11-1-2009

Smooth flexible models of nonhomogeneous Poisson processes fit to one or more process realizations

Shalaka C. Deo

Rochester Institute of Technology

SMOOTH FLEXIBLE MODELS OF NONHOMOGENEOUS POISSON PROCESSES FIT TO ONE OR MORE PROCESS REALIZATIONS

A Thesis

Submitted in Partial Fulfillment of the

Master of Science in Industrial Engineering

in the

Department of Industrial & Systems Engineering

Kate Gleason College of Engineering

by

Shalaka C. Deo

B.E., Electronic and Telecommunications, Mumbai University

November, 2009

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

KATE GLEASON COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Shalaka C. Deo

has been examined and approved by the

Thesis Committee as satisfactory for the

thesis requirement for the

Master of Science degree

Approved by:

_____

Dr. Michael E. Kuhl, Thesis Advisor

_____

Dr. Sudit Moises

ABSTRACT

Simulation is a technique of creating representations or models of real world systems or processes and conducting experiments to predict behavior of actual systems. Input modeling is a critical aspect of simulation modeling. Stochastic input models are used to model various aspects of the system under uncertainty including process times and interarrival times. This research focuses on input models for nonstationary arrival processes that can be represented as nonhomogeneous Poisson processes (NHPPs). In particular, a smooth flexible model for the mean-value function (or integrated rate function) of a general NHPP is estimated. To represent the mean-value function, the method utilizes a specially formulated polynomial that is constrained in least-squares estimation to be nondecreasing so the corresponding rate function is nonnegative and continuously differentiable. The degree of the polynomial is determined by applying a modified likelihood ratio test to a set of transformed arrival times resulting from a variance stabilizing transformation of the observed data. Given the degree of polynomial, final estimates of the polynomial coefficients are obtained from original arrival times using least-squares estimation. The method is extended to fit an NHPP model to multiple observed realizations of a process. In addition, the method is adapted to a multiresolution procedure that effectively models NHPPs with long term trend and cyclic behavior given multiple process realizations. An experimental performance evaluation is conducted to determine the capabilities and limitations of the NHPP fitting procedure for single and multiple realizations of test processes. The method is implemented in a Java-based programming environment along with a web interface that allows user to upload observed data, fit an NHPP, and generate realizations of the fitted NHPP for use in simulation experiments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

In a fast-paced dynamic world, modeling a system's behavior is often of prime importance. Important decisions regarding a system can be aided by modeling a system's behavior using simulation. By investigating various scenarios of system implementation, the best course of action can be determined for the system. System modeling is commonly used in manufacturing, service, health care, and defense industries. In the manufacturing industry, models are developed for assembly lines to identify system bottlenecks. Various alternatives for removing these bottlenecks and improving overall system utilization are tested using simulation models prior to their implementation. Implementing these alternatives can be very expensive; hence, simulation modeling plays a vital role. Novice pilots are trained in a simulation environment, avoiding the risk of actual take off. Simulation models answer innumerable questions and provide valuable feedback. Capabilities of simulation models have progressed along with technological advances. With advanced computer-programming languages, simulation of complex systems has been possible. Strategic decisions in various fields of businesses depend on correct prediction of system behavior.

A typical system receives input, processes the input, and delivers output. During system simulation, the quality of results or output depends on the quality of inputs provided to the system. If system inputs are not modeled correctly, a simulation model loses credibility; hence, modeling inputs of a system correctly is vital. This research contributes to the field of simulation in the arena of input modeling. Henderson (2003) defines an *input model* as a collection of distributions together with associated parameters as primitive inputs in a simulation model. Inputs to a system often follow complex patterns varying over time. Available historical data can be converted into important information only when this data is modeled correctly using various

statistical distributions. A considerable proportion of input data patterns are identified as nonstationary processes.

Nonstationary processes are processes, which have statistical parameters such as the mean and variance that are functions of time. Many industrial, biomedical, health care, and financial applications have time-series data demonstrating nonstationary behavior.

Since nonstationary processes encompass a variety of applications with wide spread usage in diverse fields, a number of researchers have developed various methods for modeling these processes using mathematical functions. The difficulty in modeling them lies in their gross non-homogeneity. A number of complex methods and probability models have been developed to simulate nonstationary processes. Often these nonstationary processes are described using Nonhomogeneous Poisson Processes (NHPPs) whose intensity or rate function varies with time. NHPPs are also used for modeling processes exhibiting cyclic effect; for example, customers patronizing a restaurant or store White (1999) or incoming calls to a Customer Service Center during a day Massey et al. (1996). Processes showing long-term trends are also simulated using NHPP Kuhl et al. (1997). Considerable amount of research has been dedicated to this field of statistics with various methods being developed for simulating a NHPP with varying levels of complexity for specific applications. This research develops a method of simulating an NHPP and testing its applicability in various situations. The model fits an NHPP using data from single or multiple realizations of process. This thesis also implements a web interface for the modeling method, which can be accessed through the internet.

1.1 Nonhomogeneous Poisson Processes

In probability, a *nonhomogeneous Poisson process* is a process for which the event arrival (or event occurrence), hereafter referred to as 'arrival rate', is a function of time or varies

with time. A *process* is defined as a nonhomogeneous Poisson process when the number of events in any of the non-overlapping finite set of intervals is represented as independent random variables with the integrated rate function for the process being a monotone nondecreasing right continuous rate function bounded in any finite interval. In following discussion, *N(t)* refers to the total number of arrivals in time interval t.

Nonhomogeneous Poisson process can be defined (Çinlar 1975) as a stochastic process in which for some small value *h*,

- $N(0) = 0;$
- $P\{N(t, t+s) = 0\} = 1 - \lambda(t)s + o(s);$
- $P\{N(t, t+s) = 1\} = \lambda(t)s + o(s);$
- $P\{N(t, t+s) > 1\} = o(s).$

In above definition $\lambda(t)$ refers to instantaneous arrival rate at time instant t. Aggregate number of arrivals till time t is given by integrated rate function or mean-value function $\mu(t)$ where,

$$\mu(t) = \int_0^t \lambda(z)\mathrm{dz} \text{ for all } \lambda(t) > 0.$$

1.2 Applications of NHPPs

A wide variety of nonstationary processes can be modeled as Nonhomogeneous Poisson processes. NHPPs have been used to simulate arrival times of queuing systems and failure times of repairable systems, (Ebeling 1997). In addition to mechanical failures, NHPP models have been employed for predicting software failures, (Wang et al. 2007). Preventive maintenance scheduling has been based on NHPP models developed to simulate failure rates of the system. NHPPs have been used to predict the failures in terms of warranty claims rate for automobile

policies, (Majeske 2007) and for earthquake occurrence, (Vere-Jones 1970). Hardware and software network configuration decisions have been based on the NHPP models for the arrival rates of calls (Lewis and Shedler 1976b).

