Empirical Analysis of Merge sort in Personal Computer by Curve Fitting Technique

Arijit Chakraborty [#], Avik Mitra [#], Dipankar Das[#]

[#] *Department of BCA, The Heritage Academy*
Chowbaga, Anadapur, East Kolkata Township, Kolkata-700107, India

Abstract—**In this paper, empirical analyses of mergesort algorithm has been performed to observe its run-time behaviour in personal computer. It has been observed that the behavioural pattern of its worst case matches best with series of Fourier curves, and we found that Fourier series with five harmonics matches the best.**

***Keywords*— *Curve fitting, experimental algorithmics, Fourier fit, merge sort, performance analysis, residual analysis, worst case*.**

I. INTRODUCTION

Empirical algorithmics *(*also called experimental algorithmics*)* is the area in computer science that uses empirical methods *to study the behavior of* algorithms. *It can be used for run-time analysis of algorithms* [29]. There are two main types of empirical algorithmics: i) one deals with analyzing behavioral performance of algorithms; ii) second one deals with empirical methods for its performance improvement. In this paper, we have explored Mergesort [30], which is a divide and conquer algorithm which is also comparison based algorithm. Our aim is to identify curve(s) that fits in empirical timeline based data of mergesort in its worst case where we found that a *series of Fourier curves* are matching with the empirical data.

II. RELATED WORKS

Mergesort can be taken to operate in three phases:

- *Phase 1:* Dividing a finite list of elements into sub-lists recursively until the number of elements in each sub-list is less than or equal to a threshold m, and sorting each of these sub-lists.
- *Phase 2:* Taking k number of sub-lists at a time and merging using appropriate mechanisms repeatedly. This phase can be termed as *k-way merge process.*
- *Phase 3:* Creating the final sorted list. This phase is an extension of phase 2.

The implementation of first two phases and the computational environment used to implement the algorithm results in the variation in its actual run-time. Thus, the research works on Mergesort can be classified primarily into two categories:

- Development of variations of mergesort.[1]-[8] This category corresponds to phase 1 and phase 2 of mergesort.
- Analyzing the effects of computational resources on mergesort and/or its variants [9]-[12]. This category can be termed as *external effects on mergesort*.

The variants of mergesort can be further classified into whether the modifications are proposed for phase 1 [1], phase 2 [2]-[7], or both the phases [8]. To make merging process of sub-lists, i.e. phase 2, more efficient, the authors in [2] and [5] have used binary search to find the place in the second list where an element in the first list could be inserted. To make phase 2 faster, [3] proposed to start the merging process of two sorted lists as soon as one of the two sorted lists and one element of the other list are available; this compensates the delay caused by differences in the processor speeds. To make the merging process optimal, [4] uses binomial search trees in the merging process and makes the final sorted list also as a binomial search tree. To reduce the number of elements in processing queries in database applications, [6] uses a linear scan after each merge process to eliminate duplicate entries. Each of these merge process uses 2-way merge. However [7] points out that 4-way merge process performs best amongst 2-way, 3-way and 8-way merge process.

The effects of computational resources on the performance of mergesort are analyzed with respect to: cache replacement policies [9]; load distribution on processors [10]; parallel architectures of processors [1], [5], [8], [11]; programming environment like Open MP, MPI and Concurrent Java [12].

Most of these experimental analyses concentrate on the run-time performance of mergesort algorithm. Though [5] finds upper-bound on the worst case run-time of the algorithm, none of the works so discussed have given expression for actual run-time of the algorithm. Moreover, each of these analyses has used specialized configurations which do not always depict the typical computation environment that we use day to day activities. This motivates us for an analysis of the mergesort that establishes the actual run-time model for those computation systems that can be easily accessible and affordable, i.e., performance of mergesort in personal computers.

Rest of the papers is organized as follows: Section III states the objective of the study; section IV gives details of the methodology and concepts used for the analysis and also includes hardware and software the platform, the datasets used and the empirical models used; section V uses the methodologies to arrive at a concluding model, followed by limitations of the used methodology, conclusion and references.

