Advances and Applications in Statistics
Volume 29, Number 2, 2012, Pages 81-99
Published Online: November 2012
Available online at http://pphmj.com/journals/adas.htm
Published by Pushpa Publishing House, Allahabad, INDIA

A NEW NON-PARAMETRIC NON-HOMOGENEOUS POISSON PROCESS SOFTWARE RELIABILITY MODEL

May Barghout

Department of Statistics
Faculty of Economics and Political Science
Cairo University
Egypt

Abstract

This paper addresses a family of probability models for the failure time process known as Non-Homogeneous Poisson Process (NHPP) models. Conventional NHPP models make rather strong distributional assumptions about the detection times: typically they assume that these come from some parametric family of distributions. Recently non-parametric models were developed in an attempt to relax these assumptions and - generally, in the tradition of non-parametric statistics - 'allow the data to speak for themselves'. We present a new non-parametric model for reliability prediction, which is based upon the use of the half folded kernel density estimators. The predictive accuracy of the model, using real data sets, is compared with that of Gaussian fix width and Gaussian adaptive kernel density estimators. The initial results are encouraging.

2010 Mathematics Subject Classification: 62N05.
Keywords and phrases: folded normal distribution, half normal distribution, kernel density estimation, non-homogeneous Poisson process models, non-parametric estimation, software reliability growth.
Received June 4, 2012

1. Introduction and Background

The problem with which this paper deals is the common one of assessing the current reliability, and predicting the future reliability, of some software that is undergoing fault removal. The program is assumed to be executing in a real or simulated operational environment. It starts its life with a number of faults, each of which eventually manifests itself by causing a failure. We shall assume here that time is continuous and truly represents the extent to which the software is used (see, for example, [10, 13] for discussions of the use of 'execution time'). At each failure it will be assumed that an attempt is made to remove the cause of the failure (the fault), whereupon the program is put back into operation. Let fault *i* (in an arbitrary labelling) be detected after a time x_i, which is a realisation of a random variable X_i. The $\{X_i\}$ are assumed to be independently identically distributed, i.i.d, random variables. The order statistics

$$0 \leq X_{(1)} \leq X_{(2)} \leq X_{(3)} \leq \cdots$$

of the $\{X_i\}$ represents the successive times of fault detection.

The inter-failure times $T_1, T_2, \ldots$ are the spacing between the order statistics, i.e.,

$$T_1 = X_{(1)},$$

$$T_2 = X_{(2)} - X_{(1)},$$

$$\vdots$$

$$T_i = X_{(i)} - X_{(i-1)}.$$

The successive inter-failure times $t_1, t_2, ..., t_{i-1}$ will therefore tend to show reliability growth as time passes, and will form the data upon which the reliability predictions will be based. This problem has been addressed by many authors over the past twenty years or so (see, for example, [1-5, 12, 22, 24]).

2. NHPP Software Reliability Models

Among the many software reliability models developed over the years several NHPP models have been proposed [1, 2, 9, 20, 24]. These models consider the debugging process as a count process characterised by its intensity function $\lambda(\tau)$ or, equivalently, its mean function $M(\tau)$ given by

$$M(\tau) = \int^{\tau} \lambda(x)dx,$$

where $\tau_{i-1} = \sum_{j=1}^{i-1} t_j$.

Once the intensity function (or mean function) of the process is defined, the inter-failure time distribution can be obtained by applying

$$f(t_i \mid \tau_{i-1}) = \lambda(t_i + \tau_{i-1}) \exp - \{M(t_i + \tau_{i-1}) - M(\tau_{i-1})\}.$$

Let the probability density function, pdf, and the cumulative distribution function, cdf, of the time to detection of a fault be denoted by $g(X)$ and $G(X)$, respectively. Assuming that the total number of initial faults in the program is Poisson distributed with mean μ, the intensity function for the NHPP is given by [17]

$$\lambda(\tau) = \mu g(\tau)$$

with mean function

$$M(\tau) = \mu G(\tau).$$

The conditional distribution for T_i given τ_{i-1} is

$$f(t_i \mid \tau_{i-1}) = \mu g(t_i + \tau_{i-1}) \exp\{-\mu[G(t_i + \tau_{i-1}) - G(\tau_{i-1})]\}.$$

Estimates of the parameters are made at each stage i using previous failure data $t_1, ..., t_{i-1}$. These estimates are then substituted in the cdf and pdf in order to make predictions about the yet unobserved T_i,

$$\hat{f}(t_i \mid \tau_{i-1}) = \hat{\mu}\hat{g}(t_i + \tau_{i-1}) \exp\{-\hat{\mu}[\hat{G}(t_i + \tau_{i-1}) - \hat{G}(\tau_{i-1})]\}.$$

The importance of the above general form is that it expresses the distribution of the independent and non-identical inter failure-times as a function of the distribution of the i.i.d times to detection of faults and μ, the expected number of failures [17].

