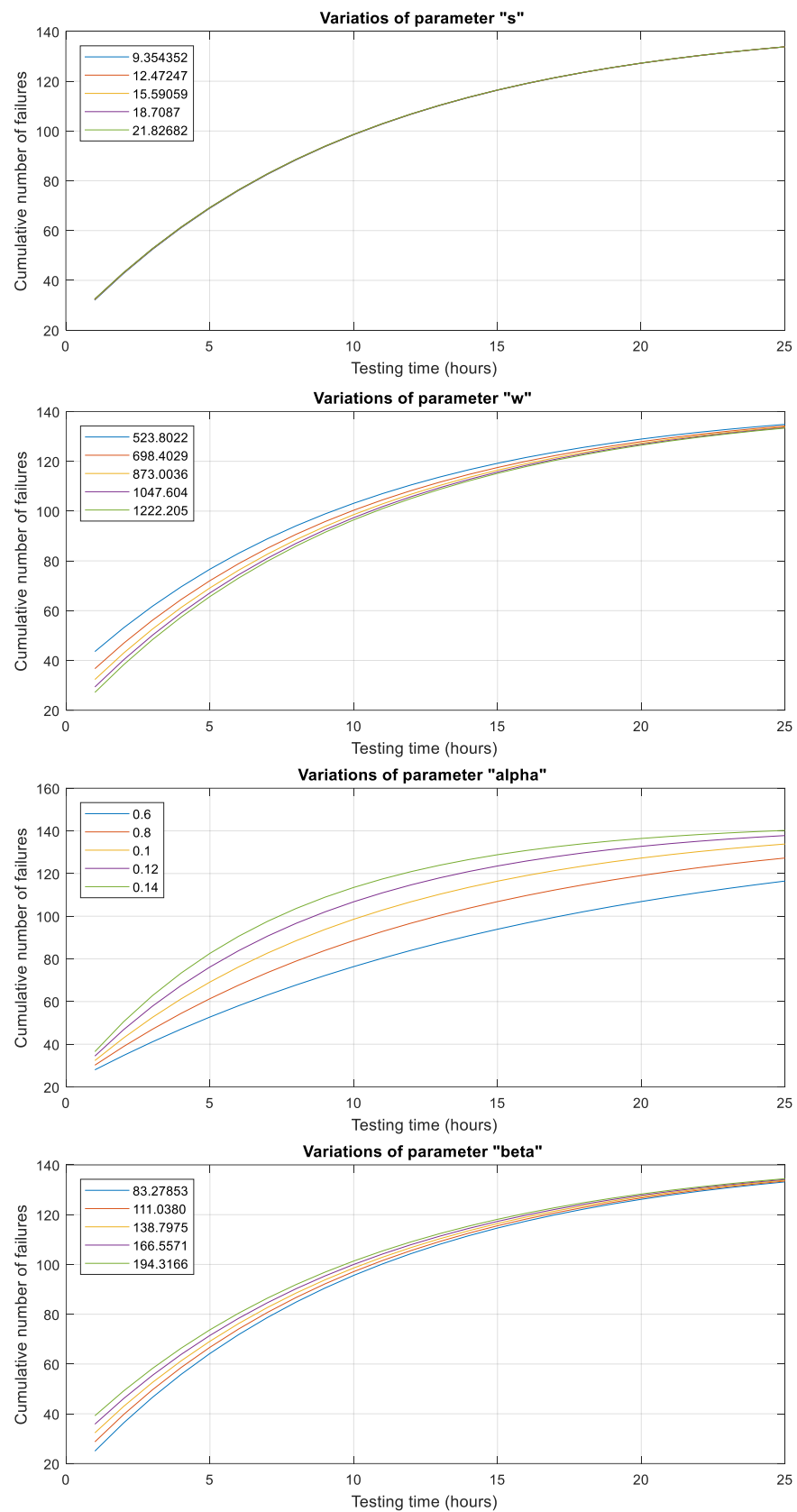


Figure 7. Cont.



**Figure 7.** Sensitivity analysis plots for the parameters of the NEW1 model using DS2.

Figure 6 also reveals that slight changes in parameter "s", corresponding to the learning rate, lead to slight changes in the model's output. Parameter "w", corresponding to the

fatigue factor, remains stable, indicating that the model's output is less sensitive to these parameter changes. Similarly, slight changes in the value " $\alpha$ " lead to minor modifications in the model's output, and weight " $\beta$ " indicates the robustness of the variations.