III. Objectives Of The Study

We aim to identify the best curve that can be fitted to the data points obtained by running worst case of mergesort algorithm in personal computer. The objective of this study is to propose a mathematical model using large datasets that can explain the actual run-time behavior of the worst case of mergesort.

IV. Research Methodology

A. Sample Dataset

We have used open source platform and ANSI C to generate the experimental data. The sample data set is given in TABLE 1.

TABLE 1: Dataset for Analysis

Sl. No	Number of elements	Time (Seconds)	Sl. No	Number of elements	Time (Seconds)
1	1000	0	12	55000	0.012
2	5000	0	13	60000	0.012
3	10000	0	14	65000	0.014
4	15000	0	15	70000	0.017
5	20000	0	16	75000	0.018
6	25000	0	17	80000	0.02
7	30000	0.005	18	85000	0.02
8	35000	0.007	19	90000	0.02
9	40000	0.009	20	95000	0.02
10	45000	0.01	21	100000	0.029
11	50000	0.01			

B. Model Fitting

In this study, we have considered 'Time' as the dependent variable and 'Number of elements' as the independent variable for analyzing these dataset. Therefore, the proposed generic mathematical model is of the form:

$$\text{Time} = f(\text{Number of elements}) \quad (1)$$

The sample data points are fitted with nineteen (19) various types of models from different types of fits (standard MATLAB fits) such as Polynomial [13], Exponential [14], Gaussian [15], Power [16] and Fourier [17]. 'Goodness of fit' statistics of all the models are obtained for choosing the best model amongst these models. TABLE 2 shows the types of fit, the model names (from MATLAB) and the model expressions used in the analysis.

C. Fitting Method, Algorithm and Confidence Level

In this paper, we have used 'Linear least squares' [28] as fitting method with 'Robust' [19] off for linear model. For all other models we have used 'Non linear least squares' [18] as fitting method with 'Robust' [19] off. All the curve fittings have been done using the 'Trust – Region' [20] algorithm. We have performed the entire analysis at 95% confidence level.

D. 'Goodness of Fit' Statistics Based Model Selection

We have used R^2, Adjusted R^2, Sum of squares due to error (SSE) and Root mean squared error (RMSE) as 'goodness of fit' statistics for selecting the best model amongst the fitted model. A better fit is indicated by high value of R^2 (close to 1), high

TABLE 2: Models used for Analysis

Type of fit	Model name	Model expression
Polynomial	Linear	$y = \sum_{i=1}^{2} p_i x^{2-i}$
Polynomial	Quadratic	$y = \sum_{i=1}^{3} p_i x^{3-i}$
Polynomial	Cubic	$y = \sum_{i=1}^{4} p_i x^{4-i}$
Polynomial	Polynomial of degree 4	$y = \sum_{i=1}^{5} p_i x^{5-i}$
Power	Power1	$y = ax^b$
Power	Power2	$y = ax^b + c$
Exponential	Exponential1	$y = ae^{bx}$
Exponential	Exponential2	$y = ae^{bx} + ce^{dx}$
Gaussian	Gaussian1	$y = a_1 e^{\left[-\left(\frac{x-b_1}{c_1}\right)^2\right]}$
Gaussian	Gaussian2	$y = \sum_{i=1}^{2} a_i e^{\left[-\left(\frac{x-b_i}{c_i}\right)^2\right]}$
Gaussian	Gaussian3	$y = \sum_{i=1}^{3} a_i e^{\left[-\left(\frac{x-b_i}{c_i}\right)^2\right]}$
Gaussian	Gaussian4	$y = \sum_{i=1}^{4} a_i e^{\left[-\left(\frac{x-b_i}{c_i}\right)^2\right]}$
Gaussian	Gaussian5	$y = \sum_{i=1}^{5} a_i e^{\left[-\left(\frac{x-b_i}{c_i}\right)^2\right]}$
Gaussian	Gaussian6	$y = \sum_{i=1}^{6} a_i e^{\left[-\left(\frac{x-b_i}{c_i}\right)^2\right]}$
Fourier	Fourier1	$y = a_0 + a_1 \cos(wx) + b_1 \sin(wx)$
Fourier	Fourier2	$y = a_0 + \sum_{i=1}^{2} (a_i \cos(iwx) + b_i \sin(iwx))$
Fourier	Fourier3	$y = a_0 + \sum_{i=1}^{3} (a_i \cos(iwx) + b_i \sin(iwx))$
Fourier	Fourier4	$y = a_0 + \sum_{i=1}^{4} (a_i \cos(iwx) + b_i \sin(iwx))$
Fourier	Fourier5	$y = a_0 + \sum_{i=1}^{5} (a_i \cos(iwx) + b_i \sin(iwx))$