In health care industries NHPPs have been utilized for modeling rate of patient arrivals (Pritsker, 1998). One such example is the UNOS Liver Allocation Model (ULAM), where NHPP models time series data exhibiting long-term trend and multiple cycles. NHPPs have been used to simulate the arrival rates of customers to determine optimum utilization of available resources (White 1999).

NHPPs have a wide variety of applications; hence, tremendous amount of research is dedicated to developing methods to model NHPPs using its cumulative mean-value function or instantaneous rate function.

2. Problem Statement

A typical NHPP can be completely defined using the instantaneous rate function $\lambda(t)$ or the integrated rate function, also known as *cumulative mean-value function* $\mu(t)$, Çinlar (1975). Estimating the instantaneous or mean-value function tends to be complex and many mathematical models are available in literature. Mathematical models developed by various researchers' posses various capabilities and are applicable to different patterns observed in NHPPs. Complex methods attempting to address various behavioral patterns of NHPPs tend to be computationally complex and can consume a great deal of machine time when automated. This research aims to develop a method to satisfy five main objectives.

The first objective of this research is to develop a smooth, flexible, semiparametric method for modeling NHPPs given one or more process realizations. The method aims to model the continuous rate change of NHPPs by estimating mean-value function of the NHPP. The developed method can be used to model any typical NHPP and has ability to receive arrival times from single or multiple process realizations. The ability to model NHPP from multiple process realizations proves to be beneficial in cases where number of arrivals over single process realization is inadequate for accurate estimation of mean-value function.

The second objective of this method is to automate the developed semi parametric method to determine values of the parameters. Automated process accepts arrival times as inputs and returns output in form of parameter values. For automating the process an algorithm of estimation process is determined and implemented using programming language Java.

The third objective of this research is to adapt the multiresolution method of Kuhl, Sumant, and Wilson (2006) to allow for multiple process realizations. Multiresolution procedure

is applicable to NHPPs exhibiting long term trend and periodic cyclic effects from single process realization.

The fourth objective of this research is to conduct statistical performance tests for developed method for single and multiple process realizations. The goodness-of-fit measures are used to evaluate performance of method in estimating mean-value function. Method is tested for various scenarios to ensure consistency in the quality of outputs.

The last objective of this research is to develop a web-interface which enable users to access the method over internet. Users can upload arrival times as input file, execute the developed method over server and download results.

3. Literature Review

Simulation of a nonhomogeneous Poisson process has been a challenge for several years due to complex patterns and trends observed in ever increasingly complex systems under study. Modeling of NHPPs involves advanced mathematics and statistical principles; hence, a general method that is easily accessible by a wide range of individuals is an area of interest. Over a period of time, many parametric, semiparametric and nonparametric methods have been developed to model the rate and integrated rate functions. Some of these methods that have been developed for particular applications are summarized in the section below. Some of the simulation models have been theoretical while others have been implemented into statistical data fitting software. The importance of simulation models in predicting system behavior has increased in the competitive world. Technological advancements have made it is possible to store and generate large amounts of data regarding a system. This historical data can be used efficiently to develop models enabling organization to study their system and achieve meaningful results. NHPP models used as inputs simulation modeling and analysis have been shown to aid organizations in achieving these meaningful results. This section summarizes methods developed by researchers in the arena of NHPP modeling and simulation.

3.1 Parametric Models for NHPPs

As indicated previously, an NHPP can be modeled by specifying instantaneous rate function, $\lambda(t)$ or the mean-value function, $\mu(t)$. Parametric models have been developed for both the rate and mean-value functions. Different types of rate function are modeled in literature. Lewis and Shedler (1976) use a thinning procedure to simulate NHPPs with complex rate function, showing periodicities. Thinning is an acceptance-rejection technique in which an NHPP with a general rate function $\lambda(x)$ is realized by thinning another NHPP with a rate

function $\lambda^*(x)$, such that $\lambda^*(x) >= \lambda(x)$. Advantages of this method are its simplicity of computation, as there is no need to estimate the integrated rate function. The method is often used in cases where a high amount of data is available. White (1999) uses a thinning procedure to estimate the cumulative rate function of customer arrivals at a reputed electronics store. Historical arrival data is pruned or filtered in an ad-hoc manner with daily and hourly thinning factors computed. The model is specially developed with an aim of deciding a staffing schedule depending on the forecasted arrival rate of customers. Literature shows many NHPP models being developed for a particular company or situation. Depending on commonality of certain situations, various forms of integrated rate function have been developed.

Lewis and Shedler (1976) model a log-linear rate function for an NHPP. The mathematical form of this rate function is:

$$\lambda(t) = \exp(\alpha_0 + \alpha_1 t),$$

where α_0 and α_1 are constants of the function. The log-linear rate is always positive, but the slope can be positive or negative; hence, the model can accommodate increasing or decreasing trends over the specified interval.

The power law rate function is an NHPP model applied to predicting failure rate in systems either improving or deteriorating with time. NHPP models the special patterns of varying failure rates as discussed by Crowder et al. (1991); it is also called the Rate of Occurrence of Failure (ROCOF). Mathematically, the power law rate function and power law intensity function can be represented as:

$$\lambda(t) = vbt^{b-1} (v, b > 0, t \geq 0)$$

and

$$\mu(t) = vt^b.$$

The log-linear and power law rate functions are monotonically increasing functions due to their exponential component. Since these rate functions have only a few parameters, they are relatively easy to estimate. These functions are mostly used in reverse engineering and reliability systems.

Recently, a number of NHPP models have been developed to predict the failure rates for software applications; these are called as *Software Reliability Growth Models* (SRGM) (Wang 2007). Apart from predicting failure rates in industries, NHPP models are also implemented in determining the claim rates for automobile policies. Model developed by Majeske (2007), it is used to predict the warranty claim rate for automobile insurance policies.