Figure 7 demonstrates that variations in parameter " $s$ " has no impact on the value of the NEW1 model, reaffirming its stability. Changes in parameters " $w$ " and " $\beta$ " do not result in noticeable modifications to the model's output. However, the slight parameter variations in " $\alpha$ " lead to slight model value fluctuations. Overall, the sensitivity analyses highlight the significance of the predicted number of initial defects (parameter " $a$ ") in driving the behavior of the NEW1 model. Parameters " $w$ ", " $s$ ", and " $\beta$ " are considered stable and robust, while parameter " $\alpha$ " exhibit relatively minor effects on the model's output.

Figure 8 illustrates the sensitivity analysis results of the NEW1 model using DS3. It can be observed that the parameter " $w$ " exhibits stability, meaning that variations in its value have a minimal impact on the model's overall value. On the other hand, changes in the parameters " $s$ ", " $\alpha$ " and " $\beta$ " lead to minor fluctuations in the model's value.

Similar sensitivity analyses can be performed for other models using a similar approach.

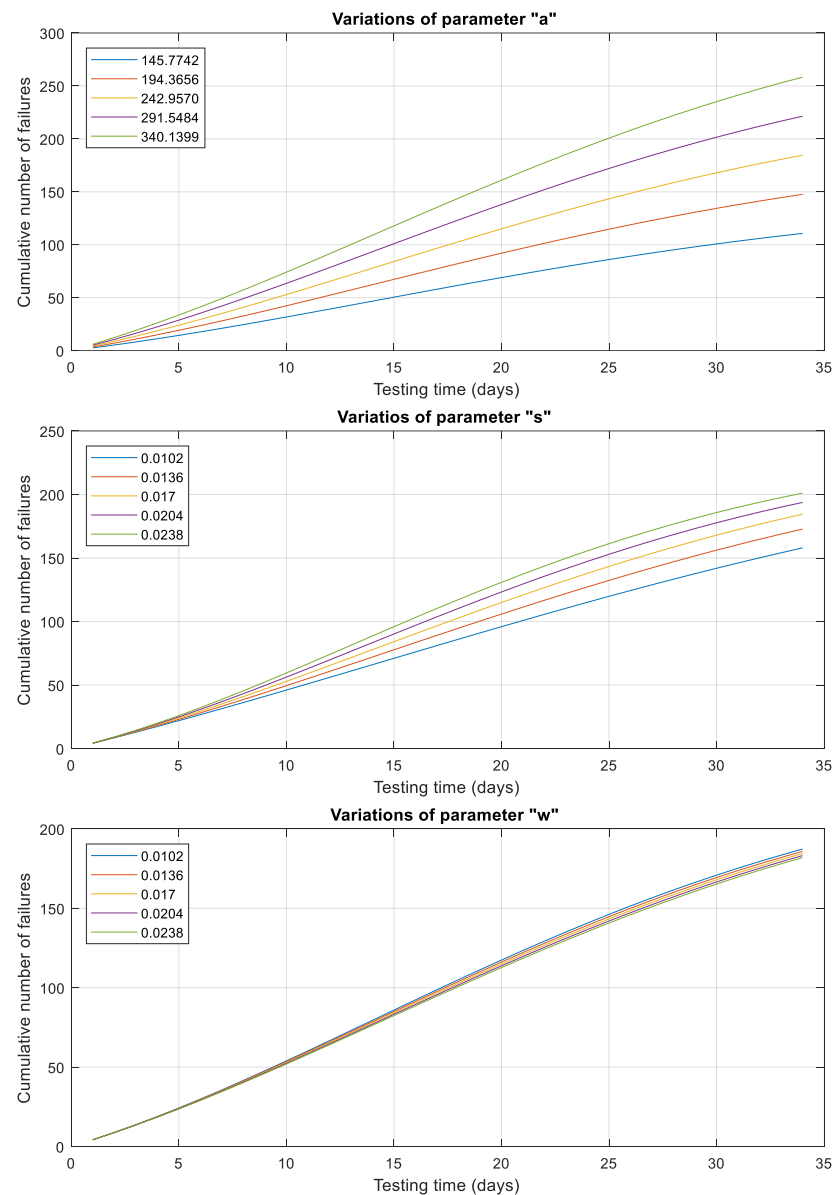
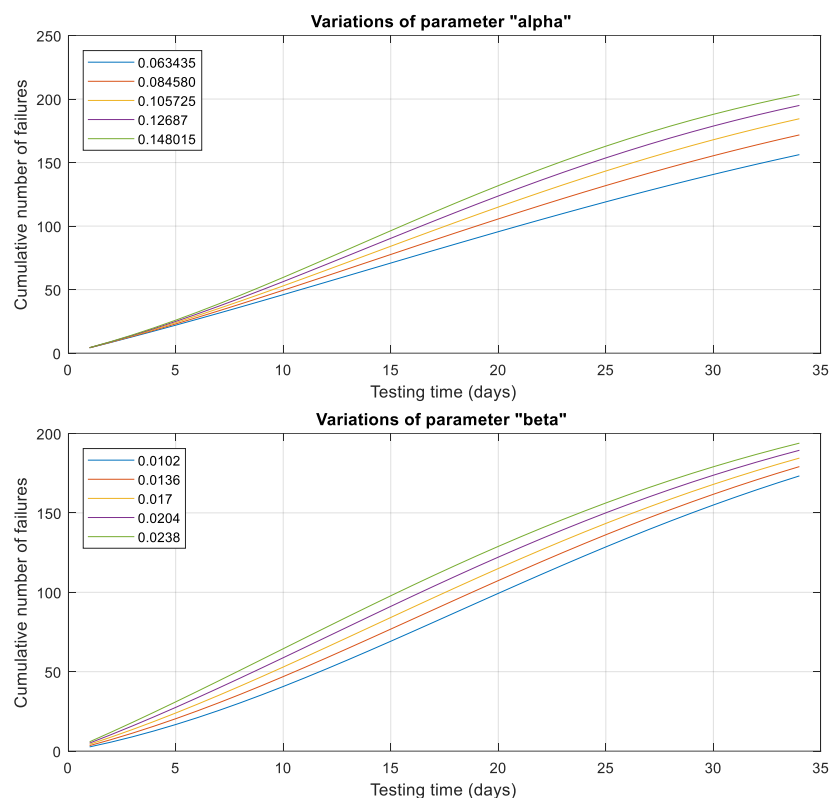