value of Adjusted R^2 (close to 1), low value of SSE (close to 0) and low value of RMSE (close to 0) [21]. In this paper we have chosen the best model which has highest R^2, highest Adjusted R^2, lowest SSE and lowest RMSE values.

E. Diagnostic Procedure

Residual is defined as:

Residual = Observed – Predicted

In this paper, we have used the graphical residual analysis to assess the quality of regression since the quality of regression can be assessed by using residual plots [22][23].

In this study Residual vs Predictor plot, Residual plot, Residual lag plot, Histogram of the residuals and Q-Q plot of residuals these five (5) types of residual plots have been employed.

We may suggest that the model fits the data well if the residuals in the Residual vs Predictor plot does not show any systematic structure [24]. Error variance is checked with the help of Residual plot. The variance of the residuals is constant if the Residual plot shows a horizontal band pattern [22]. The Residual lag plot which is a scatter plot of residuals (i) on the y axis and the residuals (i-1) on the x axis is used for checking the independence of the error term [22]. We may suggest that the errors are independent if the points on the Residual lag plot appear to be randomly scattered [25]. Normal distribution of the residuals is tested with Histogram of the residuals and Q-Q plot of the residuals. We may suggest that the residuals are normally distributed if a symmetric bell shaped histogram which is evenly distributed around zero is obtained [26]. Again, if the points on the Q-Q plot which is a probability plot are linear then it suggests that the data follow normal distribution [27].

F. Software Used

Experimental data was generated using Fedora 14 (Kernel Version 2.6) and ANSI C. We have used MATLAB 7.7.0, SPSS 17.0 and MS Excel for data analysis.

G. Hardware Platform

The Mergesort algorithm for worst case has been executed and simulated in a personal computer. The characteristics of the computer are listed in the following table (TABLE 3).

TABLE 3: Hardware Configurations

Processor	Intel Core 2 Duo CPU T6570 @2.10
RAM	3GB DDR-3, frequency – 399.0 MHz
L1 data cache size	32 KB (8-way set associative)
L1 instruction cache size	32 KB (8-way set associative)
L2 cache size	2048 KB (8-way set associative)

V. Data Analysis and Findings

A. Best Model Selection

The 'Goodness of fit' statistics of various fitted models have been tabulated in the following table (TABLE 4 and equation (1)) for identifying the best model amongst all the fitted models based on the decision rule discussed in sub section IV.D.

TABLE 4: Goodness of fit statistics

Fit label	**SSE**	**R^2**	**Adjusted R^2**	**RMSE**
Linear	0.000072	0.952569	0.950072	0.001951
Quadratic #	0.000065	0.957148	0.952387	0.001905
Cubic #	0.000063	0.958649	0.951352	0.001926
Polynomial of degree 4 #	0.000042	0.972364	0.965455	0.001623
Power1	0.000069	0.954693	0.952308	0.001907
Power2	0.000063	0.958838	0.954264	0.001867
Exponential1	0.000158	0.896679	0.891241	0.002880
Exponential2 $	NaN	NaN	NaN	NaN
Gaussian1	0.001292	0.152617	0.058464	0.008473
Gaussian2	0.000843	0.447100	0.262800	0.007497
Gaussian3	0.000809	0.469481	0.115801	0.008211
Gaussian4	0.000521	0.658240	0.240533	0.007610
Gaussian5	0.000042	0.972183	0.907276	0.002659
Gaussian6	0.001125	0.262258	-3.918280	0.019365
Fourier1	0.000063	0.958394	0.951051	0.001932
Fourier2	0.000042	0.972364	0.963152	0.001676
Fourier3	0.000036	0.976646	0.964071	0.001655
Fourier4	0.000007	0.995561	0.991930	0.000784
Fourier5	0.000005	0.997037	0.993416	0.000709

