Journal of Statistical Research *https://doi.org/10.3329/jsr.v59i2.88070*
2025, Vol. 59, No. 2, pp. 279-303 *ISSN 0256 - 422 X*

MODIFIED MAXIMUM LIKELIHOOD ESTIMATION THROUGH ARTIFICIAL NEURAL NETWORKS: A CASE STUDY ON ESTIMATING THE RATE OF RAYLEIGH PROCESS' OCCURRENCE

ZEDAN Z. MASHIKHIN, HALKAWT R. HUSSEIN, AND ADEL S. HUSSAIN
IT Department, Amedi Technical Institutes, University of Duhok Polytechnic, Duhok, Iraq

ABDULGHAFOR M. HASHIM
Catholic University in Erbil, Erbi, Iraq

EMAD A. AZ-ZO'BI
Department of Mathematics and Statistics, Mutah University, Mutah P.O.Box 7, Karak 61710, Jordan

MOHAMMAD A. TASHTOUSH*
Department of Basic Sciences, Al-Huson University College, Al-Balqa Applied University, Salt 19117, Jordan.
Faculty of Education and Arts, Sohar University, Sohar 311, Oman.
Email: tashtoushzz@su.edu.om

SUMMARY

The research develops a new Software Reliability Growth Model that utilizes the Non-Homogeneous Poisson Process (NHPP) with a Rayleigh process mean function. A final stage applies Modified Maximum Likelihood Estimator (MMLE) together with Artificial Neural Network (ANN) for better parameter value estimation to achieve improved estimation accuracy and convergence rates. The proposed model achieves a comprehensive evaluation by comparing its results against standard Inverse Rayleigh and Exponential and Half-Logistic, and Gamma distribution models. Testing occurred through the utilization of Mean Squared Error (MSE) and Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), and log-likelihood ($\log L$). The Rayleigh-based NHPP model exhibits superior performance against competing models because it produces lower MSE results, which reflects its strong ability for software reliability assessments. The research presents a study that evaluates MMLE methods versus Bayesian methods for estimation. MMLE produces lower MSE outcomes than Bayesian estimation for every dataset in the four experimental conditions, as shown in Table 8. The MSE from MMLE amounts to 0.0200, while Data Set 1 exhibits superior performance compared to the Bayesian method's values of 0.1120. New evidence demonstrates that MMLE gives superior performance to Bayesian estimation when estimating parameters of software failure data in Data Sets 2 through 4. The parameter uncertainty of Bayesian estimation produces MSE values higher than the values obtained from MMLE, although it incorporates prior beliefs. The research demonstrates how MMLE works well for practical reliability engineering applications, although Bayesian Variational Inference and Markov Chain Monte Carlo (MCMC) should be used for future efficiency improvements.

Keywords and phrases: Rayleigh Process, Artificial Neural Network Algorithm, Modified Maximum Likelihood Estimator, Mean Square Error, Software Reliability Growth Model (SRGM)

AMS Classification: 60J80, 60J85, 60K10

* Corresponding author

1 Introduction

It is common knowledge that computers are utilized for a wide range of purposes. Software is becoming more and more important; thus, it is absolutely necessary to have trustworthy software (Musa, 1987). Software does not inherently malfunction until its flaws cause it to do so. In general, handling software errors is more challenging. From the moment the program is installed on the computer, every design flaw is evident. Until a software breakdown occurs, a program's inherent flaw in software is not hazardous. In light of this, the idea of software dependability is rather reliant on software failure and how frequently it occurs than on the unknowable number of latent flaws in the software (Wood, 1996; Okumoto, 1979). As a result, rather than the flaws present in it, software reliability may be defined as the likelihood of failure-free operation of software. We cannot, however, overlook the reality that the quantity of errors also affects software dependability. In this sense, statistical analysis and the theory of probability have grown to be crucial to the creation of a model that assesses the dependability of software systems in real data. Software reliability growth models are studied in an attempt to quantify software quality in terms of dependability (Huang et al., 2022; Rao, 2013).

Software reliability models are statistical models that, given the system's failure history, may be used to forecast the failure rate of a software system. The models assume a defect detection and elimination procedure. The model's structure and the interpretation of its parameters are determined by these presumptions. Some recent works in this regard are by Colak (2024), Hussain et al. (2025b), Lee (2023), Groeneboom et al. (2010), Shafiq (2022) and Sindhu (2023) are a few recent studies in this area. In light of this, we investigate the modeling of software dependability using a mean value function derived from the Rayleigh process and NHPP. Similar attempts have been using the Pareto distribution (Hussain et al., 2025a; Luo et al., 2023; Hussein et al., 2025).

Developing from these basic models, researchers have conducted additions to the topic. For example, developed the model that introduced the use profile into the picture with the aim of increasing accuracy in reliability estimation. This model stresses need to know the operation situation in which the software works, and there is a need for a model that can change depending on this environment (Kim and Lee, 2022; Nguyen, 2022; Kim and Lee, 2025; Chupradit et al., 2022).

Furthermore, ANNs have also been embraced in field studies since they provide many tools for estimating parameters in complicated models. The use of ANNs in software reliability modelling has been studied by different researchers, one of whom is who proved that ANNs could capture the non-linear part of failure data to provide higher reliability and better accuracy of the model parameters (Huang et al., 2022; De Oliveira et al., 2012; Ravikumar et al., 2024; Yue et al., 2024).

Here there are some possible limitations due to the following assumptions in the proposed model. An assumption that failure events are stochastically independent is made along with the assumption that the Rayleigh model accurately describes software reliability. These assumptions could be a source of error if failures are dependent and if the failure rate does not change location. Pertinently, the effectiveness of the proposed model is realized from the quality and the quantity of data used in the training of the ANN and may be biased (Wu et al., 2024; Hussein et al., 2025). In part, this could reduce the generality, future applicability, and reliability of the model across different types of software systems to different classes of software systems and situations (Rao, 2011). The motivation

for this study arises from the desire to explore and enhance the modeling of software dependability using a mean value function derived from the Rayleigh process for NHPP. The approach proposed in this study builds upon previous work, particularly the efforts made by (Wang et al., 2024). In their research, the Pareto distribution within NHPP framework was utilized to model software dependability. The novelty in this paper is to propose a new model based on a modified maximum likelihood technique with the help of an ANNthe to estimate the parameters for the Rayleigh process (Shafiq, 2023; Ning et al., 2024).

This paper is organized to include an introduction in Section One. Section Two presents the model's origins and evolution along with the essential information regarding NHPP. Section 3 presents the ANN. In Section 4, the MMLE of the parameters of the proposed SRGM, is introduced using ANN. Section 5 presents the simulation study. Section 6 contains numerical examples with real data. In Section 7, the suggested SRGM is then contrasted with alternative software reliability growth models and contains a comparison Study of SRGMs Using the NHPP Model and a summary. Section eight contains conclusions. With a foundational understanding of software reliability, we now establish the mathematical framework underlying the proposed model. The next section develops an NHPP-based approach to characterize software failure occurrences over time (Inoue, 2004).