In general NHPP software reliability models can be classified into two categories; parametric and non-parametric models. All the parametric models make rather strong distributional assumptions about the detection times: typically they assume that these come from some parametric family of distributions. In attempt to relax the parametric assumptions about the detection time distribution some authors considered non-parametric estimation methods for software reliability [16-19, 23, 25].

2.1. Non-parametric NHPP models

Sofer and Miller, as cited in [23], assumed a polygonal line estimator of the software intensity function for the usual Non-Homogeneous Poisson Process (NHPP)-based software reliability models and proposed a smoothing algorithm based on the common quadratic programming. Gandy and Jensen, as cited in [23], used the well-known Nelson-Aalen multiplicative estimator for the NHPP-based software reliability models. Recently, Wang et al. [25] applied the kernel intensity estimation method to develop a non-parametric NHPP-based software reliability models and proposed a quite different approach from the existing ones. Barghout [17] suggested a generalized NHPP software reliability model and proposed a non-parametric method based on the well-known kernel density estimation. Instead of assuming a distribution for the detection times a non-parametric kernel density estimation method was used to estimate this distribution. Three different kernel functions were chosen, namely, the Gaussian, double exponential and the log-normal kernel. The performances of these three kernel functions were investigated and compared on a number of data sets. In general, the analysis revealed poor performance of the Gaussian kernel function [17].

The performance of a kernel estimator is expected to be influenced by the choice of the kernel function, on one hand, and the smoothing parameter

(bandwidth), on the other hand. In attempt to improve the performance of the model a non-parametric density estimation model that is based on an adaptive kernel estimator was proposed [16]. The rationale behind the adaptive kernel is to allow the bandwidth to vary from one point to another and thus, using a broader kernel in regions of low densities. The preliminary results revealed an improvement in the predictive accuracy of the adaptive kernel estimator compared with its fixed width kernel variant on the data sets analysed.

An alternative approach is to focus on the kernel function itself. The use of the Gaussian kernel function has been criticized for being defined on the whole real line, and thus gives an estimated density on the whole real line - in fact the true density is known to be defined on the positive real line [16, 17]. An alternative function, the folded normal function, which is defined only on the positive half of the real line, is, herein, proposed.

2.2. The folded normal NHPP model

Details of kernel estimation can be found elsewhere [6] but the basic idea is a simple one. Assume that we have a sample of n real observations $x_1, x_2, ..., x_n$ whose underlying density is to be estimated. The kernel estimate assumes that around each observation, x_i a kernel function $K_h(x - x_i)$ is centred. The kernel density estimator is then obtained by averaging over these kernel functions in the observation, i.e.,

$$\hat{g}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i),$$

where $K_h(x) = \frac{1}{h}K\left(\frac{x}{h}\right)$ for any general kernel $K(x)$ and h is a smoothing parameter. The kernel density estimator coincides with the characteristic function estimator when the kernel function is the inverse Fourier transformation of the characteristic function [8, 14].

Clearly, the choice of the mathematical form of the kernel function here will be very important.

The folded normal distribution has been widely used in a variety of applications such as quality control [7] and dealing with measurement errors [15]. In general, the distribution of $X = |Z|$, where Z is distributed as $N(\mu, \sigma)$, is a folded normal distribution. The normal probability function is folded at $Z = 0$. The density function of the folded normal distribution is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}[e^{-(x-\mu)^2/2\sigma^2} + e^{-(x+\mu)^2/2\sigma^2}], \quad x \geq 0.$$

In the special case when Z is a standard normal random variable the density function of X reduces to

$$f(x) = \frac{2}{\sqrt{2\pi}} e^{-x^2}, \quad x \geq 0$$

which is known as the half-normal distribution [7, 15, 21] and is to be considered as the kernel function in this paper.