Equation is badly conditioned

$ Fit could not be computed

From the above table (TABLE 4) we have identified *Fourier5* as the best fitted model amongst all the fitted models since the model is having lowest SSE (0.000005), lowest RMSE (0.000709), highest R^2 (0.997037) and highest Adjusted R^2 (0.993416).

B. Diagnostic Procedure

In this sub section the graphical residual analysis of the best model identified from TABLE 4 of sub section V.A has been carried out to assess the quality of regression.

1) Residual vs. Predictor Plot:

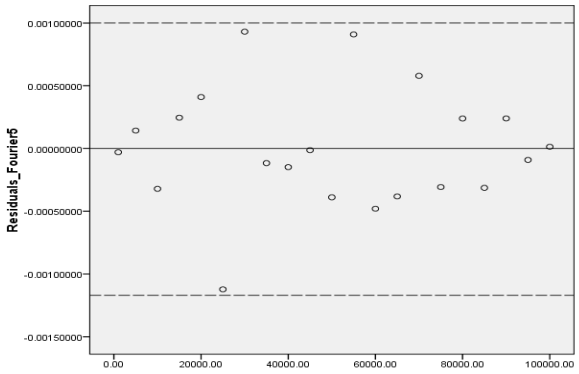

Fig. 1 Residual plot of Fourier5 model

It is evident from Figure 1 that the residuals appear to behave randomly. Therefore, it suggests that the model fits the data well.

2) Residual Plot:

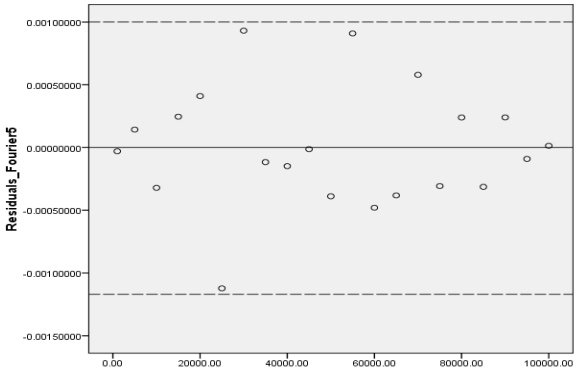

Fig. 2 Residual plot of Fourier5 model

From Figure 2, a horizontal band pattern can be identified which suggests that the variance of the residuals is constant and the regression is a good one.

3) Residual Lag Plot:

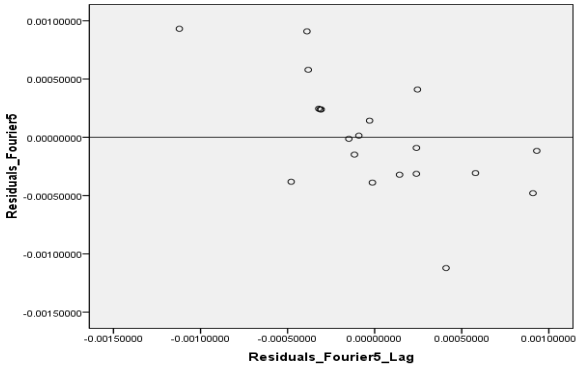

Fig 3. Residual Lag Plot of Fourier5

From Figure 3, it is evident that the points on the plot appear to be randomly scattered which suggests that the errors are independent.