To approximate a continuous rate function more closely, Maclean (1974) models an exponential polynomial rate function, which is mathematically defined as follows:

$$\lambda(t) = \exp\left(\sum_{m=0}^{r} \alpha_m t^m \right),$$

such that $-\infty < \alpha_0, \alpha_1, \alpha_2, ..., \alpha_m < +\infty$. A statistical procedure has been developed by Maclean (1974) for estimating the parameters of rate function. First, the degree *r* is estimated for the polynomial then the polynomial coefficients are estimated using the Newton – Raphson Method. Lewis and Shedler (1976) use this rate function to characterize workload in database management systems. To account for the cyclic behavior of nonhomogeneous Poisson process, Johnson et al. (1991) model an Exponential – Polynomial – Trigonometric (EPT) Rate function,

$$\lambda(t) = \exp\{h(m,t)\}$$

with,

$$h(m,t) = \sum_{i=0}^{m} \alpha_i t^i + \gamma \sin(\omega t + \phi),$$

where $[\alpha_0, \alpha_1, ..., \alpha_m, \gamma, \phi, \omega]$ is a vector of unknown parameters. This function is an extension of the Exponential Polynomial Rate function. The trigonometric component is specifically added to take into account the periodic cycles in the NHPP. The arrival rates at hospitals and restaurant facilities may exhibit such cyclic patterns. NHPPs are also used to model arrival rates exhibiting long-term effects and multiple periodicities. The rate function is modeled as an Exponential – Polynomial – Trigonometric Function with Multiple Periodicities (EPTMP) by Kuhl et al. (1997) and defined as

$$\lambda(t) = \exp\{h(\mathrm{t};\mathrm{m},\mathrm{p},\Theta)\}, \qquad t \in [0, S],$$

with

$$h(\mathrm{t};\mathrm{m},\mathrm{p},\Theta) = \sum_{i=0}^{m} \alpha_i t^i + \sum_{k=1}^{p} \gamma_k \sin(\omega_k t + \phi_k),$$

where

$$\Theta = [\alpha_0, \alpha_1, ..., \alpha_m, \gamma_1, ..., \gamma_p, \phi_1, ..., \phi_p, \omega_1, ...\omega_p],$$

is a vector of continuous parameters. A wide range of nonstationary processes can be modeled using this procedure. The model first determines the initial estimates of the parameters. The final parameters of the models are then calculated from these initial estimates using the Newton – Raphson method. To estimate the parameters of the EPTMF Rate function developed by Kuhl et al. (1997) in an efficient manner, Kuhl and Wilson (2000) use the Least Squares Estimate method. The parameters of the mean value function are estimated by minimizing the residuals obtained.

In many situations the data to be modeled using NHPP follows a simple pattern. Massey et al. (1996) form a model for an NHPP with linear rate over subintervals to model arrival rate of call requests at the AT&T long distance network. The model assumes the rate of NHPP is linear

over the appropriate subintervals or piecewise linear. Mathematically, the arrival rate function of NHPP over interval of length T is,

$$\lambda(t) = a + bt, \text{ such that } 0 \le t \le T.$$

Estimators are developed for parameters a, b, number of subintervals and length of subinterval.

Parametric methods use parametric form of mean-value or rate function to define NHPPs. Methods use observed arrival data of NHPPs to estimate parameter values for modeling NHPPs.

3.2 Nonparametric Methods for Estimating the Rate Function

Nonparametric estimators do not assume any form of rate function; hence, can be applied to cases where functional form of intensity is unknown (Wang et al. 2007). Lewis and Shedler (1976) use a nonparametric estimator to model workload in database systems. The estimator defined for the rate function is as follows:

$$\hat{\lambda}(t; n, t_0) = \frac{1}{b(n)} \sum_{j=1}^{n} W\left(\frac{t - T_j}{b(n)}\right),$$

where, t_0 is upper limit of interval; $W(u)$ is a bounded nonnegative integrable weight function with $\int_{-\infty}^{+\infty} W(u) = 1$ and $b(n)$ is a positive bandwidth function, which tends to 0 as $n \to \infty$. Leemis (1991) define a nonparametric method to estimate the cumulative rate function from multiple realizations. The piecewise-linear estimator of cumulative rate function between the time values in superposition is:

$$\hat{\Lambda}(t) = \frac{in}{(n+1)k} + \left[\frac{n(t - t_{(i)})}{(n+1)k(t_{(i+1)} - t_i)}\right]$$

with,

$$t_i \le t \le t_{(i+1)} : i = 0, 1, 2, ..., n.$$

Kuhl and Bhairgond (2000) develop a nonparametric estimation of rate function of nonhomogeneous Poisson process using wavelets. Generally, a wavelet divides a given function into different frequency components then studies each component at a resolution matching its scale. The method is developed to model this wavelet estimator for a typical NHPP. The main advantage of this method is its flexibility as it can be applied to a wide range of nonstationary processes which may or may not show long- term trend or cyclic effects. Moreover it can also be used for modeling processes, where the prior knowledge of the process may not be available.

3.3 Multiresolution Procedure to Model NHPPs

A semiparamteric method for the estimation of the mean value function of an nonhomogeneous Poisson process is the multiresolution procedure developed by Kuhl et al. (1997). This technique can be used for NHPPs exhibiting cyclic behavior or long-term trend. The number of cycles in the NHPP determines the number of resolution. The method is based on two assumptions as stated by Kuhl et al. (1997)

1) There are p distinct cycles in the length of observation interval with the cycle length of lower resolution being an integral multiple of higher resolution cycle length. If b_i then the cycle length of cycle i then:

$$b_1 > b_2 > b_3 ..... > b_p .$$

2) The arrival rate at any time within a particular resolution in any cycle is proportional to a single base line function.

The nonparametric nature of this method pertains to the fact that the mean value function needn't be specified in a parametric form. The procedure needs the basic knowledge of the number of resolutions for a particular process and length of each cycle. This information can be obtained by observing the arrival time data or by spectral analysis. Each periodic

component is referred to by the term 'resolution'. To estimate the mean-value function $\mu(t)$ over entire observation length of S, a function $R_i(s)$ is estimated at each resolution $i = 1,..., p$ for resolution length $s \in [0, b_i)$ and $i = 1,..., p$. The function $R_i(s)$ serves as function modeling data pertaining to specific resolution i, $i = 1,..., p$. The estimator of function $R_i(s)$ is denoted by $\hat{R}_i(s)$ and it is scaled to the unit observation length and cumulative fraction arrivals. The estimator $\hat{R}_i(s)$ is such that $\hat{R}_i(0) = R_i(0) = 0$ and $\hat{R}_i(b_i) = R_i(b_i) = 1$ for $i = 1,..., p$. At the lowest resolution a monotonically increasing function $\hat{R}_0(t)$ is estimated to fit the points, $\left\{\left[jb_1, N(jb_1 / N(S))\right]^T : j = 0,1,..., S / b_1\right\}$, where:

$$G_i(S) = \frac{1}{N(S)} \sum_{l=0}^{(S/b_i)-1} \left[N(l(b_i + s) - N(lb_i)\right]$$

for all $s \in [0, b_i)$. For higher resolutions, $(i = 1, 2,..., p-1)$, a monotonically increasing function $\hat{R}_i(s)$ is constructed to estimate cumulative proportion of arrivals expected to occur during first S time cycles where $s \in [0, b_i)$. The estimator $\hat{R}_i(s)$ is of form,

$$\hat{R}_i(s) \equiv \begin{cases} s / b_i, & \text{if } r = 1 \\ \sum_{k=1}^{r-1} \beta_k (s / b_i)^k + \left(1 - \sum_{k=1}^{r-1} \beta_k\right)(s / b_i)^r, & \text{if r >1.} \end{cases}$$