Finally, it is necessary to consider the estimation of the parameter h in the kernel estimator. The most common approach is the likelihood cross validation method. This is a natural development of the idea of using likelihood to judge the adequacy of fit of a statistical model. The rationale behind the method, as applied to density estimation, is as follows. One observation $x_{(j)}$ from the sample used to construct the density estimate is omitted. The non-parametric estimate of $g(x_{(j)})$, $\breve{g}_{-j}(x_{(j)})$, is the density estimate constructed from all the $(i-1)$ data points except $x_{(j)}$, that is to say,

$$\breve{g}_{-j}(x_{(j)}) = (i-2)^{-1}h^{-1}\sum_{k\neq j} K\{h^{-1}(x_{(j)} - x_{(k)})\}.$$

The non-parametric estimate of $f_j(t_j)$, $\breve{f}_{-j}(t_j)$ is given by,

$$\bar{f}_{-j}(t_j \mid \tau_{j-1}) = \mu\tilde{g}_{-j}(t_i + \tau_{i-1})\exp\{-\mu[\tilde{G}_{-j}(t_i + \tau_{i-1}) - \tilde{G}_{-j}(\tau_{i-1})]\}.$$

Since there is nothing special about the choice of which observation to leave out, the log likelihood is averaged over each choice of omitted $x_{(j)}$, to give

$$CV = (i-1)^{-1}\sum_{j=1}^{i-1} \log \breve{f}_{-j}(t_j).$$

The likelihood cross validation choice of the parameters is the value of these parameters which maximise the function *CV* for the given data.

Once the parameters of a model are estimated, at any stage *i* using the previous data, these estimates are then used to estimate the distribution of T_i and to make further ahead predictions of reliability.

3. Brief Resume of Tools for Analysing Predictive Accuracy

Typically models are used to make a sequence of predictions as *i* increases; thus a sequence of successive one-step-ahead predictions, $\hat{F}_i(t)$, of the random variables T_i is generated. The accuracy of models' predictions will solely depend on its estimate, $\hat{F}_i(t)$, of the *true* $F_i(t)$. The true $F_i(t)$ is not known, even at later stages of analysis, hence it is difficult to analyze the closeness of $\hat{F}_i(t)$ to the *true* $F_i(t)$. However, t_i, the realisation of T_i, is later observed and all the analysis of the predictive quality of a model is based upon these pairs $\{\hat{F}_i(t), t_i\}$. There has been considerable progress in using the $\{\hat{F}_i(t), t_i\}$ sequence to assess the accuracy of a particular method of prediction upon a particular data source [1, 2, 22]. These tools will be used to compare the accuracy of the new model with that of the Gaussian fixed width and Gaussian adaptive models. Basically there are two main tools: the *u-plot* and the *prequential likelihood ratio* (*PLR*).

3.1. The *u*-plot

The *u-plot* is a technique for detecting systematic 'bias' in a series of model predictions. It is well known that if the random variable T_i truly had

the distribution $\hat{F}_i(t)$, then the random variable $U_i = \hat{F}_i(T_i)$ would be uniformly distributed on $(0, 1)$. If we were to observe the realisation t_i of T_i, and calculate $u_i = \hat{F}_i(t_i)$, then the number u_i will be a realisation of a uniform random variable. When we do this for a sequence of predictions, then, we should get a sequence $\{u_i\}$, which looks like a random sample from a uniform distribution. Any departure from such uniformity will indicate some kind of deviation between the sequence of predictions, $\{\hat{F}_i(t)\}$ and the truth $\{F(t)\}$. One way of looking for departure from uniformity is by plotting the sample distribution function for the $\{u_i\}$ sequence. If the $\{u_i\}$ sequence were truly uniform, then this plot should be close to the line of unit slope. Any serious departure of the plot from this line indicates that the predictions are inaccurate: one way of testing whether such a departure is statistically significant is via the Kolmogorov, K-S, distance [1, 2, 20].

3.2. The prequential likelihood ratio (PLR)

The PLR is a very general and powerful means of comparing the accuracy of sequences of predictions coming from two competing prediction systems *A* and *B*. For the one-step-ahead predictions of T_i that we are considering here, the PLR compares the successive *probability density function* estimates $\hat{f}_i^A(t_i)$ and $\hat{f}_i^B(t_i)$, each evaluated at the *later observed* t_i. The PLR is a running product of the ratio of these over many successive predictions:

$$PLR_j^{AB} = \prod_{i=s}^{i=j} \frac{\hat{f}_i^A(t_i)}{\hat{f}_i^B(t_i)},$$

where *s* is the starting sample size.