Figure 8. Cont.



**Figure 8.** Sensitivity analysis plots for the parameters of the NEW1 model using DS3.

## 6. Conclusions

In this study, we aimed to develop a novel software reliability model that integrates two critical human-related factors: learning and fatigue of software debuggers. While existing research has examined the impact of learning and experience on software reliability, there is a noticeable gap in the literature concerning the study of other human-related factors, such as fatigue. This work considered fatigue's effects on error making, incorporating fatigue as a crucial factor in constructing the software reliability model. The findings presented in this paper demonstrate the robust performance of the model across all the datasets examined, showcasing its efficacy in predicting software reliability. By employing the tanh function to represent learning and the exponential decay function to model fatigue, we have contributed to the existing knowledge in this field. The successful application of these functions to represent the FDR highlights their suitability for capturing the dynamics of human-related factors in the reliability estimation process. Despite the promising results, it is essential to acknowledge the limitations and constraints of our study. The unavailability of new datasets restricted our ability to test the model on more recent datasets. However, the older datasets are still relevant and valid in understanding the underlying principles in the current studied domain, as researchers widely use them.

Additionally, the choice of the FDR function was constrained to ensure the solvability of the resulting differential equation. For future research, we recommend exploring the development of alternative models that incorporate other factors affecting fault introduction. By considering a more comprehensive set of variables, we can further enhance the accuracy and applicability of software reliability models.