The $\hat{R}_i(s)$ is a polynomial function and vector coefficients $\{\beta_k : k = 1,..k-1\}$ are constrained so that derivative of estimator, $\hat{R}_i'(s) > 0$ for all $s \in [0, b_i)$. The polynomial function proves to be advantageous because of its inherent capability of modeling the constant arrival rate using a linear mean-value function and a complex arrival rate using higher degrees of polynomial. The degree of polynomial function, r, corresponds to the complexity of data

pattern. Estimation of the degree r is automated by Kuhl et al. (2006). The degree r of the polynomial is determined by likelihood ratio test that been adapted to constrained non linear regression and is applied to transformed arrival data. The algorithm for the likelihood ratio test to determine the polynomial degree r is shown in the figure 3.1 as devised by Kuhl et al. (2006) for reference. The estimators $\hat{R}_i(s)$ at each resolution are used for mean-value function estimator $\hat{\mu}_t$ defined for observation length S and cumulative observed arrivals $N(S)$ is computed as,

$$\hat{\mu}_t = N(S)\hat{Q}_o(t) \text{ for } t \in [0, S] ,$$

where function $\hat{Q}_i(t) : i = p, p-1, ..., 1, 0\}$ are defined iteratively by,

$$\hat{Q}_p(t) = \hat{R}_p(t - (j_{p,t} - 1)b_p),$$

and for $i = p-1, ...1, 0$

$$\hat{Q}_i(t) = \hat{R}_i\left(\left(j_{i+1,t} - 1\right)b_{i+1} - \left(j_{i,t} - 1\right)b_i\right)$$
$$+\left[\hat{R}_i\left(j_{i+1,t}, b_{i+1} - \left(j_{i,t} - 1\right)b_i\right) - \hat{R}_i\left(j_{i+1,t} - 1\right)b_{i+1} - \left(j_{i,t} - 1\right)b_i\right]\hat{Q}_{i+1}(t)$$

where the interval containing t is within each resolution index given by index. In the term, $j_{i,t}$ is the unique integer j such that $(j-1)b_i \le t \le jb_i$.

LRT-Based Multi-Resolution Estimation Procedure

[0] Initialize the resolution index $i \leftarrow 0$.

[1] If $i > p$ then deliver the fitted mean-value function $\hat{\mu}(t) = \hat{\mu}(t; \tilde{\beta}_0, ..., \tilde{\beta}_p)$ using the estimated regression coefficients $\{\tilde{\beta}_0, ..., \tilde{\beta}_p\}$ then stopping. If $i \le p$, then take the composite of the arc- sine and square-root transformations of the original arrival data within the basic resolution-i cycle $[0, b_i]$.

[2] Test the adequacy of the degree-1 polynomial $\hat{R}_i(\bullet)$ as an estimator of $R_i(\bullet)$.

(*a*) If $SSE_1/SST < 0.01$ or $m_i < 2$, then set $r \leftarrow 1$ and $\tilde{\beta}_i \leftarrow 1$ for the original (untransformed) responses at resolution i; set $i \leftarrow i+1$; and go to [1].

(*b*) If $SSE_1/SST \ge 0.01$ and $m_i \ge 2$, then set $r \leftarrow 2$ and go to [3].

[3] Compute the OLS estimator $\tilde{C}_r$ of the coefficient vector C_r for the transformed responses at resolution i as:

(a) Use starting value $\hat{C}_r = \left[\tilde{C}_{r-1}, 0\right]^T$ to obtain $\tilde{C}_r$; for example, to compute $\tilde{C}_2$, start with $\hat{C}_2 = \left[\tilde{C}_1, 0\right]^T = [\pi/2, 0]^T$.

(b) Compute $\tilde{C}_r$ using least square estimates

(c) Compute maximum likelihood estimate for $\tilde{\sigma}_r^2 = SSE_r / m_i$

[4] If $-m_i \ln\left(\sigma_r^2 / \sigma_{r-1}^2\right) \le \chi_{1-\alpha}^2$ then set $r \leftarrow r-1$ and go to [5]; otherwise set, $r \leftarrow r+1$ and go to [3].

[5] Using the fitted degree $\tilde{r}$ compute the OLS fit to for the original (untransformed) data; and saving the resulting $\tilde{r}$-dimensional estimator $\tilde{\beta}_i$ of the vector of regression coefficients in estimator $R_i(\bullet)$. Set $i \leftarrow i+1$ and go to [1].

Figure 3.1 LRT Based Estimation Procedure for Mean-Value Function

3.4 Summary and Discussion

The literature review discusses various parametric, semi-parametric and nonparametric methods developed by the researchers. Table 3.1 summarizes the capabilities of various methods relevant to this research.

Table 3.1 Summary of Literature Review

Method Capabilities	Deo (2009)	Leemis (1991)	Mac Lean (1997)	Kuhl et al. (1997)	Kuhl et al. (2006)
Model Continuous Rate Change	✓		✓	✓	✓
Multiple Cycles and long-term trend in arrival rate	✓			✓	✓
Nonstationary noncyclic arrival rate	✓	✓	✓		
Designed to model from multiple process realizations	✓	✓		✓	
Automated Fitting Procedure	✓				✓

Leemis (1991) develops a nonparametric method to estimate the arrival rate from multiple process realizations while MacLean (1997) devises a parametric method to model continuous arrival rate of NHPP. The parametric method by Kuhl et al. (1997) model NHPPs with periodic cycles and long term trend. Kuhl et al. (2006) formulate an automated semiparamteric method to model mean-value function of NHPPs exhibiting periodic cycles and long term trend. The methods mentioned in table 3.1 possess different capabilities and model NHPPs with certain patterns of arrival rate. Application of these methods depends on the

availability of data regarding arrival times of NHPPs; for example Leemis's (1991) method is generally applicable to the NHPPs where arrivals are recorded over multiple process realizations. Motivation of this research is to develop a method which combines abilities of different methods. The aim of this research is to devise a semiparamteric method which can model continuous rate of NHPPs exhibiting periodic cycles and long term trend and of NHPPs with noncyclic nonstationary arrival rate from one or more process realizations. Finally, this research aims to automate the developed method to estimate the parameter values.