There is a sophisticated asymptotic theory underpinning the use of the PLR, which is beyond the scope of the present paper, but the following is an intuitive explanation of how it works. Let us assume that the *A* predictions actually are accurate, and the *B* predictions are not: i.e., the predictive

densities from *A* are always close to the true ones. The observation t_i will tend to take a value where the true density (and hence the accurate *A* prediction) is large (e.g., rather than in the tail of the true density), and where the *B* predictive density is not so large. Thus, if the predictions from *A* are more accurate than those from *B*,

$$\hat{f}_i^A(t_i)/\hat{f}_i^B(t_i)$$

will tend to be larger than 1. The PLR, which is a running product of such terms, will therefore tend to increase with *i* if the *A* predictions are better than the *B* predictions: if we detect such a clear trend upwards we shall have evidence of the superiority of the *A* predictions [1, 2, 20].

4. Application of the Model to Real Software Failure Data

The predictive accuracy of the new model is examined on four real data sets using the techniques previously described. The performance of the folded normal kernel predictor is compared with both the Gaussian fixed width kernel and the adaptive kernel. For convenience these models will be referred to by folded, fix width and adapt, respectively.

Due to space constraints it is necessary to limit the number of plots shown: a detailed analysis will be carried out on the first data set while the results on the other three data sets will be summarised.

The first data set Musa system 1 is a set of 136 failures collected by Musa [11]; this data set has been widely used in the software reliability community for model comparison. The second and third data sets are part of a larger data set collected at the Centre for Software Reliability at the City University of London. The original data set USBAR consists of 397 inter-failure times recorded on a single-user workstation. The set was subdivided according to different categories, such as the usage under which the failures occurred or the type of fault that caused the failure. The second data set, USPSCL consists of 104 failures which occurred when compiling and running Pascal programs. The third data set TSW includes 129 software failures which occurred when the system software did not behave as

required, e.g., incorrect output, operating system crash etc. The models will then be applied to the original data set USBAR.

The analyses for all data sets are conducted in the same manner. If the data set contains n inter-failure times, $s < n$ is chosen as a starting sample size where s is a number suitably large for making the first prediction and small enough so that the remaining data is sufficient to obtain a sequence of predictions of reasonable size to allow a proper analysis. In all data sets considered s was taken equal to 35. At stage i, a model is used to predict for the yet unobserved T_i based upon observations $t_1, t_2, ..., t_{i-1}$. The sample is thereafter increased by t_i and the observations $t_1, t_2, ..., t_i$ are used to predict for T_{i+1}. As the data evolves we can repeatedly make one step ahead predictions from each model.

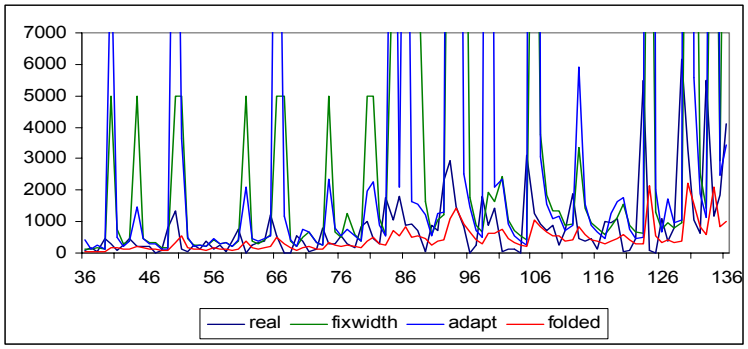

Figure 1. Successive one-step ahead median predictions from the models, of the time to next failure T_i, i = 36 to 136, plotted against i, for data set Musa system 1.

Figure 1 shows plots of the successive current median times to next failure for Musa system 1 Data as calculated by the models. Thus, in the plot at stage i the predicted median of T_i is calculated based upon all the data that has been observed prior to this stage, i.e., inter-failure times $t_1, t_2, ..., t_{i-1}$. The actual data is superimposed upon the median predictions for comparison. Figure 1 shows that there is great disagreement between the different predictors in how they predict the medians.

The median predictions from the folded model exhibit less noise and are in general lower than the predictions obtained from both the fix width and adapt models. Moreover, they are lower than the real inter-failure times. This suggests that the predictions of the folded model are in general too pessimistic.