2 Method and Estimation

2.1 SRGM as NHPP

Suppose that we are interested in observing the occurrences of a repeatable event over a period of time. The significance in this instance is the frequency of software malfunctions that transpire throughout a specified period for testing or operation. However, it's worth noting that while failures may not follow a predictable pattern individually, the aggregate behavior of failures over time can sometimes reveal underlying trends or patterns. Therefore, statistical methods like random counting procedures are valuable tools for understanding and managing failure processes in various systems and industries. Let the number of occurrences at random in the range $[0, t]$ be denoted by $N(t)$, where t is any non-negative real number. If a counting process exhibits steady independent failure, it is considered a Poisson process, in this study, the objective is to observe the recurrence of repeating events over time, specifically focusing on the frequency of software malfunctions occurring within a designated testing or operational period. As failures typically lack a predictable pattern, a common approach involves employing a random counting procedure to count these erratic occurrences. The number of failures observed within a certain time period can serve as a method to identify such a failure process, with the mean lambda being equal to (Ning et al., 2024)

$$p[N(t+s) - N(t) = y] = \frac{[\lambda s]^y e^{-\lambda s}}{y!}, y = 1, 2, 3 \ldots \tag{2.1}$$

According to this mathematical model, the variations in $N(t)$ comparing a period to another, for example, $[t, t_s]$ rely solely on the interval's duration and not the extremes t, t_s of the interval. λ is referred to as the intensity of failure. If we consider a Poisson process, whose mean relies on both

the length of the interval s and the beginning t, then an equation may be used to explain a Poisson process as (Shirawia et al., 2024)

$$p\left[N\left(t\right)=y\right]=\frac{\left[m\left(t\right)\right]^{y}e^{-m(t)}}{y!},y=1,2,3\ldots \tag{2.2}$$

NHPP designed to analyze the failure process of software can be conceptualized as a Poisson process with a mean value function derived from the cumulative distribution function (CDF) of a continuous positive valued random variable. Given the plethora of distributions available in statistical science, multiple NHPP models can be formulated, each based on a distinct CDF. Among these models, the foundational one is attributed to (Hussain et al., 2025c), which relies on the exponential distribution. Subsequently, numerous NHPP models have been proposed and investigated by various researchers, as documented in works such as (Huang et al., 2022; Hussain et al., 2025c; Kim and Lee, 2022; Nguyen, 2022; Chupradit et al., 2023; Abiodun et al., 2018; Hussain et al., 2025a) along with related references. Assume the following NNPP, which is described by a time-varying rate of occurrence

$$\lambda(t)=(1/b^{2})t,\ \ t\geq 0,b>0, \tag{2.3}$$

where b represents the scale parameter. The process parameter, $m_1(t)$, is the cumulative function of the temporal rate of occurrence and signifies the mean rate and is given by (Colak, 2024)

$$m_{1}(t)=\int_{0}^{t}\lambda\left(u\right)du=\int_{0}^{t}\frac{u}{b^{2}}du=\frac{t^{2}}{2b^{2}}. \tag{2.4}$$

For the stochastic process, the inter-arrival times are determined by (Chupradit et al., 2022)

$$f(t)=\lambda\left(t\right)\exp\{-\int_{0}^{t}\lambda(u)du\}. \tag{2.5}$$

For the Rayleigh process, we may express the probability density function as follows

$$f(t)=(t/b^{2})\exp(-t^{2}/(2b^{2})),\ t>0. \tag{2.6}$$

Our analysis involves NHPP with a mean value function that is stated as a function of the cumulative function of the time rate of occurrence, as follows

$$m(t)=at^{2}/(2b^{2}),\ \ a>0,t>0. \tag{2.7}$$

Equation (2.7) is known as NHPP, based on software reliability, or as RGM. It can be seen that $m(t)$ tends to an as t approaches infinity and that $m(t)$ is a positively valued, non-decreasing function of time t. The likelihood that there will be no failures in the time period $[0,t]$ is represented by the software system's reliability, which is modeled as previously mentioned. It may be written as follows

$$R(t)=p\{N(t)=0\}=e^{-m(t)}. \tag{2.8}$$

Reliability $R\left(\frac{x}{t}\right)$ generally indicates the likelihood that there won't be any failures during the period $[t, t+x]$ is given by

$$R\left(\frac{x}{t}\right) = p\left\{N\left(t+x\right) - N\left(t\right) = 0\right\} = e^{-[m(t+x)-m(t)]}. \tag{2.9}$$

The formula found in Equation (2.9) is commonly referred to as the SRGM or software reliability based on an NHPP. We may obtain the software dependability value at any moment by providing the mean value function with all of its parameters. In order to evaluate the software quality, an estimate of the software reliability must be obtained, and if the parameters of the mean value function are unknown, they must be calculated using software failure data in the form of failure counts. In Section 4, we introduce the MMLE estimate of parameters in an NHPP based on the Rayleigh process (Huang et al., 2022).

A major roadblock exists for precise parameter estimation in traditional NHPP-based models that use a structured approach for modeling software failures. This problem requires ANN as a computational tool that enhances parameter estimation capabilities. The subsequent segment introduces ANN concepts while explaining their role in our approach.

2.2 ANN

ANN are very simple electrical models that draw inspiration from the architecture of the brain. Because the brain is an experience-based learning machine, it can naturally solve some issues that are now beyond the capabilities of traditional computers. One such example is neural networks. Energy economical packages (Luo et al., 2023). (A neuron may constitute the fundamental building block of an ANN. A neuron is AN information-processing unit that's essential to the operation of a neural network. Figure 1 shows the model of a neuron, which forms the idea for ANN's. The substitute neurons utilized in constructing our neural networks are considerably primitive in comparison to those inherent in the brain. A synthetic neuron typically features multiple inputs but only one output. Here, we identify three fundamental components of the neuron model) (Luo et al., 2023):

I. The connecting points, or synapses, each have a unique weight and strength. More precisely, x_j at the input of conjugation j connected to a neuron is increased by the colligation weight w_j.

II. Associate degree activation operates to limit the amplitude of the output of a neuron.

III. The model of a neuron. Additionally includes associate degree external bias, denoted by b, which has the impact of skyrocketing or lowering internet input of the activation.

From a mathematical perspective, a neuron is represented by:

$$y = \varphi\Big(\sum_{j=1}^{N} w_j x_j + b\Big), \tag{2.10}$$

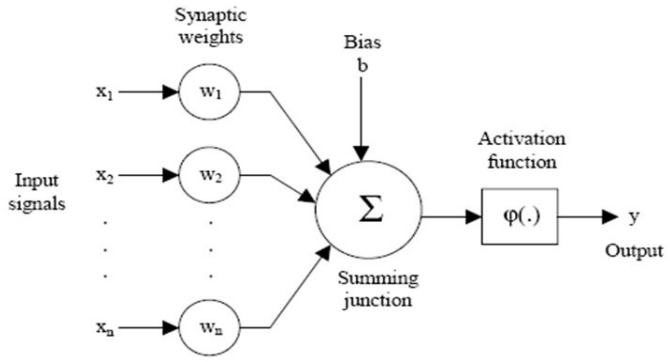


Figure 1: Model of Nonlinear Neurons

where $x_1, \ldots, x_n$ are the input signals, $w_1, \ldots, w_n$ stands for the neuron's synaptic weights, and b for the bias, $\phi(.)$ is the activation function and y are the output signal of the neuron.

The next part focuses on introducing the MMLE method after deepening our knowledge of ANN usage in parameter assessment. The method uses ANN learning ability to improve parameter estimates, which enhances software reliability assessment accuracy.

3 Method of Estimation

3.1 MMLE

Assume that some software failure data is expressed as (y_i, t_i), where i is a number between 1 and n. Here, y_i denotes the number of failures that have been seen within the time interval $[0, t_i]$, where time points fall into the range $0 < t_1 < t_2 < \ldots < t_n$. Usually, this kind of information is called failure count data. The NHPP parameters may be estimated by building a log-likelihood function, which has the following (Hussain et al., 2025a)

$$LLF = \sum_{i=1}^{n} (y_i - y_{i-1}) \log \left[m\left(t_i\right) - m\left(t_{i-1}\right) \right] - m(t_n). \tag{3.1}$$

Compute $m(t_i) - m(t_{i-1})$;

$$m\left(t_i\right) - m\left(t_{i-1}\right) = \frac{\mathrm{a}}{2\mathrm{b}^2} \left(t_i^2 - t_{i-1}^2\right). \tag{3.2}$$

Rewriting LLF:

$$LLF = \sum_{i=1}^{n} (y_i - y_{i-1}) \log \left[\frac{\mathrm{a}}{2\mathrm{b}^2} \left(t_i^2 - t_{i-1}^2\right) \right] - \frac{\mathrm{a}}{2\mathrm{b}^2} t_n^2, \tag{3.3}$$

using logarithm properties, then;

$$LLF = \sum_{i=1}^{n} (y_i - y_{i-1}) \left[\log a + \log \left(\frac{\mathrm{a}}{2\mathrm{b}^2} \left(t_i^2 - t_{i-1}^2 \right) \right) \right] - \frac{\mathrm{a}}{2\mathrm{b}^2} t_n^2. \tag{3.4}$$