4. Methodology

This research aims to develop an automated procedure to model a general nonhomogeneous Poisson process using a smooth, flexible mean-value function. The mean-value function is estimated from a single or multiple process realizations of the target NHPP over the observation interval $(0, S]$. The mean-value function is modeled as a polynomial function of special form estimated using least square estimates following the procedure established by Kuhl et al. (2006). The method proposes a semi-parametric model of form,

$$\mu(t) = \mu(S)R(t) \quad \text{for all } t \in (0, S] \qquad (1)$$

where $R(t)$ is a monotonically nondecreasing function representing the cumulative proportion of arrivals up to time t. As discussed in Section 3, for the multiresolution procedure, a polynomial function of form:

$$\hat{R}_i(s) = \begin{cases} s/b_i & \text{if } r=1, \\ \sum_{k=1}^{r-1} \beta_k (s/b_i)^k + \left(1 - \sum_{k=1}^{r-1} \beta_k\right)(s/b_i)^k & \text{if } r>1, \end{cases} \qquad (2)$$

for $s \in (0, b_i]$ is utilized at each resolution i for $i = 0,1,2,...,p$ have a similar form of a polynomial function $R(t)$ is fitted to arrival times observed over interval $(0, S]$. The Polynomial function is of form:

$$R(t) = \begin{cases} t/S & \text{if } r=1, \\ \sum_{k=1}^{r-1} \beta_k (t/S)^k + \left(1 - \sum_{k=1}^{r-1} \beta_k\right)(t/S)^r & \text{if } r>1. \end{cases} \qquad (3)$$

The Coefficient vector $\{B_r : \beta_1, \ldots, \beta_{r-1}\}$ is constrained to yield $R'(t) \geq 0$ for all $t \in (0, S]$. The coefficient vector B_r,

$$\mathrm{B} \equiv \begin{cases} 1 & \text{if r =1,} \\ [\beta_1, \ldots, \beta_{r-1}]^{\mathrm{T}}, & \text{if r > 1,} \end{cases}$$

is constrained to yield $\hat{R}(t) > 0$ for all $t \in (0, S]$. The equation (2) ensures the initial value of equation is always zero $R(0) = 0$ and the final value of the equation equals to unity $R(S) = 1$ for all values of B_r. The derivative of the auxiliary function $R'(t) \geq 0$ is always non-negative for all $t \in (0, S]$; hence, the integrated rate function is always a nondecreasing function ensuring that the arrival rate will never be negative.

In principle, a uniformly accurate approximation to the function $R(t)$ can always be achieved using a polynomial of sufficiently high degree, r. The polynomial form of mean-value function provides flexibility of modeling very simple and complex arrival processes.

4.1 Estimation of R(t)

This section develops a method for fitting a polynomial function of form (3) to a single observed realization of an NHPP with $N(S)$ observed arrivals over observation length $t \in (0, S]$. The number of arrivals $N(S)$ over observation length S consists of arrivals times $t_1, t_2 ...., t_{N(S)}$ sorted in ascending order so that $t_{(1)} \le t_{(2)} \le ....t_{(N(S))}$. The polynomial function $R(t)$ is estimated to fit the cumulative fraction of arrivals over the observation interval. We let denote $W_i = i / N(S)$ as cumulative proportion of arrivals up to time $t_{(i)}$ of i^{th} arrival for $i = 1, 2, ..., N(S)$. We seek to fit the polynomial function $R(t)$ to the points $[t_{(i)}, W_i]$ for $i = 1, 2, ..., N(S)$ in following steps:

- Transform the data using a variance-stabilizing transformation;
- Using the transformed data, estimate the degree *r* of the best-fitting polynomial using a modified likelihood ratio test;
- Given the degree r estimate the polynomial estimate coefficient vector by applying least B_r squares estimates to the original data.

The following discussion explains the details of the fitting method.

4.1.1 Transformation of Arrival Times

We seek to fit a polynomial function of degree r to a data set of cumulative arrivals $[t_{(i)}, W_i]$ where $t_{(i)}$ are the ordered arrival times in observation interval length and W_i is the corresponding cumulative fraction of arrivals at $t_{(i)}$. For convenience, we scale arrivals times $t_{(i)}$ on unit interval such that $Z_i = t_{(i)} / S$ for $i = 1, ..., m$, where $m = N(S)$. The data set for the arrival points along with the corresponding cumulative arrivals are represented by:

$$\{(Z_j, W_j)^T : j = 1, ..., N(S)\}$$

where $Z_1, ..., Z_{N(S)}$ represent the scaled arrival times in ascending order. Regression analysis is often performed to estimate relationship of responses to the regressor. Methods such as forward selection and backward elimination are applied when a polynomial function model is estimated to fit responses (Montgomery, 2006). One of the implicit assumptions of regression analysis is that the responses are independent and normally distributed with a constant variance. This assumption is violated for the observed NHPP arrival dataset. Observations $\{W_i\}$ are neither normally distributed nor have constant variance. Therefore, variance stabilization is performed over data set using a data transformation technique. In particular, an arc-sine transformation (Box, Hunter and Hunter 1978) is used to obtain a data set with homogeneous variance such that

$$Y_i = \sin^{-1}\left[\sqrt{W_i}\right] \qquad (4)$$

for $i = 1, 2, ..., N(S)$. Equation (3) yields to responses that are approximately normal with constant variance σ^2.

4.1.2 Likelihood Ratio Test to Determine Polynomial Degree

The least square estimation method is applied to determine the polynomial degree *r* fit to the transformed data. We use method of constrained least square to model transformed data set. A statistical model of form

$$\mathrm{E}[Y_i] = f_r(Z_i; \mathrm{C}_r) + \varepsilon_i \qquad (5)$$

is fit to transformed data such that,

$$f(u; \mathrm{C}_r) = \begin{cases} \left(\frac{\pi}{2}\right)u, & \text{if } r = 1 \\ \sum_{k=1}^{r-1} C_k u^k + \left(\frac{\pi}{2} - \sum_{k=1}^{r-1} C_k\right)u^r, & \text{if } r > 1, \end{cases}$$

for $u \in [0,1]$, where C_r is the coefficient vector

$$\mathrm{C}_r \equiv \begin{cases} \pi/2, & \text{if } r = 1 \\ [\mathrm{C}_1, ..., \mathrm{C}_{r-1}] & \text{if } r > 1, \end{cases}$$

subject to the constraint $f_r^{'}(u; \mathrm{C}_r) \geq 0$ for all $u \in [0,1]$. Note that for $r \geq 2$, the nonnegativity constraint is equivalent to requiring the zeroes of the degree $(r-1)$ polynomial $f_r^{'}(u; \mathrm{C}_r)$ lies outside the interval $(0,1)$.To determine the appropriate degree for the statistical model (5), a modified likelihood ratio test is used. For successive values of r, the vector C_r is estimated via constrained least squares, yielding:

$$\tilde{\mathrm{C}}_r = \arg\min \sum_{i=1}^{m} \left[Y_i - f_r(Z; \hat{\mathrm{C}}_r \right]^2$$

for $r \geq 2$. The corresponding error sum of squares for the degree – r fit is:

$$\mathrm{SSE}_r = \sum_{i=1}^{m} [Y_i - f_r(Z; \hat{\mathrm{C}}_r)]^2$$

for $r \geq 2$; and for $r = 1$, we take:

$$\mathrm{SSE}_1 = \sum_{i=1}^{m} \left[Y_i - \frac{\pi i}{2(m+1)} \right]^2.$$

The corresponding total sum of squares is:

$$\mathrm{SST} = \sum_{i=1}^{m} (Y_i - \bar{Y})^2,$$

where

$$\bar{Y} = m^{-1} \sum_{i=1}^{m} Y_i,$$

is used in the first step of the modified likelihood-ratio procedure is appropriate.