The bias of the 3 predictors is investigated by studying their *u*-plots, Figure 2. The *u*-plots indicate that although all the models are optimistic for predictions associated with the left hand tail of the distribution of time to next failure they differ when predicting for the right hand tail: the folded predictions are pessimistic as evident by their plots crossing the line of unit slope from the right. The *u*-plots for the fix width and adapt models are everywhere above the line of unit slope, i.e., the predictions are too optimistic, as suspected from the earlier median plots.

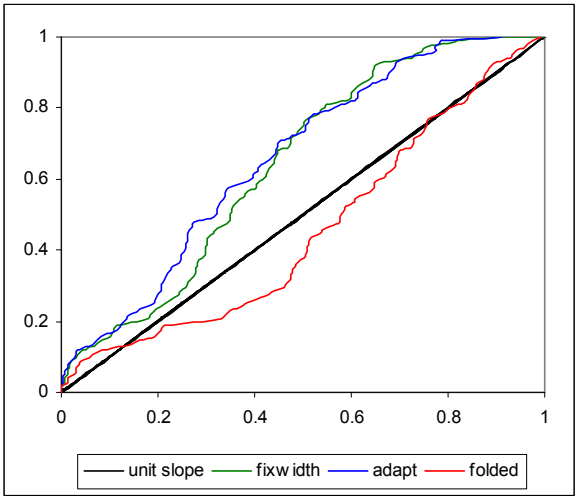

Figure 2. *u*-plots for predictions T_i, i = 36 to 136 from the models for Musa system 1 data.

Table 1 shows summarized results for the K-S distances of the *u*-plots. Although a substantial improvement is noticed in the K-S distance of the *u*-plot of the folded model all the predictors are biased in their predictions with *u*-plots significant at 1%.

Table 1. *u*-plot K-S distances for the different predictors and data sets

	Fix width	Adapt	Folded
Musa system 1	0.269	0.261	0.164
USPCL	0.246	0.251	0.136
TSW	0.273	0.237	0.114
USBAR	0.268	0.268	0.083

The predictions of the folded model are not only less biased than those obtained from both the fix width and adapt models but are in general more accurate as indicated by the PLR plots.

Figure 3 shows the PLR plot for the fix width and adapt predictors against the folded predictor. The folded predictor is chosen as the reference model. Any downward trend (notice that the plot is of the log of PLR) indicates the superiority of the folded model.

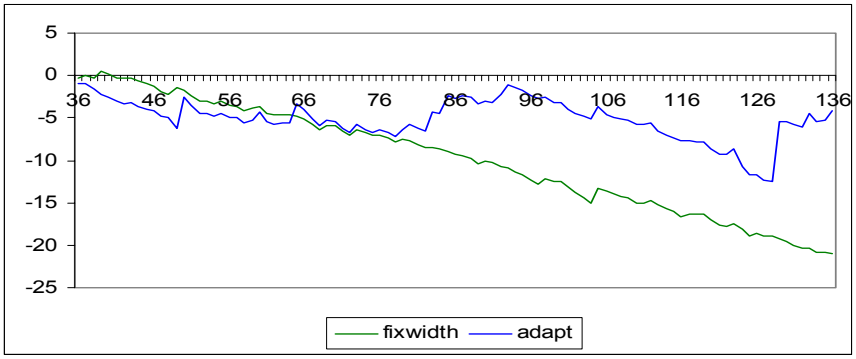

Figure 3. log (PLR) plots for predictions T_i, i = 36 to 136 from the fix width and adapt models versus the folded model, for data set Musa system 1.

The jump exhibited by the adapt predictor near the end of the data set occurred when predicting for an inter-failure time which is far higher than the highest previously observed inter-failure time. Excluding this jump, the plots for the fix width and adapt predictions show a downward trend discrediting these predictors in favour of the folded predictor.

Figure 4 shows the *u*-plots of the different predictors for USPSCL data.

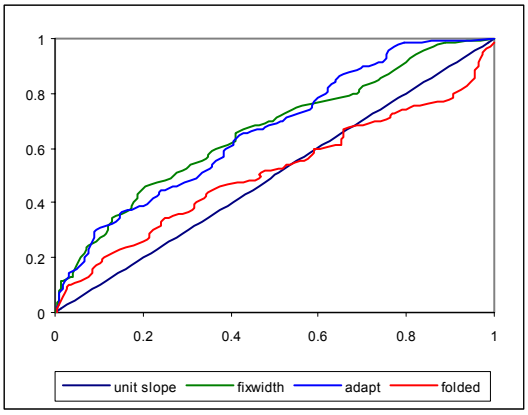

Figure 4. *u*-plots for predictions T_i, i = 36 to 104 from the models for USPCL system 1 data.