**Author Contributions:** Conceptualization, T.Y. and M.-F.L.; software, T.Y. and M.-F.L.; formal analysis, T.Y. and M.-F.L.; data curation, T.Y. and M.-F.L.; writing—original draft preparation, T.Y. and M.-F.L.; writing—review and editing, T.Y. and M.-F.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data availability is not applicable to this article as no new data were created or analyzed in this study.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Pham, H. *System Software Reliability. Reliability Engineering Series*; Springer: London, UK, 2006.
2. Yamada, S. *Software Reliability Modeling: Fundamentals and Applications*; Springer: Tokyo, Japan, 2014; Volume 5.
3. Baghdadi, G.; Jafari, S.; Sprott, J.C.; Towhidkhah, F.; Golpayegani, M.H. A chaotic model of sustaining attention problems in attention deficit disorder. *Commun. Nonlinear Sci. Numer. Simul.* **2015**, *20*, 174–185. [\[CrossRef\]](#)
4. Zhang, X.; Pham, H. An analysis of factors affecting software reliability. *J. Syst. Softw.* **2000**, *50*, 43–56. [\[CrossRef\]](#)
5. Pham, H.; Nordmann, L. A generalized NHPP software reliability model. In Proceedings of the 3rd Int'l Conference on Reliability and Quality in Design, Anaheim, CA, USA, 12–14 March 1997; pp. 116–120.
6. Goel, A.L.; Okumoto, K. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Trans. Reliab.* **1979**, *28*, 206–211. [\[CrossRef\]](#)
7. Yamada, S.; Ohba, M.; Osaki, S. S-shaped reliability growth modeling for software error detection. *IEEE Trans. Reliab.* **1983**, *32*, 475–484. [\[CrossRef\]](#)
8. Ohba, M. Inflection S-Shaped Software Reliability Growth Model. In *Stochastic Models in Reliability Theory. Lecture Notes in Economics and Mathematical Systems*; Osaki, S., Hatoyama, Y., Eds.; Springer: Berlin/Heidelberg, Germany, 1984; Volume 235.
9. Yamada, S.; Osaki, S. Software reliability growth modeling: Models and applications. *IEEE Trans. Softw. Eng.* **1985**, *12*, 1431–1437. [\[CrossRef\]](#)
10. Song, K.Y.; Chang, I.H.; Pham, H. A testing coverage model based on NHPP software reliability considering the software operating environment and the sensitivity analysis. *Mathematics* **2019**, *7*, 450. [\[CrossRef\]](#)
11. Yamada, S.; Tokuno, K.; Osaki, S. Imperfect debugging models with fault introduction rate for software reliability assessment. *Int. J. Syst. Sci.* **1992**, *23*, 2241–2252. [\[CrossRef\]](#)
12. Pham, H.; Zhang, X. An NHPP software reliability model and its comparison. *Int. J. Reliab. Qual. Saf. Eng.* **1997**, *4*, 269–282. [\[CrossRef\]](#)
13. Pham, H.; Nordmann, L.; Zhang, Z. A general imperfect-software-debugging model with S-shaped fault-detection rate. *IEEE Trans. Reliab.* **1999**, *48*, 169–175. [\[CrossRef\]](#)
14. Li, Q.; Pham, H. A testing-coverage software reliability model considering fault removal efficiency and error generation. *PLoS ONE* **2017**, *12*, e0181524. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Chiu, K.C.; Huang, Y.S.; Lee, T.Z. A study of software reliability growth from the perspective of learning effects. *Reliab. Eng. Syst. Saf.* **2008**, *93*, 1410–1421. [\[CrossRef\]](#)
16. Iqbal, J.; Ahmad, N.; Quadri, S.M.K. A software reliability growth model with two types of learning. In Proceedings of the 2013 International Conference on Machine Intelligence and Research Advancement, Katra, India, 21–23 December 2013; pp. 498–503.
17. Wang, J.; Wu, Z.; Shu, Y.; Zhang, Z. An imperfect software debugging model considering log-logistic distribution fault content function. *J. Syst. Softw.* **2015**, *100*, 167–181. [\[CrossRef\]](#)
18. Wang, J.; Wu, Z. Study of the nonlinear imperfect software debugging model. *Reliab. Eng. Syst. Saf.* **2016**, *153*, 180–192. [\[CrossRef\]](#)
19. Al-Turk, L.I.; Al-Mutairi, N.N. Enhancing reliability predictions by considering learning effects based on one-parameter Lindley distribution. In Proceedings of the 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, Gold Coast, Australia, 16–18 December 2020; pp. 1–7.
20. Huang, Y.S.; Chiu, K.C.; Chen, W.M. A software reliability growth model for imperfect debugging. *J. Syst. Softw.* **2022**, *188*, 111267. [\[CrossRef\]](#)
21. Verma, V.; Anand, S.; Kapur, P.K.; Aggarwal, A.G. Unified framework to assess software reliability and determine optimal release time in the presence of fault reduction factor, error generation and fault removal efficiency. *Int. J. Syst. Assur. Eng. Manag.* **2022**, *13*, 2429–2441. [\[CrossRef\]](#)
22. Luo, H.; Xu, L.; He, L.; Jiang, L.; Long, T. A Novel Software Reliability Growth Model Based on Generalized Imperfect Debugging NHPP Framework. *IEEE Access* **2023**, *11*, 71573–71593. [\[CrossRef\]](#)
23. Wood, A. *Software Reliability Growth Models*; TANDEM Technical Report; Tandem Computers: Cupertino, CA, USA, 1996; Volume 96.
24. Jeske, D.R.; Zhang, X.; Pham, L. Adjusting software failure rates that are estimated from test data. *IEEE Trans. Reliab.* **2005**, *54*, 107–114.
25. Pham, H.; Deng, C. Predictive-ratio risk criterion for selecting software reliability models. In Proceedings of the 9th International Conference on Reliability and Quality in Design, Honolulu, HI, USA, 7–9 August 2003; pp. 17–21.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.