The response variance σ^2 for each postulated value of r,

$$\tilde{\sigma}^2 = \mathrm{SSE}_r / m \text{ for } r = 1, 2, ...$$

The associated likelihood function is the joint distribution of observations Y_i and has the form:

$$L_r(\tilde{\mathrm{C}}_r; \mathrm{Y}) = \prod_{i=1}^{m} \frac{1}{\tilde{\sigma}_r \sqrt{2\pi}} \exp\left\{ -\frac{\left[Y_i - f_r(Z_i; \tilde{\mathrm{C}}_r \right]^2}{2\tilde{\sigma}_r^2} \right\}$$

$$= (2\pi)^{-m/2} (\tilde{\sigma}_r^2)^{-m/2} \exp\left\{ \frac{-\mathrm{SSE}_r}{2\tilde{\sigma}_r^2} \right\}$$

$$= (2\pi e \tilde{\sigma}_r^2)^{-m/2};$$

and the resulting log-likelihood function for degree r is

$$\psi_r(\tilde{\mathrm{C}}_r; Y) = -\frac{m}{2}[\ln(2\pi) + 1 + \ln(\tilde{\sigma}_r^2)].$$

The degree r is determined using the following likelihood ratio test at the level of significance α where $0 < \alpha < 1$. The estimate of r is:

$$\tilde{r} = \begin{cases} 1, & \text{if } \mathrm{SSE}_1 / \mathrm{SST} < 0.01 \text{ or } m \le 2, \\ \min\left\{ r : r \ge 2; -m \ln\left(\frac{\tilde{\sigma}_r^2}{\tilde{\sigma}_{r-1}^2} \right) \le \chi_{1-\alpha}^2(1) \right\} - 1, & \text{otherwise,} \end{cases}$$

where $\chi_{1-\alpha}^2$ (1) denotes $1-\alpha$ quantile of the chi-squared distribution with 1 degree of freedom. The Ordinary Least Squares (OLS) procedure is implemented (Kuhl et al. 1997) to estimate coefficients of polynomial vector. In the procedure the fit for degree $r = 2$, does not show significant improvement over degree, $r = 1$, on occasion the fit $r = 1$ and $r = 2$ are both poor and the decision resulting from (6) is $r = 1$. To avoid such underfitting, LRT is not considered for degree $r = 1$. R^2 test is used to decide validity of fit at $r = 1$ (Walpole et al. 1998). For data set $[X_j, W_j]$ of m data points such that $j = 1, ..., m$, assuming that the regression line is estimated , then the total corrected sum of squares of W is given by:

$$SST = SSR + SSE$$

An alternative formulation for sum of square errors is given by;

$$\sum_{j=1}^{m_i}\left(W_j-\bar{W}\right)^2=\sum_{j=1}^{m_i}\left(\hat{W}_j-\bar{W}\right)^2+\sum_{j=1}^{m_i}\left(W_j-\hat{W}_j\right)^2,$$

where $\bar{W}$ is the mean $W_j(j:1,....,m_i)$ and $\hat{W}_j(j:1,....,m_i)$ are fitted data points using degree $r=1$. Value of R^2 is computed as:

$$R^2=\left(SSR/SST\right)*100\,.$$

If $R^2>99$ then a fit of degree $r=1$ is accepted for the dataset. The fitting process is outlined completely in algorithm shown in the figure 4.1.

The underfitting phenomenon is also observed for NHPPs with inadequate data points. The value of LRT statistic for fitted degree $r=1$ and $r=2$ is very small and hence improvement of quadratic fit over the linear fit seems to be insignificant. To avoid this problem, the existing procedure is modified so that if:

$$-m\ln\left(\frac{\tilde{\sigma}_2^2}{\tilde{\sigma}_1^2}\right)\le\chi_{1-\alpha}^2\,,$$

coefficient vector $\tilde{C}_3$ is estimated with a maximum of likelihood estimates for variance being calculated. The test is repeated at to check:

$$-m\ln\left(\frac{\tilde{\sigma}_3^2}{\tilde{\sigma}_2^2}\right)\le\chi_{1-\alpha}^2\,.$$

A linear fit is estimated only if both the conditions are satisfied. If at degree, $r=3$, it is found that the fitting improves with additional coefficient then successively higher order fits are estimated until the further addition of coefficients does not improve the quality of fit per equation (6). The basis for this modification is explained in the appendix E where the

underfitting is seen for the test sets with a small number of arrivals. The modified algorithm of the estimation procedure is shown in Figure 4.2.

4.1.3 Polynomial Coefficients for Original Data

Given the results of likelihood ratio test algorithm described in Section 4.1.2 a degree $\tilde{r}$ polynomial is fit to points $[t_{(i)}, W_i]\ i = 1, 2, ..., m$. The polynomial vector $\mathrm{B}_{\tilde{r}}$ is estimated by minimizing the function $R(t; r, \mathrm{B}_{\tilde{r}})$ by applying constrained least squares to original data (untransformed data), yielding

$$\tilde{\mathrm{B}}_{\tilde{r}} = \underset{\hat{\mathrm{B}}_{\tilde{r}}: R'(u;\tilde{r},\hat{\mathrm{B}}_{\tilde{r}})\geq 0}{\arg\min} \sum_{i=1}^{m} \left[W_i - R(t_{(i)}; \tilde{r}, \hat{\mathrm{B}}_{\tilde{r}}) \right]^2$$

provided $r \geq 2$; and if $\tilde{r} = 1$ then we take $\tilde{\mathrm{B}}_1 = \tilde{\beta}_1 = 1$ in equation (2) resulting in a linear mean-value function . The final estimator for mean value function (1) is:

$$\tilde{\mu}(t) = N(S) R(t; \tilde{r}, \tilde{\mathrm{B}}_{\tilde{r}}) \text{ for all } t \in (0, S].$$

The rate function of NHPP is estimated from the mean-value function as $\tilde{\lambda}(t) = \tilde{\mu}'(t)$ for all $t \in (0, S]$.

Modified LRT-Based Estimation Procedure

[1] Test the adequacy of the degree-1 polynomial $\hat{R}(t)$ as an estimator of $R(t)$.

(a) If $\text{SSE}_1/\text{SST} < 0.01$ or $m < 2$, then set $r \leftarrow 1$ and $\tilde{\beta} \leftarrow 1$ for the original (untransformed) responses at resolution and stop execution.