As for Musa system 1 data, the *u*-plots for both the fix width and adapt models are everywhere above the line of unit slope suggesting that these predictors are optimistic in their predictions. On the other hand, the folded plot crosses the line of unit slope from the right indicating that its predictions are optimistic for predictions associated with the left hand tail of the distribution of time to next failure and pessimistic when predicting for the right hand tail. This is more obvious on this data set compared to Musa system 1 data. In general the *u*-plot for the folded predictions is in close agreement with the line of unit slope as evident by its K-S distance shown in Table 1. While both the fix width and adapt plots are significant at 1% the K-S distance for the folded plot is insignificant at 20%.

Figure 5 shows the PLR plot for the fix width and adapt predictors against the folded predictor. Once again, the folded predictor is chosen as the reference model and any downward trend (notice that the plot is of the log of PLR) indicates the superiority of the folded model. There is a clear downward trend in the fix width and adapt plots, indicating that the folded model is producing more accurate predictions than both the fix width and adapt model.

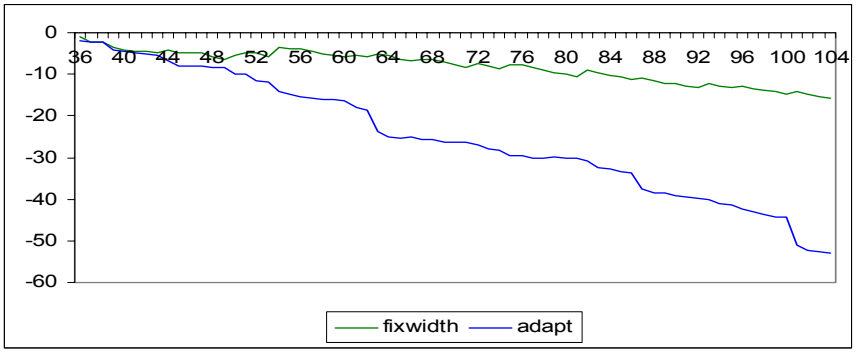

Figure 5. log (PLR) plots for predictions T_i, i = 36 to 104 from the adapt and folded models versus the fix width model, for data set USPSCL.

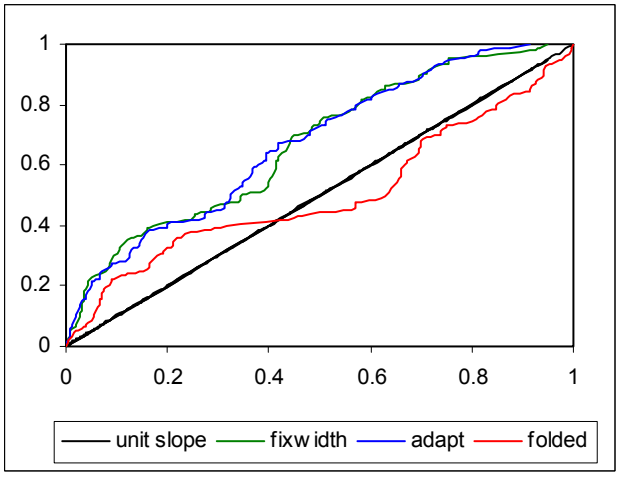

Figure 6. *u*-plots for predictions T_i, i = 36 to 129 for TSW data.

The *u*-plots, Figure 6 and Figure 7, for the TSW and USBAR data, respectively, support the previous analysis. The *u*-plots for the fix width and adapt predictors are in close agreement with one another and are entirely above the line of unit slope indicating that their predictions are too optimistic. The folded model predictions are optimistic when predicting for the left hand tail of the distribution and pessimistic when then right hand tail is considered. In general, the folded plots are closer to the line of unit slope as evident from the K-S distances reported in Table 1. While the K-S

distances for both the fix width and adapt plots are significant at 1% for both data sets the K-S distances for the folded plot are insignificant at 5% level and 1% level for TSW and USBAR data sets, respectively.