Set $\frac{\partial LLF}{\partial a} = 0$, then;

$$\frac{\partial LLF}{\partial a} = \sum_{i=1}^{n} \left(\tfrac{1}{a} \right) (y_i - y_{i-1}) - \tfrac{1}{2b^2} t_n^2 = 0, \tag{3.5}$$

solving for a, yields;

$$a = y_n \{ 1 - e^{-\frac{\mathrm{t}_n{}^2}{2\mathrm{b}^2}} \}^{-1}. \tag{3.6}$$

The following formula is used to obtain the derivative of the logarithm of the probability function with respect to the parameter b: $\frac{\partial LLF}{\partial b} = 0$, we get:

$$\sum_{i=1}^{n} (y_i - y_{i-1}) \frac{\frac{t_{i-1}^2}{b^3} e^{-\frac{t_{i-1}^2}{2b^2}} - \frac{t_i^2}{b^3} e^{-\frac{t_i^2}{2b^2}}}{e^{-\frac{t_{i-1}^2}{2b^2}} - e^{-\frac{t_i^2}{2b^2}}} - \frac{y_n (t_n)^2}{b^3 \left(e^{-\frac{t_n^2}{2b^2}} - 1 \right)} = 0, \tag{3.7}$$

MLE estimates for the parameters a and b may be obtained by simultaneously solving Equations (3.6) and (3.7). It is important to note that these equations may be worked out repeatedly. Therefore, we introduce a MMLE method that incorporates one of the most influential artificial intelligence techniques ANN by using the algorithm in Appendix.

3.2 Confidence Intervals

We are assuming that we are working with a set of observed data $t_1, t_2, \ldots, t_n$ that follow the distribution defined by Equation (2.6). To compute confidence intervals for b, we will estimate b using the MLE method, and then compute the confidence intervals using its distribution.

$$L(b) = \prod_{i=1}^{n} \frac{1}{b^2} t_i e^{-\frac{1}{2b^2} t_i^2}, \tag{3.8}$$

By taking the logarithm of the likelihood function, then;

$$l(b) = \log L(b) = -n \log b^2 + \sum_{i=1}^{n} \log t_i - \frac{1}{2b^2} \sum_{i=1}^{n} t_i^2. \tag{3.9}$$

From Equation (3.9) we can find the MLE b as it comes;

$$\hat{b}_{MLS} = \sqrt{\sum_{i=1}^{n} t_i^2 / 2n}, \tag{3.10}$$

The MLE estimator for b is asymptotically normal, and its variance can be estimated using the Fisher information. The Fisher information is the negative expected value of the second derivative of the log-likelihood function. We calculate:

$$\frac{d^2\ell\left(b\right)}{db^2} = \frac{2n}{b^2} - \frac{3}{b^4}\sum_{i=1}^{n} t_i^2. \tag{3.11}$$

The Fisher information $I(b)$ is the negative expectation of this second derivative. After simplifying, we obtain

$$I\left(b\right) = n/b^4. \tag{3.12}$$

Therefore, the asymptotic variance of $\hat{b}_{MLS}$ is:

$$Var\left(\hat{b}_{MLS}\right) = b^4/n. \tag{3.13}$$

The asymptotic distribution of $\hat{b}_{MLS}$ is approximately normal with mean b and variance $\frac{b^4}{n}$. Thus, for large n, the 0.95 confidence interval for b is

$$\left[\hat{b}_{MLS} - 1.96\sqrt{\hat{b}_{MLS}^4/n}, \hat{b}_{MLS} + 1.96\sqrt{\hat{b}_{MLS}^4/n}\right]. \tag{3.14}$$

3.3 Propose Networks

A neural network with a single hidden layer is a common and foundational architecture in artificial neural networks. Its ability to learn complex patterns in data through the adjustment of weights and biases during training has made it widely applicable in various fields. The lack of consensus regarding the optimal number of nodes and hidden layers in neural network architectures is indeed a well-known challenge in the field of deep learning. This variability arises due to the fact that the ideal configuration often depends on the specific characteristics and complexity of the problem being addressed.

In this study, a multilayer Feed Forward Neural Network (FFNN) that contains two hidden layers is chosen: one input and one output. The first hidden layer contains six nodes, and the second hidden layer contains nine nodes. After several experimental iterations, it was found that the neural network architecture with two hidden layers outperformed those with one or three hidden layers. Specifically, when utilizing six nodes in the first hidden layer and nine nodes in the second hidden layer, this configuration yielded statistically superior results. To estimate the parameters of the Rayleigh process. The proposed FFNN divides the inputs for 0.70 for training and 0.30 for testing. The error to be computed is given by

$$MSE = \sum\nolimits_{i=1}^{n} \left(y_i - m\left(t_i\right)\right)^2/(n-N). \tag{3.15}$$

3.3.1 Estimation Using ANN

This algorithm describes the initialization and training process of ANN model; it involves preparing the input data for the neural network. Each input variable represents a feature or characteristic of

the data. These inputs are fed into the neurons in the input layer of the neural network. Before training begins, the weights and biases of the neural network need to be initialized. This is typically done randomly, often from a uniform distribution. The weights represent the strength of connections between neurons in adjacent layers, while biases provide each neuron with an additional parameter to adjust the output (Kim and Lee, 2025).

Once the weights and biases are initialized, the input data is fed forward through the network. Each neuron in the hidden layer receives input from the input layer, applies a weighted sum along with a bias term, and then applies an activation function to produce an output. We chose a multilayer FFNNnodes that contains two hidden layers: one input and one output. The first hidden layer contains six nods, and the second hidden layer contains nine nodes. After several experimental iterations, we found that the neural network architecture with two hidden layers outperformed those with one or three hidden layers. Specifically, when utilizing six nodes in the first hidden layer and nine nodes in the second hidden layer, this configuration yielded statistically superior results. The output of the neural network is compared to the actual target values using a loss function. In this case, the MSE is commonly used. Lower MSE indicates better performance of the neural network in approximating the target values. This process continues until the model achieves satisfactory performance or until a stopping criterion is met. These processes are summarized in Algorithm 1 (Hussain, 2022).

3.4 Convergence Properties of the MMLE Method

MMLE method achieves convergence based on several elements, which include starting parameter assumptions, together with the likelihood function organization, alongside optimization process algorithms. The subsequent segment details convergence properties as follows.

3.4.1 Likelihood Function and Optimization

The likelihood function for the NHPP is derived from the Poisson process and is based on the Rayleigh distribution. The likelihood function is non-linear in the parameters a and b, which makes the optimization problem non-trivial.

When applying the MMLE methodology users need to solve first-order conditions derived from the log-likelihood function derivatives to obtain parameter value estimates. To obtain results, the method depends on iterative approaches including Newton-Raphson and descent gradient because these algorithms demonstrate convergence under specific requirements.,

3.4.2 Role of ANN in Convergence

The paper introduces an ANN to assist in the estimation process. The ANN is used to approximate the relationship between the input data (failure counts and time intervals) and the parameters a and b.

A non-linear approximation capability of the ANN assists both in improving parameter-initial guess values and in deriving parameter estimates. The MMLE method uses the ANN to find better

initial parameter guesses that decrease the required number of iterations for reaching the optimal solution.

The ANN is trained using a feed-forward architecture with two hidden layers, and the MSE is used as the loss function. The training process involves gradient-based optimization, which is known to converge to a local minimum under appropriate conditions (e.g., proper learning rate, sufficient iterations).

3.4.3 ANN

MMLE method usually determines its convergence state by checking changes in both log-likelihood function values and parameter estimates throughout successive iterations. The algorithm finishes when the changes in parameter values sink below a pre-established threshold value ϵ.

Implicit convergence evaluation in the paper uses model MSE comparisons between the proposed structure and alternative models (Inverse Rayleigh, Exponential, Half Logistic, and Gamma). The results show that the proposed model using MMLE with ANN methods attains better solutions than traditional methods, judging from its lower MSE values.

3.4.4 Empirical Evidence of Convergence

The paper presents both simulation research and real data tests that verify the MMLE method converges. Tests indicate that the Rayleigh Process with MMLE and ANN model demonstrates superior performance compared to other models regarding MSE, AIC, and BIC metrics.