4) Histogram of the residuals:

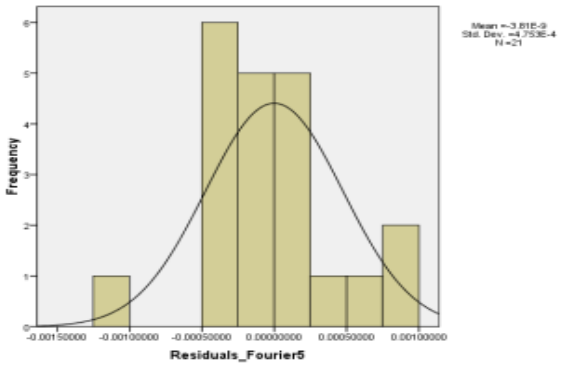

Fig 4. Histogram of the residuals for Fourier5

From Figure 4, we obtain a symmetric bell shaped histogram which is evenly distributed around zero suggesting the residuals are normally distributed.

5) Q-Q Plot of the residuals:

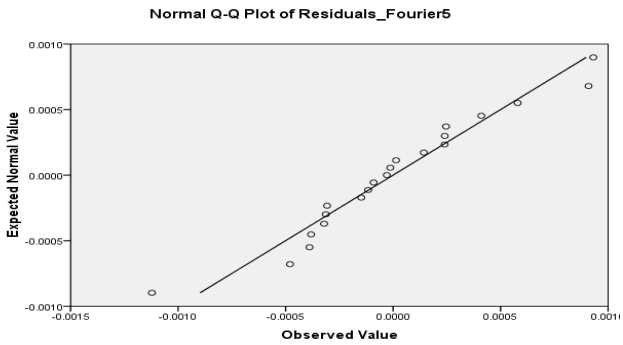

Fig 5. Q-Q Plot of residuals for Fourier5

From Figure 5, we observe that the points on the Q-Q plot are approximately linear. Hence we may suggest that the residuals follow approximately normal distribution.

C. Results of Diagnostic Procedure and Proposed Model

From the graphical residual analysis discussed in sub section V.B we can clearly suggest that the model fits the data well, the variance of the residuals is constant, the errors are independent and the residuals appear to be normally distributed.

Therefore, the proposed generic mathematical model Time ~ f(Number of elements) can be written as:

$$y = a_0 + \sum_{i=1}^{5} (a_i \cos(iwx) + b_i \sin(iwx)) \quad (2)$$

Here, y is Time and x is Number of elements.

The coefficients (with 95% confidence bounds) of the equation 2 are given below:

a_0 = 1.012 (-21.47, 23.5)
a_1 = -0.2699 (-10.72, 10.18)
b_1 = -1.735 (-39.62, 36.15)
a_2 = -1.118 (-23.46, 21.23)
b_2 = 0.3374 (-12.98, 13.66)
a_3 = 0.237 (-8.786, 9.26)
b_3 = 0.5147 (-8.16, 9.189)
a_4 = 0.1562 (-1.776, 2.088)
b_4 = -0.09643 (-3.587, 3.394)
a_5 = -0.01711 (-0.6565, 0.6222)
b_5 = -0.02543 (-0.2131, 0.1623)
w = 0.0000292 (-0.00003183, 0.00009024)

In Figure 6, the plot of the proposed model is shown with the circles indicating the data points, the solid line representing the curve of the proposed model and the dotted line representing 95% prediction bounds.

VI. Limitations and Future Scope

We have performed run-time analysis of worst case of mergesort on a particular hardware and software configuration of a personal computer, where software configuration means particular state of an operating system (Fedora 14). We would like to explore the effects of operating system configurations,

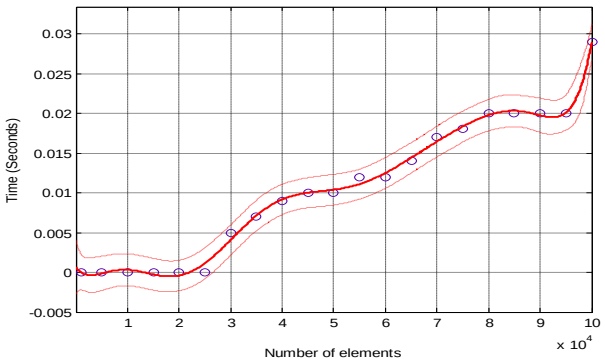

Fig. 6. Proposed model (Fourier5)

i.e., number of processors, cache misses (L1 and L2, separately and in conjunction), dependence on processor architecture, size of RAM, context switch, number of processes, effect of different operating system, etc.