(b) If $\text{SSE}_1/\text{SST} \geq 0.01$ and $m \geq 2$, then set $r \leftarrow 2$ and go to [2].

[2] Compute the OLS estimator $\tilde{\text{C}}_r$ of the coefficient vector C_r for the transformed responses at resolution i

(a) Use starting value $\hat{\text{C}}_r = \left[\tilde{\text{C}}_{r-1}, 0\right]^{\text{T}}$ to obtain $\tilde{\text{C}}_r$; for example, to compute $\tilde{\text{C}}_2$, start with $\hat{\text{C}}_2 = \left[\tilde{\text{C}}_1, 0\right]^{\text{T}} = \left[\pi/2, 0\right]^{\text{T}}$.

(b) Compute $\tilde{\text{C}}_r$ using least square estimates

(c) Compute maximum likelihood estimate for $\tilde{\sigma}^2_{r-1} = \text{SSE}_{r-1}/m$ and $\tilde{\sigma}^2_r = \text{SSE}_r/m$

[3] If $-m\ln\left(\dfrac{\sigma_2^2}{\sigma_1^2}\right) \leq \chi^2_{1-\alpha}$ then

(3a) and $-m\ln\left(\dfrac{\sigma_3^2}{\sigma_2^2}\right) \leq \chi^2_{1-\alpha}$ then deliver liner fit $r = 1$.

(3b) and $-m\ln\left(\dfrac{\sigma_3^2}{\sigma_2^2}\right) > \chi^2_{1-\alpha}$, then reject linear fit and continue to next r.

Figure 4.1 Modified Estimation Procedure

4.1.4 Illustrative Example of Single Realization

Illustrative example of fitting a Polynomial function to model Mean Value function of NHPP is discussed to gain an insight for various steps in fitting procedure discussed in Section 4.1.1 through 4.1.3. Model accepts arrivals from a single nonhomogeneous Poisson process over observation duration $S = 4$. Approximately, $N(S) = 2000$ arrivals are observed over interval S. The arrival times are generated by the NHPP Generation method explained in Section 4.3 and Polynomial used for NHPP creation is:

$$R(s) = 0.892620125\, s - 1.190605473\, s^2 + 0.783775048\, s^3 - 0.215066796\, s^4 + 0.020873734\, s^5 .$$

To visualize arrival rate of NHPP, a histogram is plotted with interval bands of 0.1 units in Figure 4.3. The histogram depicts overall pattern of continuous change in arrival rate.

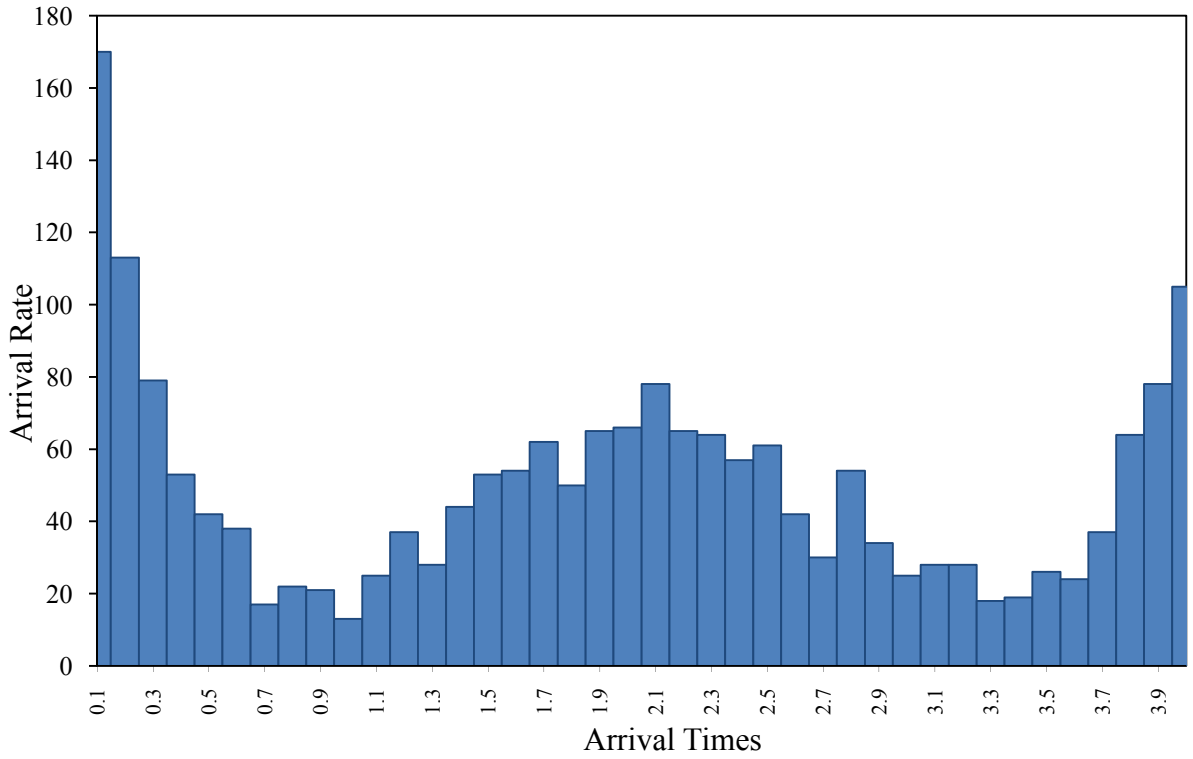

Figure 4.2 Histogram of Arrival Data t ∈ [0,4]

The first step of estimation is to transform arrival set $[t_i, W_i]$ using variance stabilization techniques to determine degree r which can fit transformed data. The arc sine conversion is utilized to obtain transformed dataset $[Z_i, Y_i]$ as discussed in algorithm in Figure 4.1. The likelihood ratio test is employed to estimate degree r of polynomial fitting transformed cumulative arrivals scaled over unit axis. Degree $r = 5$ is fitted to transformed data set by executing steps detailed in Algorithm 4.1 and Estimated Transformed Vector is,

$$\Gamma(s) = 8.96360816\, s\text{-}50.1049823\, s^2 + 126.1871233\, s^3\text{-}137.631791\, s^4 + 54.1568379\, s^5$$

for $t \in (0,1)$. Figure 4.2 plots transformed data and fitted transformed function. The coefficient vector for transformed data is then rescaled to fit for transformed cumulative arrivals over observation duration $S = 4$ such that:

$$\Gamma(s) = 2.24090203\, s\text{-}3.13156139\, s^2 + 1.9716738\, s^3\text{-}0.53762418\, s^4 + 0.05288754\, s^5,$$

for $t \in (0,4]$. Estimation of degree r for transformed data is followed by fitting a polynomial of equal degree to original data. Polynomial coefficients minimizing the sum of square errors are estimated for original data as:

$$R(s) = 0.90926476\, s\ \text{-}1.2125674\, s^2\ \text{+}0.8001907\, s^3\text{-}0.21965397\, s^4\ \text{+}0.02127268\, s^5.$$