3.4.5 Summary of Convergence Properties

The method proposed in the paper shows excellent convergence properties after utilizing ANN. Through ANN the estimation process becomes more efficient because it produces precise starting values and creates an approximation for the nonlinear association between parameters and input data.

There is empirical evidence that demonstrates the proposed model produces accurate parameter estimation through MSE measurements that surpass those of competing models.

3.5 Model Validation and Performance

To assess the reliability of the MMLE model, we applied k-fold cross-validation and bootstrap validation to estimate the stability of the parameters and the model's generalizability.

3.5.1 k-Fold Cross-Validation

With $k = 10$ folds, where the dataset was divided into 10 equal subsets. The model was trained on 9 subsets and tested on the remaining one, iteratively. MSE was computed for each fold, and the final error estimate was obtained by averaging the errors across all folds. The final cross-validation result was $MSE = 206,259.95$.

3.5.2 Bootstrap Validation

We performed 1000 bootstrap iterations, where random samples (with replacement) were drawn from the dataset. Each sample was used to train the MMLE model, and predictions were made on the entire dataset. The MSE was computed for each iteration, and a 0.95 confidence interval was estimated. The final bootstrap estimate was MSE = 186,570.56, with a 0.95 confidence interval of (165,833.34 - 247,785.28).

3.5.3 Discussion Result

The validation results indicate that the MMLE model's MSE is 206,259.95 (cross-validation) and 186,570.56 (bootstrap), with a confidence interval of 165,833.34 - 247,785.28. While this suggests reasonable performance, further refinements such as feature transformations and hyperparameter tuning may improve the accuracy. Future research could explore increasing the dataset size or testing alternative estimation techniques to validate the robustness of MMLE.

3.6 Bayesian Estimation

The estimation of stochastic processes parameters through Bayesian estimation can be considered the most efficient method due to its strong framework that integrates prior information. The method uses a previously specified parameter distribution to include existing information about the parameters while incorporating observed data through the likelihood function. Performing integration between the probability of observed data under parameter conditions expressed by the likelihood function and Bayes' theorem leads to obtaining the posterior distribution of the parameters. After seeing the data, the posterior distribution represents the new and improved parameter knowledge (Hussain, 2022). The application works under an inverse gamma distribution assumption for the parameter because of its convenient mathematical properties alongside its conjugacy benefits with specific likelihood functions (De Oliveira et al., 2012). When applying the likelihood function in Equation (3.8), the inverse gamma prior serves as an analytical framework that provides flexibility during Bayesian inference processes.

The Prior Distribution, since b represents a scale parameter, a common prior is the Jeffreys prior, which is invariant under reparameterization

$$p\left(b\right) \propto \frac{1}{b}. \tag{3.16}$$

Alternatively, we can assume a conjugate prior, such as an inverse gamma

$$p(b) = \frac{\beta^{\alpha}}{\Gamma\left(\alpha\right)} b^{-\alpha-1} e^{-\beta/b}, \tag{3.17}$$

where $\alpha, \beta > 0$ are hyper-parameters. Then the posterior distribution function is

$$p\left(b|t_1, t_2, \ldots, t_n\right) \propto p\left(b\right) L\left(b\right), \tag{3.18}$$

For the Jeffreys prior for Equation (3.15), the posterior simplifies to

$$p\left(b|t_1,t_2,\ldots,t_n\right) \propto \frac{1}{b^{2n+1}} e^{-\sum \frac{1}{2b^2}\ t_i{}^2}. \tag{3.19}$$

Using a change of variables $x = \frac{1}{b^2}$, this corresponds to an inverse gamma distribution

$$b^2|t_1,t_2\ldots,t_n \ \sim \ \text{Inverse} - \text{Gamma}\Big(n, \frac{1}{2}\sum_{i=1}^{n} t_i^2\Big). \tag{3.20}$$

Thus, the Bayesian estimate of b (posterior mean) is

$$\hat{b}_{Bayes} = \sqrt{\sum_{i=1}^{n} t_i^2/2n}. \tag{3.21}$$

4 Numerical Example

In this section, we fit the proposed model and existing models with actual data to estimate the criteria and compare their goodness of fit. Firstly, we fit the data set to each model (Mean Value Function) and estimate the parameters of each model using the modified maximum likelihood method. Then, we calculate the criteria using MSE and compare the goodness of fit.

The data set practiced in this study was collected from a real-time command and control system developed by Bell Laboratories (Kim and Lee, 2022); the data is used to compare the goodness of fit of the different models. The results are shown in Table 1.

Secondly, the four datasets are graphically examined to investigate their suitability for the Rayleigh process in SRGM. This is achieved by plotting the distribution using an artificial neural network, as shown in Figure 2.

A logarithmic scale was used in Figure 2 to enhance visualization of the cumulative number of failures across datasets. Since failure counts vary significantly over time, a logarithmic transformation helps in highlighting patterns that may not be clearly visible on a linear scale, especially for early-stage failures where values are small. This approach improves interpretability by emphasizing proportional differences rather than absolute changes, making trends in software failure growth more apparent. The simulation analysis of our proposed ANN-based MMLE approach enables testing its performance metrics. The analysis verifies parameter estimation precision when measuring sample size while evaluating the efficacy of our method in contrast to conventional techniques (Ning et al., 2024).

5 Simulation Study

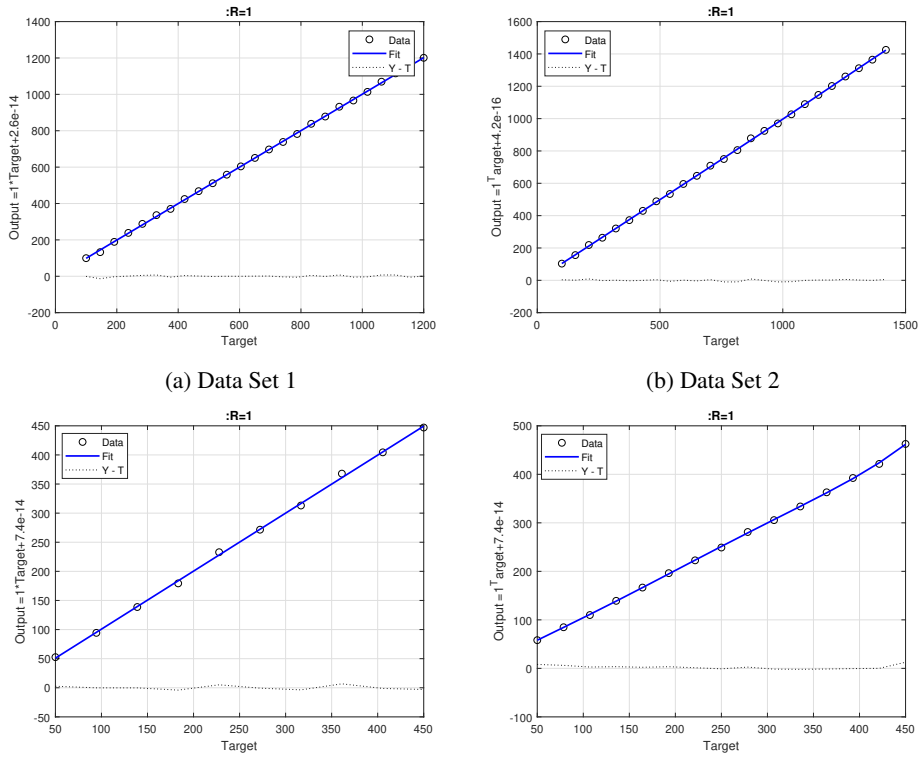
Simulation is a scenario designed to compare any system with the real world and is defined as the attempt to simulate a particular process under specific circumstances using artificial methods that resemble natural conditions. This includes building a smaller model that is an identical copy of the

Table 1: The real data error corresponds to the observed errors during the system tests.