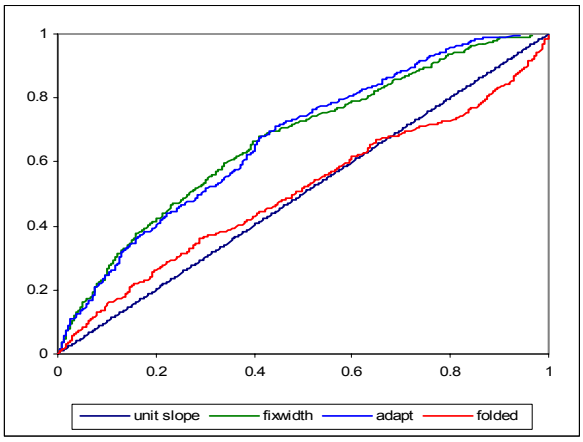

Figure 7. *u*-plots for predictions T_i, i = 36 to 397 for USBAR data.

Figures 8 and 9 show the PLR plots for the fix width and adapt models predictions versus the folded predictions for TSW and USBAR data sets, respectively. Both the fix width and adapt plots exhibit a clear downward trend discrediting these models in favour of the folded model. This is evident for both data sets.

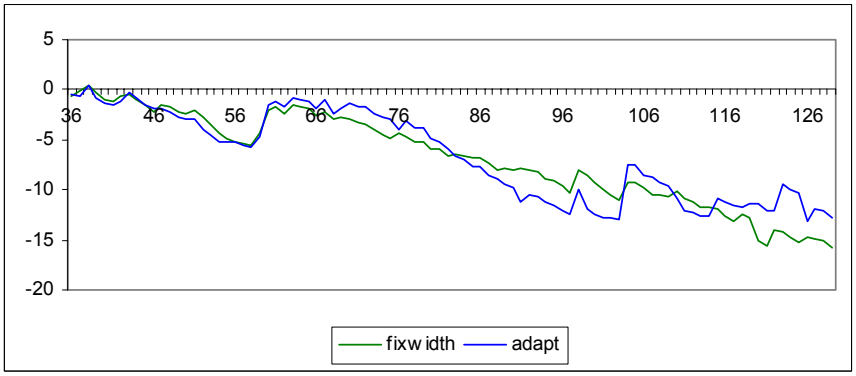

Figure 8. log (PLR) plots for predictions T_i, i = 36 to 129 from the fix width and adapt models versus the folded model, for data set TSW.

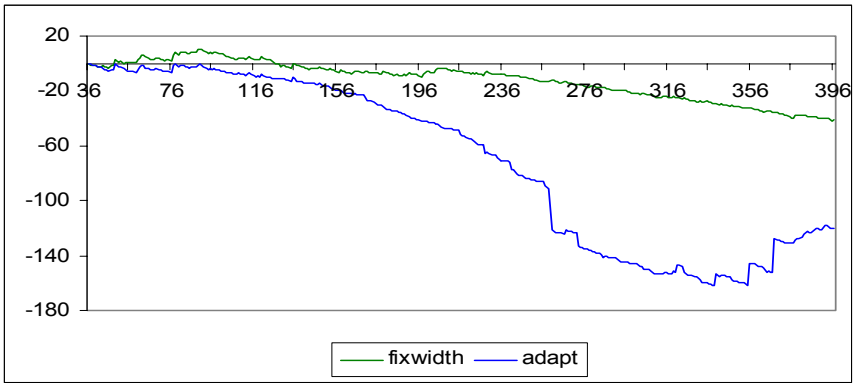

Figure 9. log (PLR) plots for predictions T_i, i = 36 to 397 from the adapt and folded models versus the fix width model, for data set USBAR.

5. Summary and Conclusions

Earlier non-parametric density estimation models used kernel estimators to estimate the distribution for the detection of failure times [16-19]. The performances of different kernel functions were compared on a number of real data sets [16-19]. Previous studies reported poor performance of the Gaussian kernel function with a fixed bandwidth [17]. In attempt to improve the performance of the model an adaptive kernel estimator was proposed [16]. The initial results were not as promising as desired. An alternative approach is to focus on the kernel function itself. One disadvantage of the Gaussian kernel is that it is a symmetric function defined on the whole real line, and thus gives an estimated density on the whole real line - in fact the true density is known to be defined on the positive real line. To overcome this disadvantage a half folded normal function is proposed as the kernel function.

As for the earlier non-parametric models the performance of the new model, folded, is investigated and compared with the Gaussian fixed width and the adaptive Gaussian models on four real data sets.