From algorithmic perview, different the variations of mergesort algorithm, i.e., the different variations in phase 1 and phase 2 of the algorithm, and their optimality, are yet to be explored.

The correlation between system configuration (software and hardware) and the phases of the algorithm can be explored in future.

This article has explored the execution time of the mergesort algorithm as an average of many runs and has not taken the variations of the run-time in these runs, which may give insight on the above said correlation.

VII. CONCLUSION

In this paper, we have used curve-fitting technique to examine the trend of experimentally simulated data for the worst case performance of mergesort. After plotting the obtained dataset (<number of elements, time-duration>), we see that the plot follows a pattern. After analyzing the pattern with different curves, namely, polynomial (linear, quadratic, cubic, and bi-quadratic), power (first and second degree), exponential, Gaussian and Fourier (one to five terms), we found that the Fourier series with five terms (Fourier5) fits best amongst these curves, since it has lowest SSE, lowest RMSE, highest R2, and highest adjusted R2; moreover, the residuals appear to be normally distributed having constant variance, and errors are independent. Thus, we can say that the worst case performance of mergesort follows Fourier type fit. Our study shows that the Fourier type of models may be explored for analyzing performance of sorting algorithms and their correlation with the software and hardware configuration of the system in which a sorting algorithm is run.

REFERENCES

[1] Andrew Davidson, David Tarjan, Michael Garland, and John D. Owens, "Efficient Parallel Merge Sort for Fixed and Variable Length Keys", Proceedings of Innovative Parallel Computing, InPar' 12, pp 1-9, May 2012.

[2] Jyrki Katajainen, Tomi Pasanen, and Jukka Teuhola, "Practical In-place Mergesort", Nordic Journal of Computing, Vol. 3.1., pp 27-40, 1996.

[3] S. Todd, "Algorithm and Hardware for a Merge Sort Using Multiple Processors", IBM Journal of Research and Development, Vol 22, no. 5, pp 509-517, 1978.

[4] Enrico Nardelli, Guido Proietti, "Efficient Unbalanced Mergesort", Information Sciences, Vol. 176, Issue 10, pp 1321-1337, May 2006.

[5] Lasse Natvig, "Investigating the Practical Value of Cole's O(log n) Time CREW PRAM Merge Sort Algorithm", Proceedings of The Fifth International Symposium on Computer and Information Sciences (ISCIS V), pp 627-636, October 1990.

[6] Dina Bitton and David J. Dewitt, "Duplicate Record Elimination in Large Data Files", ACM Transactions on Database Systems, Vol.8, no. 2, pp 255-265, 1983.

[7] Jyrki Katajainen and Jesper Larsson Traff, "A Meticulous Analysis of Mergesort Programs", Lecture Notes in Computer Science, Vol. 1203, pp 217-228, 1997.

[8] Jatin Chhugani, William Mary, Aram Baranasi, Anthony D. Ngyuyen, Mostafa Hagog, Sanjeev Kumar, Victor W. Lee, Yen-Kuang Chen, Pradeep Dubey, "Efficient Implementation of Sorting on Multi-core SIMD CPU Architecture", Proceedings of VLDB Endowment, Vol. 1, Issue 2, pp 1313-1324, August 2008.

[9] Richa Gupta and Sanjiv Tokekar, "Proficient Pair of Replacement Algorithms on L1 and L2 Cache for Merge Sort", CoRR, 2010.

[10] Minsoo Jeon and Dongseung Kim, "Parallel Merge Sort with Load Balancing", International Journal on Parallel Programming, Vol. 31, No. 1, pp 21-33, February 2003.

[11] Stephan Wong, Stamatis Vassiliadis and Jae Young Hur, "Parallel Merge sort on a Binary Tree On-chip Network", Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC), pp 365-368, 2005.