Test Week	Data Set 1		Data Set 2		Data Set 3		Data Set 4	
	CPU hours	defects found	CPU hours	defects found	CPU hours	defects found	CPU hours	defects found
1	519	16	384	13	162	6	254	1
2	968	24	1186	18	499	9	788	3
3	1430	27	1471	26	715	13	1054	8
4	1893	33	2236	34	1137	20	1393	9
5	2490	41	2772	40	1799	28	2216	11
6	3058	49	2967	48	2438	40	2880	16
7	3625	54	3812	61	2818	48	3593	19
8	4422	58	4880	75	3574	54	4281	25
9	5218	69	6104	84	4234	57	5180	27
10	5823	75	6634	89	4680	59	6003	29
11	6539	81	7229	95	4955	60	7621	32
12	7083	86	8072	100	5053	61	8783	32
13	7487	90	8484	104			9604	36
14	7846	93	8847	110			10064	38
15	8205	96	9253	112			10560	39
16	8546	98	9712	114			11008	39
17	8923	99	10083	117			11237	41
18	9282	100	10174	118			11243	42
19	9461	100	10272	120			11305	42
20	10000	100						

real model performing tests on the miniature model and examining the results and generalizing them to the original model, or computer simulation by writing a program for the methods to be chosen under realistic programming conditions and then observing the results obtained with the program and drawing a conclusion based on them (De Oliveira et al., 2012).

There are different simulation methods, namely the (analogy method), the (mixed method) and the (Monte Carlo method). The Monte Carlo method is one of the most important and widely used simulation methods, in which a random sample of the phenomenon is generated, that corresponds to the behavior of a certain probability distribution that the phenomenon has. To achieve this, the probability distribution of the phenomenon it has (CDF) is known that the set of samples random



(a) Data Set 1

(b) Data Set 2

(c) Data Set 3

(d) Data Set 4

Figure 2: The scatter diagram for the logarithmic distribution of the cumulative number.

in this way possesses the property of independence because random samples in this method is by applying the mathematical method to each sample separately (Ravikumar et al., 2024).

To put the previously discussed ideas into practice, the practical part of the research focused on the estimators of the suggested model during the fuzzy phase for both of the approaches used, utilizing a simulation method. The objective was to apply the Mean Square Error (MSE) statistical criteria to various sample sizes in order to assess the optimality of these estimators. The purpose of the simulation model was to provide a comparison study of the approaches that were evaluated while accounting for a variety of real data situations. By showing how the estimate techniques affect the following variables, this strategy seeks to determine the best technique for estimating parameters inside the interval of the Rayleigh Process (Wu et al., 2024).

5.1 Stages of Building a Simulation Experience

First stage It is the most important stage on which the program's steps and procedures depend. Below are the steps for this stage:

1. Choose default values for the parameters of the Rayleigh Process. Several default values were chosen for the shape parameter a and the scale parameter b for the Rayleigh Process by reviewing previous studies and experimenting with many default values for the parameters, which led us to choose the best of these values, as follows: (a=0.9;1.2;1.1 and b=0.9;1.5;1.1)

2. Choose sample sizes. Several different sample sizes (small, medium, large) were chosen as follows: (n = 20; 60; 80; 100).

Second stage: Data generation: At this stage, random data is generated using the inverse transformation method and according to the Rayleigh Process, as follows:

1. Generating a random variable u_i that follows a uniform distribution with the interval $(0, 1)$ using the cumulative distribution function with the help of the Rand.

$$u_i \sim U(0,1), i = 0, 1, 2 \ldots, n, \tag{5.1}$$

   where u_i represents a continuous random variable that follows a uniform distribution.

   Convert the data generated in step (first) that follows a uniform distribution into data that follows a Rayleigh Process using the inverse function (CDF) transformation method and according to Equation (2.7) and as in the following formula.

$$t_i = \sqrt{2b^2 u/a}, \quad i = 0, 1, 2 \ldots, n. \tag{5.2}$$

Third stage: At this stage, parameters are estimated over the period for the Rayleigh Process SRGMs and for all methods, which are the FFNN.

Fourth stage: The experiment is repeated (1000) times.

Model simulations indicate that the proposed model outperforms the others based on the MSE criterion, as observed in the table above and illustrated in Figure 3.

Our proposed model exhibits robustness according to simulation results, which lead to a performance evaluation against current SRGMs. Our proposed ANN-enhanced approach demonstrates superior performance in estimation accuracy along with improved model fit according to this comparative assessment.

6 Comparison Study of SRGMs Using the NHPP Models

This paper presents a thorough comparison analysis of the existing NHPP foundation of the SRGM versus predetermined preference criteria. Among the standard models being examined are those that are based on: Inverse Rayleigh cumulative distribution function, Half logistic cumulative distribution function, Gamma cumulative distribution functions with shape parameter 2, Exponential cumulative distribution function.

The first NHPP is called Inverse Rayleigh (Wu et al., 2024). The second NHPP is called Goel -Okumoto Model (Wang et al., 2024). The third NHPP is called software reliability growth model

Table 2: The simulation table shows the MSE used to evaluate models across all sample sizes, based on the assumed sample sizes and the process parameter values of the five models.

Sample Size	Parameters		MSE				
	a	b	Proposed Model	Inverse Rayleigh	Exponential	Half Logistic	Gamma
20	1.1	0.9	0.1666	0.5143	1.4816	9.1129	8.1727
	0.9	1.1	0.4612	0.5815	1.8369	9.0403	9.6451
	1.2	1.5	0.1265	0.3536	2.4855	9.2952	8.8927
60	1.1	0.9	0.0961	0.1427	0.9662	5.4158	4.5670
	0.9	1.1	0.3123	0.3497	1.2278	5.4767	4.0557
	1.2	1.5	0.0250	0.2941	1.2845	5.2506	6.1790
80	1.1	0.9	0.0378	0.1394	0.8111	4.6346	2.4328
	0.9	1.1	0.2560	0.3115	0.8694	4.4992	3.7652
	1.2	1.5	0.0517	0.1539	1.1882	4.6807	3.6041
100	1.1	0.9	0.0649	0.0951	0.7735	4.2098	0.1559
	0.9	1.1	0.1079	0.1486	0.7690	4.1568	0.1484
	1.2	1.5	0.2049	0.2659	0.9831	4.0313	0.2298

based on half logistic model (Shafiq, 2023). The fourth NHPP is called Yamada S-shaped software reliability growth model (Chupradit et al., 2022). The mean value functions and related differentiation results are given in the next section for easy access. The following information is crucial to determining a MMLE of the parameters for the four competing models as well as our suggested model:

1. Rayleigh Distribution (The proposed model):

$$\sum_{i=1}^{n}(y_i - y_{i-1})\,\frac{\frac{t_{i-1}^2}{b^3}e^{-\frac{t_{i-1}^2}{2b^2}} - \frac{t_i^2}{b^3}e^{-\frac{t_i^2}{2b^2}}}{e^{-\frac{t_{i-1}^2}{2b^2}} - e^{-\frac{t_i^2}{2b^2}}} - \frac{y_n(t_n)^2}{b^3\left(e^{-\frac{t_n^2}{2b^2}} - 1\right)} = 0, \tag{6.1}$$

   where $a = y_n/\left(1 - e^{-\frac{t_n{}^2}{2b^2}}\right)$.

2. Inverse Rayleigh Distribution Model (Wu et al., 2024):

$$\sum_{i=1}^{n}(y_i - y_{i-1})\,\frac{(t_{i-1})^2 e^{-\frac{b}{(t_{i-1})^2}} - (t_i)^2 e^{-\frac{b}{(t_i)^2}}}{e^{-\frac{b}{(t_i)^2}} - e^{-\frac{b}{(t_{i-1})^2}}} + y_n(t_n)^2 = 0, \tag{6.2}$$