The predictions of the folded model are not only less biased than those of the fix width and adapt models, as indicated by their K-S distance, but the

nature of the biased is different. While the fix width and adapt predictions are too optimistic the folded predictions are optimistic for predictions of the left hand tail of the distribution and pessimistic for predictions of the right hand tail. This is evident from the *u*-plots crossing the line of unit slope from the right.

The PLR plots suggest that, in general, the predictions of the folded model are more accurate than the predictions of both the fix width and adapt models. The plots of both the fix width and adapt predictions versus the folded predictions exhibited a clear downward trend, in all data sets, discrediting both the fix width and adapt models in favour of the folded model on all data sets. Although the preliminary results are encouraging it is evident that more research and analysis of software failure data is required to obtain more conclusive results.

References

[1] A. A. Abdel-Ghaly, Analysis of predictive quality of software reliability models, Ph.D. Dissertation, City University, London, 1986.

[2] A. A. Abdel-Ghaly, P. Y. Chan and B. Littlewood, Evaluation of competing software reliability predictions, IEEE Trans. Software Engineering 12(9) (1986), 950-967.

[3] A. L. Goel and K. Okumoto, Time-dependent error-detection rate model for software and other performance measures, IEEE Trans. Reliability 28(2) (1979), 206-211.

[4] B. Littlewood, Stochastic reliability growth: a model for fault removal in computer programs and hardware designs, IEEE Trans. Reliability 30 (1981), 313-320.

[5] B. Littlewood and J. L. Verrall, A Bayesian reliability growth model for computer software, J. Royal Statist. Soc. C 22 (1973), 332-346.

[6] B. W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, 1986.

[7] C. Chou and H. Liu, Properties of the half-normal distribution and its application to quality control, Journal of Industrial Technology (1998), 1-5.

[8] E. Lukacs, Characteristic Functions, 2nd ed., Charles Griffin & Company Limited, London, 1970.

[9] H. Pham, L. Nordmann and X. Zhang, A general imperfect software reliability debugging model with S-shaped fault detection rate, IEEE Transaction on Reliability 48(2) (1999), 159-168.

[10] J. D. Musa, Operational profiles in software-reliability engineering, IEEE Software 10(2) (1993), 14-32.

[11] J. D. Musa, Software Reliability Data, Technical Report, Data Analysis Centre for Software, Rome Air Development Centre, Griffins, AFB, New York, 1979.

[12] J. D. Musa, and K. Okuumoto, A Logarithmic Poisson Execution Time Model for Software Reliability Measurement, Proc. COMSAC 84, Chicago (1994), 230-238.

[13] J. D. Musa, A. Iannino and K. Okumoto, Software Reliability: Measurements, Prediction Application, McGraw-Hill, 1987.

[14] J. F. Fames, A Student's Guide to Fourier Transforms with Application in Physics and Engineering, 3rd ed., Cambridge University Press, New York, 2011.

[15] J. M. Bland. The half-normal distribution method for measurement error: two case studies, Sciences New York (2005), 1-14.

[16] M. Barghout, A non-parametric NHPP software reliability model, Adv. Appl. in Stat. 14(1) (2010), 31-47.

[17] M. Barghout, Software reliability: A non-parametric approach, Ph.D. Dissertation, Cairo University, 1998.

[18] M. Barghout, B. Littlewood and A. A. Abdel-Ghaly, A non-parametric order statistics software reliability model, software testing, Verification and Reliability 8 (1998), 113-132.

[19] M. Barghout, B. Littlewood and A. A. Abdel-Ghaly, A non-parametric approach to software reliability prediction, Proceedings of the 8th International Symposium on Software Reliability Engineering, IEEE Computer Society (1997), 366-377.

[20] M. Lyu, Handbook of Software Reliability Engineering, McGraw-Hill, New York, 1995.

[21] R. C. Elandt, The folded normal distribution: two methods of estimating parameters from moments, Technometrics 3(4) (1961), 551-562.

[22] S. Brocklehurst and B. Littlewood, New ways to get accurate reliability measures, IEEE Software 9(4) (1992), 34-42.

[23] M. Shintaro and T. Daho, A refined non-parametric algorithm for sequential software reliability estimation, ASEA, CCIS 59, (2009), 330-337.

[24] Z. Jelinski and P. B. Moranda, Software reliability research, Statistical Computer Performance Evaluation, W. Freiberger, ed., Academic Press, New York, 1972, pp. 465-484.

[25] Z. Wang, J. Wang and X. Liang, Non-parametric estimation for NHPP software reliability models, J. Appl. Stat. 34(1) (2007), 107-119.