[12] M.Rajasekharan Babu, P. Venkata Krishna, Dinesh Kumar, V. Shravan, "Performance Analysis of Odd-even Merge Sort by Using OpenMP, MPI and Concurrent Java", International Journal of Engineering Research and Applications, Vol. 1, Issue 3, pp 1200-1202, October 2011.

[13] Polynomial curve fitting – MATLAB polyfit – Mathworks India, polyfit, [online], http://www.mathworks.in/help/matlab/ref/polyfit.html (Accessed: April, 2014).

[14] Exponential Models – MATLAB & Simulink – Mathworks India, Exponential Models, [online], http://www.mathworks.in/help/curvefit/exponential.html (Accessed: April, 2014).

[15] Gaussian Models – MATLAB & Simulink – Mathworks India, Gaussian Models, [online], http://www.mathworks.in/help/curvefit/gaussian.html (Accessed: April, 2014)

[16] Power Series – MATLAB & Simulink – Mathworks India, Power Series, [online], http://www.mathworks.in/help/curvefit/power.html (Accessed: April, 2014).

[17] Fourier Series – MATLAB & Simulink – Mathworkd India, Fourier Series, [online], http://www.mathworks.in/help/curvefit/fourier.html (Accessed: April, 2014)

[18] Nonlinear Least Squares (Curve Fitting) – MATLAB & Simulink – Math-Works India, MathWorks, [online], http://www.mathworks.in/help/optim/nonlinear-least-squares-curve-fitting.html (Accessed: November, 2013).

[19] fitoptions - MATLAB & Simulink – MathWorks India, MathWorks, [online], http://www.mathworks.in/help/curvefit/fitoptions.html (Accessed: November, 2013).

[20] S.D. Sewell, ―Overview of Matlab Curve Fitting Toolbox‖, 2002. Available from: http://dspace.mit.edu/bitstream/handle/1721.1/36390/8-13Fall-2002/NR/rdonlyres/Physics/8-13Experimental-Physics-I---II--Junior-Lab-Fall2002/031FB55D-EEFC-4B2C-9049-B7C4FD8BA3E0/0/matlabcurvefittingtoolbox.pdf (Accessed: December, 2013).

[21] Evaluating Goodness of Fit – MATLAB & Simulink – MathWorks India, MathWorks, [online], http://www.mathworks.in/help/curvefit/evaluating-goodness-of-fit.html (Accessed: November, 2013).

[22] Graphic Residual Analysis, OriginLab, [online], http://www.originlab.com/www/helponline/origin/en/UserGuide/Graphic_Residual_Analysis.html (Accessed: November, 2013).

[23] D. Das, A. Chakraborty, A. Mitra, "Sample Based Curve Fitting Computation on the Performance of Quicksort in Personal Computer", International Journal of

Scientific and Engineering Research, Volume 5, Issue 2, February 2014.

[24] 4.4.4.1. How can I assess the sufficiency of the functional part of the model?, Engineering Statistics Handbook, [online], http://www.itl.nist.gov/div898/handbook/pmd/section4/pmd441.htm (Accessed: November, 2013).

[25] 4.4.4.4. How can I assess whether the random errors are independent from one to the next?, Engineering Statistics Handbook, [online], http://www.itl.nist.gov/div898/handbook/pmd/section4/pmd444.htm (Accessed: November, 2013).

[26] Normal Probability Plot of Residuals | STAT 501 – Regression Methods, PENNSTATE, [online], https://onlinecourses.science.psu.edu/stat501/node/40 (Accessed: November, 2013).

[27] G. Bandyopadhyay, "Modeling NPA time series data in selected public sector banks in India with semi parametric approach", International Journal of Scientific and Engineering Research, Vol. 4, Issue 12, pp 1876 – 1889, December 2013.

[28] Linear Least Squares – MATLAB & Simulink – Mathworks India, Linear Least Squares, [online], http://www.mathworks.in/help/optim/linear-least-squares.html (Accessed: April, 2014).

[29] Catherine C. McGeoch, A Guide to Experimental Algorithmics, Cambridge University Press, 2012.

[30] D.E. Knuth, Art of Computer Programming 3 (2nd ed.), pp 158-168. Addision-Wesley, 1998.