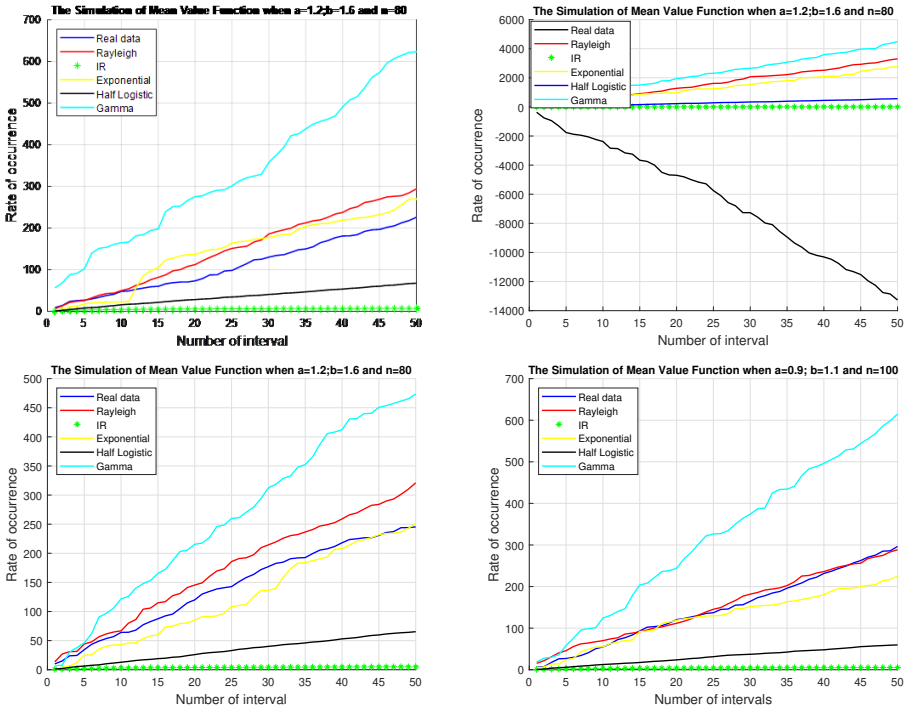


Figure 3: Plot the simulated mean value functions for the five distributions using different sample sizes and varying parameters.

where $a = y_n \times e^{\frac{b}{(t_n)^2}}$.

3. Exponential Distribution Model (Hussain et al., 2025a):

$$\sum_{i=1}^{n} (y_i - y_{i-1}) \frac{t_i \left(e^{-bt_i}\right) - t_{i-1} \left(e^{-bt_{i-1}}\right)}{\left(e^{-bt_{i-1}}\right) - \left(e^{-bt_i}\right)} - \frac{y_n t_n e^{-bt_n}}{1 - e^{-bt_n}} = 0, \tag{6.3}$$

where $a = \frac{y_n}{1-e^{-bt_n}}$.

4. Half Logistic Distribution Model (Wang et al., 2024):

$$\sum_{i=1}^{n} (y_i - y_{i-1}) \frac{\frac{t_i e^{-bt_i}}{\left(1+e^{-bt_i}\right)^2} - \frac{t_{i-1} e^{-bt_{i-1}}}{\left(1+e^{-bt_{i-1}}\right)^2}}{\hat{a}\left[\frac{-2e^{-bt_i}}{(1+bt_{i-1})\left(1-e^{-bt_{i-1}}\right)}\right]} - \frac{t_n e^{-bt_n}}{\left(1+e^{-bt_n}\right)^2} = 0, \tag{6.4}$$

where $a = y_n \times \frac{1+e^{-bt_n}}{1-e^{-bt_n}}$.

5. Gamma Distribution Model (Shafiq, 2023):

$$\sum_{i=1}^{n}(y_i - y_{i-1})\,\frac{t_i^2\left(e^{-bt_i}\right) - t_{i-1}^2\left(e^{-bt_{i-1}}\right)}{\hat{a}\left[(1+bt_{i-1})\left(e^{-bt_{i-1}}\right) - (1-bt_i)\left(e^{-bt_i}\right)\right]} - at_n^2 e^{-bt_n} = 0, \qquad (6.5)$$

where $a = y_n/[1-(1+bt_n)\exp(-bt_n)]$.

To conduct a thorough comparison of the models, we utilize the quantitative measure known as MSE. The process involves obtaining MMLE using FFNN for $m(t)$. In this procedure, we compute the MMLE via FFNN for the parameters and estimators of the mean value function in each of the four data sets (Kim and Lee, 2025). Subsequently, the MSE values are calculated for the various models under consideration. The condensed results are presented in Table 3 - Table 8.

Table 3: 95% Confidence Interval for the Rayleigh Process in the Proposed Model

Parameters	Data set 1	Data set 2	Data set 3	Data set 4
$\widehat{a}$	(0.0211, 0.0444)	(0.1753,0.2341)	(0.2653,0.3241)	(0.2753,0.3442)
$\widehat{b}$	(0.0414, 0.0646)	(0.2754,0.4344)	(0.3754,0.5344)	(0.4654,0.5314)

Table 4: Refined MMLE and Goodness-of-Fit (GOF) criteria for the 1st data

Distributions	Rayleigh Process (proposed model)	Inverse Rayleigh	Exponential	Half Logistic	Gamma
Parameters	$\widehat{a} = 999.00$	$\widehat{a} = 107.0339$	$\widehat{a} = 104.4582$	$\widehat{a} = 101.8768$	$\widehat{a} = 299.6177$
estimation	$\widehat{b} = 0.0530$	$\widehat{b} = 27.1805$	$\widehat{b} = 0.1577$	$\widehat{b} = 0.2339$	$\widehat{b} = 0.0595$
$-\log L$	100.1117	103.9409	104.1828	116.5324	114.7385
MSE	0.0200	102.4486	91.5324	71.3439	914.2080
AIC	206.2235	248.8873	231.4769	239.0537	212.2644
BIC	209.9978	250.0786	232.6240	241.7281	214.7706

6.1 Discussion

In summary, the results are presented in Table 3 - Table 8. demonstrate that the Rayleigh Process model exhibits strong performance across all datasets, as evidenced by its relatively low MSE, AIC, BIC, and $logL$ values. This suggests that the model provides a good fit with high predictive accuracy. However, the performance of alternative distribution models varies, with their suitability being contingent upon the specific characteristics of each dataset, such as skewness, tail behavior, and underlying stochastic properties.

Table 5: Refined MMLE and GOF criteria for the 2nd data

Distributions	Rayleigh Process (proposed model)	Inverse Rayleigh	Exponential	Half Logistic	Gamma
Parameters	$\widehat{a} = 1.0271$	$\widehat{a} = 156.6663$	$\widehat{a} = 122.8602$	$\widehat{a} = 122.8363$	$\widehat{a} = 126.3762$
estimation	$\widehat{b} = 0.3549$	$\widehat{b} = 96.2521$	$\widehat{b} = 0.1979$	$\widehat{b} = 0.2342$	$\widehat{b} = 0.2492$
$-\log L$	238.6755	245.6702	248.1528	238.2602	349.2658
MSE	2.2212	700.3687	208.8905	196.510	15.7047
AIC	463.2522	495.2407	502.4066	470.6207	487.0257
BIC	484.6849	499.4615	508.7401	241.7281	489.8414

Table 6: Refined MMLE and GOF criteria for the 3rd data

Distributions	Rayleigh Process (proposed model)	Inverse Rayleigh	Exponential	Half Logistic	Gamma
Parameters	$\widehat{a} = 5.0520$	$\widehat{a} = 68.2161$	$\widehat{a} = 61.117$	$\widehat{a} = 63.2061$	$\widehat{a} = 126.3762$
estimation	$\widehat{b} = 0.0014$	$\widehat{b} = 16.1002$	$\widehat{b} = 0.5253$	$\widehat{b} = 0.3358$	$\widehat{b} = 0.2492$
$-\log L$	228.5755	235.6712	238.2528	248.2612	249.2658
MSE	8.5101	20.2086	504.8928	103.4418	15.7047
AIC	363.1522	395.1407	302.4166	370.6207	387.0257
BIC	284.5849	399.5615	408.6401	341.6281	389.7414

Table 7: Refined MMLE and GOF criteria for the 4th data

Distributions	Rayleigh Process (proposed model)	Inverse Rayleigh	Exponential	Half Logistic	Gamma
Parameters	$\widehat{a} = 5.0520$	$\widehat{a} = 68.2161$	$\widehat{a} = 61.117$	$\widehat{a} = 63.2061$	$\widehat{a} = 126.3762$
estimation	$\widehat{b} = 0.0014$	$\widehat{b} = 16.1002$	$\widehat{b} = 0.5253$	$\widehat{b} = 0.3358$	$\widehat{b} = 0.2492$
$-\log L$	228.5755	235.6712	238.2528	248.2612	249.2658
MSE	2.6905	20.2086	504.8928	42.0143	1.2239
AIC	163.0522	195.0407	202.4166	170.6207	187.0257
BIC	184.4849	299.4615	308.5401	241.5281	289.6414

MSE values for different distribution, models are displayed in the table against the Rayleigh Process (proposed model) and Inverse Rayleigh and Exponential and Half Logistic and Gamma

Table 8: Comparison of MSE Values Across Different Methods

Unite	Method	Rayleigh Process (proposed model)	Inverse Rayleigh	Exponential	Half Logistic	Gamma
Data set 1	MMLE	0.0200	102.4486	91.5324	71.3439	914.2080
	Bay	0.1120	104.5576	98.5214	72.4528	915.2191
Data set 2	MMLE	2.2212	700.3687	208.8905	196.510	15.7047
	Bay	4.2312	702.4798	210.9815	198.610	16.8158
Data set 3	MMLE	8.5101	20.2086	504.8928	103.4418	15.7047
	Bay	9.6201	21.3196	505.9839	104.5527	16.8136
Data set 4	MMLE	2.6905	20.2086	504.8928	42.0143	3.2239
	Bay	3.7815	21.3076	505.9837	43.1243	4.3249

distributions using MMLE and Bayesian estimation across four datasets. The Rayleigh Process model maintains steady MSE values which prove advantageous to other models, particularly during MMLE executions. For Data Set 1, the Rayleigh Process achieves an MSE of 0.0200 using MMLE that exceeds the MSE values of Inverse Rayleigh 102.4486 and other models. MMLE delivers better fitting results than Bayesian estimation when applied to Data Set 2 because its MSE outcome 2.2212 registers lower than the, Bayesian prediction 4.2312. Data Set 3 benefits from MMLE because it generates more accurate results MSE of 8.5101, than Bayesian estimation 9.6201a according to the comparison results. The MMLE estimation method generates more precise results by delivering smaller MSE values as it proves , more efficient estimation procedure than Bayesian estimation in these datasets. and illustrated in Figure 4.

We observe from the figure above, across the four shapes, that the proposed method outperforms the other distributions. Experimental data validate the superior capability of our proposed model for parameter forecasting and accuracy prediction. Our work summarizes the gathered results and explores future research pathways while discussing practical implications in the final part.

7 Conclusions and Future Work

A new Software Reliability Growth Model was developed by combining the NHPP and the Rayleigh process to establish a mean function, while MMLE was modified through ANN to improve parameter estimation accuracy. The research used the new SRGM based on the Rayleigh process mean function to show effectiveness by comparing performance with four standard SRGMs (Inverse Rayleigh, Exponential, Half-Logistic, and Gamma models) on simulated and real-world software failure datasets. Throughout the research, MMLE showed better parameter estimation through lower MSE than Bayesian estimation methods on all tested models and sample sets while Rayleigh-based NHPP demonstrated the most reliable results, especially in cases with limited data. The main benefit

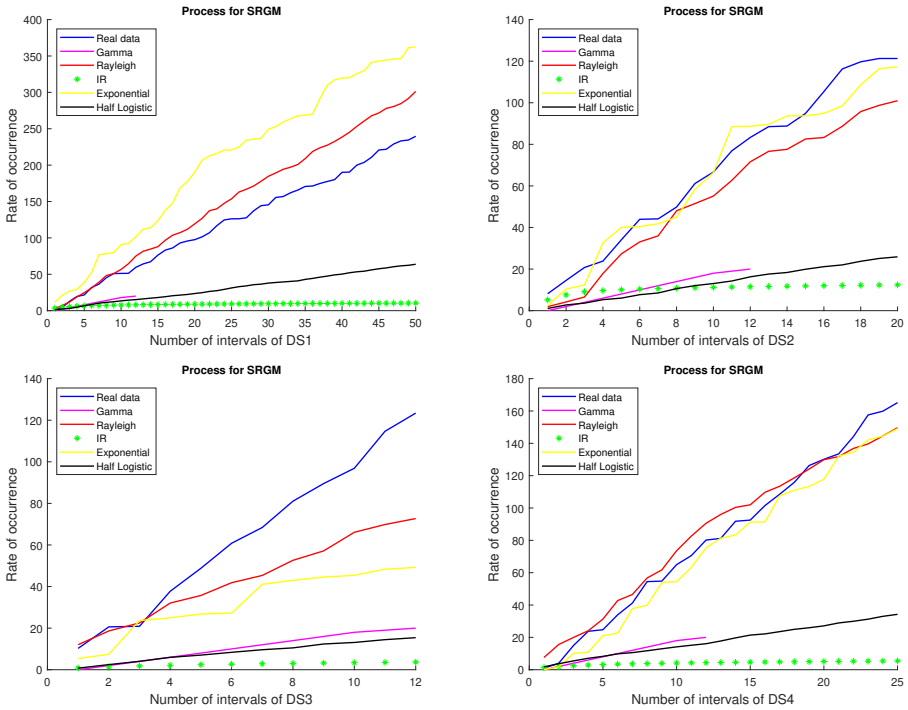


Figure 4: Plot the simulated mean value functions for the five distributions using different sample sizes and varying parameters.

of Bayesian estimation lies in its ability to incorporate previous knowledge yet its parameter estimate variability resulted in slightly greater MSE across most examined cases. The use of Bayesian estimation demonstrated better effectiveness in large samples when appropriate priors were utilized. This resulted in diminishing performance differences between MMLE and Bayesian estimation. MMLE emerges as the most accurate method for parameter estimation, although Bayesian estimation fills a valuable role in quantifying uncertainty parameters when applying prior knowledge. The research demonstrates MMLE achieves better results, yet additional improvement options exist for future work. Research on Bayesian Variational Inference, together with MCMC systems, should be investigated to upgrade Bayesian estimation while decreasing computational costs. Researchers should create a combined MMLE Bayesian framework by using MMLE-generated estimates to establish informative priors for Bayesian inference to harness the potential benefits of both approaches. Introducing time-dependent variables such as software complexity and workload intensity into NHPP models would lead to enhanced reliability prediction accuracy due to their ability to measure changing factors affecting failure risk. Better SRGM parameter estimation can be achieved through the advancement of adaptive neural network architecture and its deep learning techniques transformers and reinforcement learning. Future research must explore our developed model's validation process on industrial-scale programs, cloud-based applications, and real-time systems.

Acknowledgment

The authors are very grateful to the Duhok of Polytechnic University for providing access which allows for more accurate data collection and improved the quality of this work. The data supporting the results of this study are publicly available. This research received no external funding. The authors declare no conflicts of interest.

References

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018), "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, 4(11), e00938.

Chupradit, S., Tashtoush, M., Ali, M., AL-Muttar, M., Sutarto, D., Chaudhary, P., Mahmudiono, T., Dwijendra, N., and Alkhayyat, A. (2022), "A Multi-Objective Mathematical Model for the Population-Based Transportation Network Planning," *Industrial Engineering & Management Systems*, 21(2), 322–331.

Chupradit, S., Tashtoush, M., Ali, M., AL-Muttar, M., Widjaja, G., Mahendra, S., Aravindhan, S., Kadhim, M., Fardeeva, I., and Firman, F. (2023), "Modeling and Optimizing the Charge of Electric Vehicles with Genetic Algorithm in the Presence of Renewable Energy Sources," *Journal of Operation and Automation in Power Engineering*, 11(1), 33–38.

Colak, A. (2024), "A comparative analysis of maximum likelihood estimation and artificial neural network modeling to assess electrical component reliability," *Quality and Reliability Engineering International*, 40(1), 91–114.

De Oliveira, M. D., Colosimo, E. A., Gilardoni, and L., G. (2012), "Bayesian inference for power law processes with applications in repairable systems," *In 2022 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM)*, 447–452.

Groeneboom, P., Jongbloed, G., and Witte, B. I. (2010), "Maximum smoothed likelihood estimation and smoothed maximum likelihood estimation in the current status model," *The Annals of Statistics*, 38(1), 352—-387.

Huang, Y., Chiu, K., and Chen, W. (2022), "A software reliability growth model for imperfect debugging," *J. Syst. Softw*, 188, 111267.

Hussain, A. (2022), "Estimation of Rayleigh Process Parameters Using Classical and Intelligent Methods with Application," *In 2022 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM)*, 447–452.

Hussain, A., Mahmood, K., Ibrahim, I., Jameel, A., Nawaz, S., and Tashtoush, M. (2025a), "Parameters Estimation of the Gompertz-Makeham Process in Non-Homogeneous Poisson Processes: Using Modified Maximum Likelihood Estimation and Artificial Intelligence Methods," *Mathematics and Statistics*, 13(1), 1–11.

Hussain, A., Oraibi, Y., Mashikhin, Z., Jameel, A., Tashtoush, M., and Az-Zo'bi, E. (2025b), "New Software Reliability Growth Model: Piratical Swarm Optimization -Base Parameter Estimation in Environments with Uncertainty and Dependent Failures," *Statistics, Optimization & Information Computing*, 13(1), 209–221.

Hussain, A., Pati, K., Atiyah, A., and Tashtoush, M. (2025c), "Rate of Occurrence Estimation in Geometric Processes with Maxwell Distribution: A Comparative Study between Artificial Intelligence and Classical Methods," *International Journal of Advances in Soft Computing and Its Applications*, 17(1), 1–15.

Hussein, H., Hussein, S., Hussain, A., and Tashtoush, M. (2025), "Estimating Parameters of Software Reliability Growth Models Using Artificial Neural Networks Optimized by the Artificial Bee Colony Algorithm Based on a Novel NHPP," *Mathematical Modelling of Engineering Problems*, 12(1), 25–36.

Inoue, S. (2004), "Testing coverage dependent software reliability growth modeling," *International Journal of Quality Reliability and Safety Engineering*, 11(4), 303–312.

Kim, M. and Lee, S. (2022), "Maximum composite likelihood estimation for spatial extremes models of Brown–Resnick type with application to precipitation data," *Scand. J. Stat*, 49, 1023—1059.

— (2025), "Maximum likelihood estimation of elliptical tail," *Journal of Multivariate Analysis*, 205, 105382.

Lee, D. (2023), "Study of a New Software Reliability Growth Model under Uncertain Operating Environments and Dependent Failures," *Mathematics*, 11(18), 3810.

Luo, H., Xu, L., He, L., Jiang, L., and Long, T. (2023), "A Novel Software Reliability Growth Model Based on Generalized Imperfect Debugging NHPP Framework," *IEEE Access*, 11, 71573—-71593.

Musa, J. (1987), "Software Reliability Engineering," *IEEE Software*, 4(2), 21–28.

Nguyen, H. (2022), "New non-homogeneous Poisson process software reliability model based on a 3-parameter S-shaped function," *IET Software*, 16(2), 214–232.

Ning, J., Pang, S., Arifin, Z., Zhang, Y., Epa, U. P. K., Qu, M., and ang, H. (2024), "The Diversity of artificial intelligence applications in marine pollution: A Systematic literature review," *Journal of Marine Science and Engineering*, 12(7), 1181.

Okumoto, K. (1979), "A time dependent error detection rate model for software reliability and other performance," *IEEE Transactions on Reliability*, 28(3), 206—211.

Rao, B. (2011), "Software reliability growth model based on half logistic distribution," *Journal of Testing and Evaluation*, 39(6), 1.

— (2013), "Inverse Rayleigh software reliability growth model," *International Journal of Computer Applications*, 75(6), 0975–8887.

Ravikumar, K., Chethan, Y., Likith, C., and Chethan, S. (2024), "Prediction of Wear Characteristics for Al-MnO2 Nanocomposites using Artificial Neural Network," *Materials Today: Proceedings*, 98, 102–108.

Shafiq, A. (2022), "Reliability analysis based on mixture of Lindley distributions with artificial neural network," *Advanced Theory and Simulations*, 5(8), 2200100.

— (2023), "Modeling and survival exploration of breast carcinoma: a statistical, maximum likelihood estimation, and artificial neural network perspective," *Artificial Intelligence in the Life Sciences*, 4, 100082.

Shirawia, N., Kherd, A., Bamsaoud, S., Tashtoush, M., Jassar, A., and Az-Zo'bi, E. (2024), "Dejdumrong Collocation Approach and Operational Matrix for a Class of Second-Order Delay IVPs: Error Analysis and Applications," *WSEAS Transactions on Mathematics*, 23, 467–479.

Sindhu, T. (2023), "Reliability study of generalized exponential distribution based on inverse power law using artificial neural network with Bayesian regularization," *Quality and Reliability Engineering International*, 39(6), 2398–2421.

Wang, Y., Dai, H., Chen, Z., He, S., Wang, W., and Gao, M. (2024), "Simulation study on a novel solid-gas coupling hydrogen storage method for photovoltaic hydrogen production systems," *Energy Conversion and Management*, 299, 117866.

Wood, A. (1996), "Predicting software reliability," *IEEE Computer*, 11:69–77.

Wu, C., Wang, Q., Wang, X., Sun, S., Bai, J., Cui, D., and Sheng, H. (2024), "Effect of Al2O3 nanoparticle dispersion on the thermal properties of a eutectic salt for solar power applications: Experimental and molecular simulation studies," *Energy*, 288, 129785.

Yue, Z., Shen, X., Chang, Z., and Zou, G. (2024), "Experimental and simulation study on quasi-static and dynamic fracture toughness of 2A12-T4 aluminum alloy using CTS specimen," *Theoretical and Applied Fracture Mechanics*, 133, 104556.

Received: October 05, 2024

Accepted: April 20, 2025

A Algorithm

Algorithm 1 Neural Network Training with Hessian-Based Optimization

Require: Training data $\{(x_i, y_i)\}_{i=1}^{n}$, activation functions, step size a, learning parameter b, threshold ϵ, adjustment factor B

Ensure: Optimized weights and biases

Initialize weights and biases: $w, s, r, c \leftarrow$ random or heuristic values
Select hyperparameters a, b
Define objective function using Equations (3.6) and (3.7)
repeat
  for each input x_i **do**
    Compute input to first hidden layer: $h_{\text{in}}^{(1)} \leftarrow \sum w_i x_i + b$
    Compute output of first hidden layer: $h_{\text{out}}^{(1)} \leftarrow \text{activation}(h_{\text{in}}^{(1)})$
    Compute input to second hidden layer: $h_{\text{in}}^{(2)} \leftarrow \sum s_i h_{\text{out}}^{(1)} + b$
    Compute output of second hidden layer: $h_{\text{out}}^{(2)} \leftarrow \text{activation}(h_{\text{in}}^{(2)})$
    Compute input to output neuron: $o_{\text{in}} \leftarrow \sum r_i h_{\text{out}}^{(2)} + c$
    Compute network output: $\hat{y}_i \leftarrow \text{activation}(o_{\text{in}})$
  end for
  Compute Mean Square Error: $MSE \leftarrow \frac{1}{n-N} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
  if $MSE < \epsilon$ **then**
    Stop training, finalize weights and biases
  end if
  Compute gradients g_k of weights and biases
  Compute inverse Hessian approximation M_k^{-1}
  Update output layer: $r_{\text{new}} = r_{\text{old}} - aM_k^{-1} g_k, c_{\text{new}} = c_{\text{old}} - aM_k^{-1} g_k$
  Improve update using Hessian-based method:

$$r_{\text{new}} = r_{\text{old}} - \left[L_k^T L_k + bI\right]^{-1} g_k, c_{\text{new}} = c_{\text{old}} - \left[L_k^T L_k + bI\right]^{-1} g_k$$

  Update first hidden layer:

$$w_{k+1} = w_k - \left[L_k^T L_k + bI\right]^{-1} g_k, c_{k+1} = c_k - \left[L_k^T L_k + bI\right]^{-1} g_k$$

  Update second hidden layer:

$$s_{k+1} = s_k - \left[L_k^T L_k + bI\right]^{-1} g_k, c_{k+1} = c_k - \left[L_k^T L_k + bI\right]^{-1} g_k$$

  Re-evaluate MSE
  if $MSE_{\text{new}} \leq MSE_{\text{old}}$ **then**
    Update $M_{\text{new}} \leftarrow M_{\text{old}}/B$
    Go to Step 3
  else
    Update $M_{\text{new}} \leftarrow M_{\text{old}} \cdot B$
    Go to Step 18
  end if
until convergence