Figure 4.3 compares the estimated mean-value function with the step function plotted for original data. The graph demonstrates the ability of method in modeling the integrated rate function using a degree r polynomial. The arrival rate is estimated for NHPP as derivative of the mean-value function. The rate function is computed for all arrival times in $t \in (0,4]$ as:

$$R(s) = 0.90926476\ \text{-}2.4251348\, s\ \text{+}2.4005721\, s^2\text{-}0.87861588\, s^3\ \text{+}0.1063634\, s^4.$$

Figure 4.4 represents the instantaneous rate plotted for arrival times in $t \in (0,4]$ which can be compared with histogram plotted for original data for similarities in actual and estimated rate function. A fitting example visually demonstrates goodness of fitting method in estimation. An

experimental performance evaluation is conducted in Section 5 to establish statistical performance of modeling method.

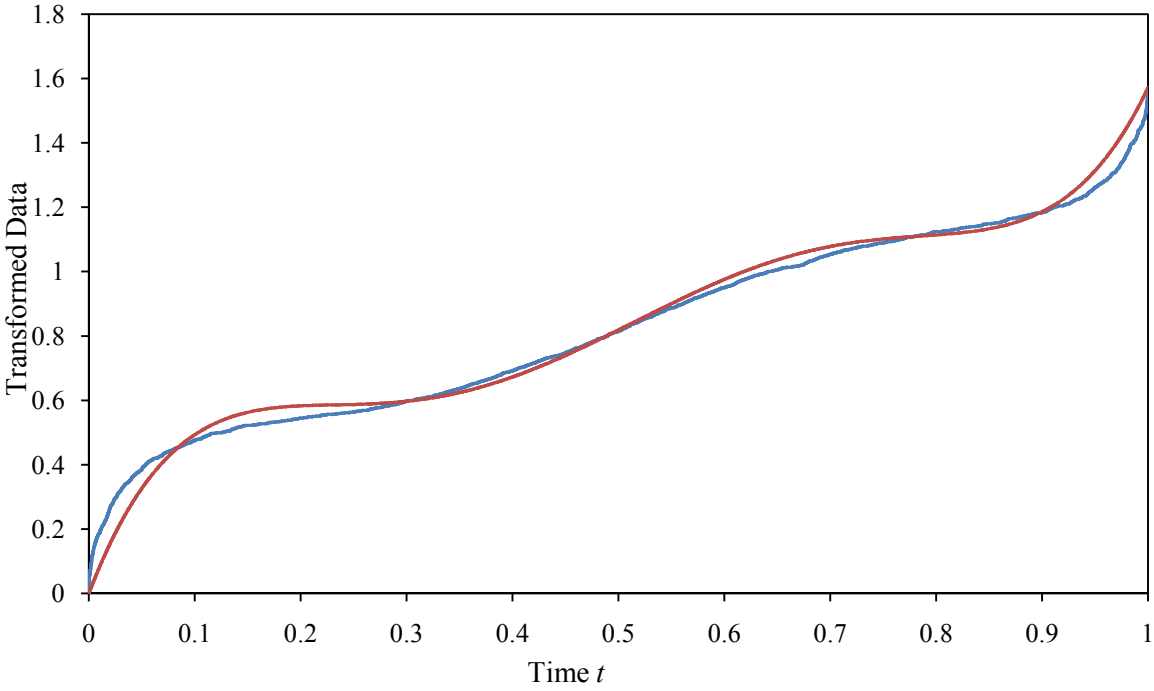

Figure 4.3 Estimated Function for Transformed Data t ∈ [0,1]

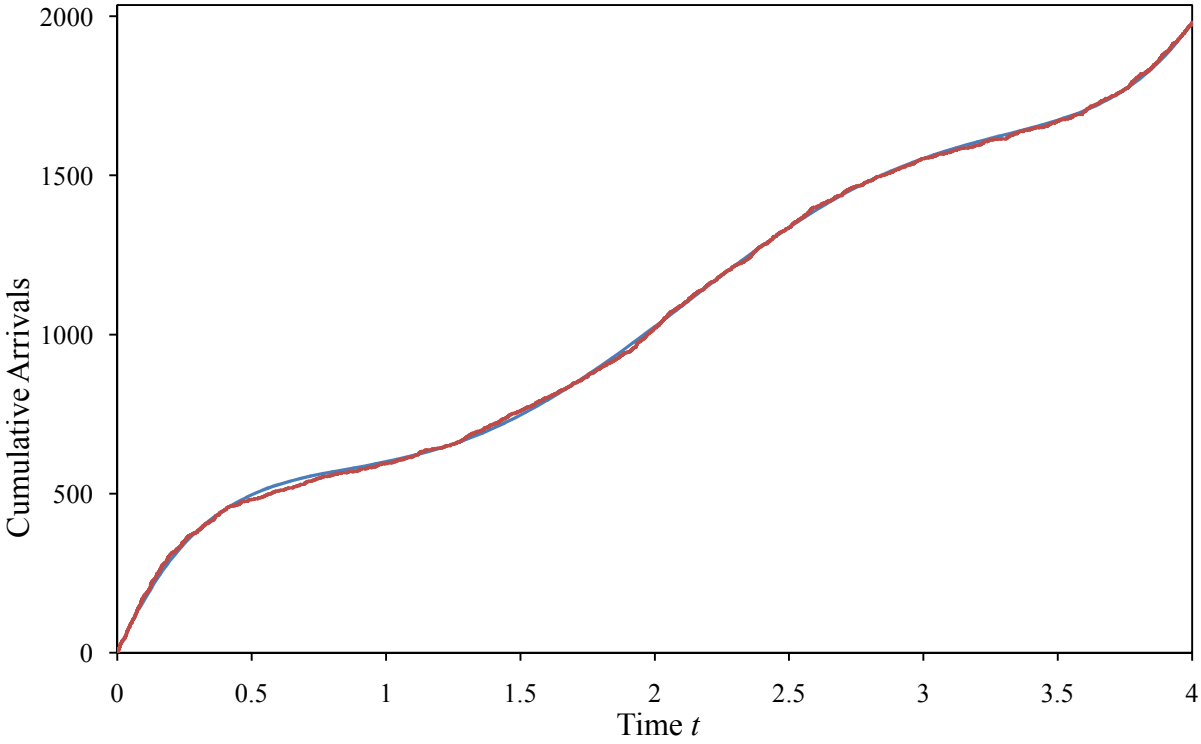

Figure 4.4 Estimated Mean Value Function for Original Data t ∈ [0,4]

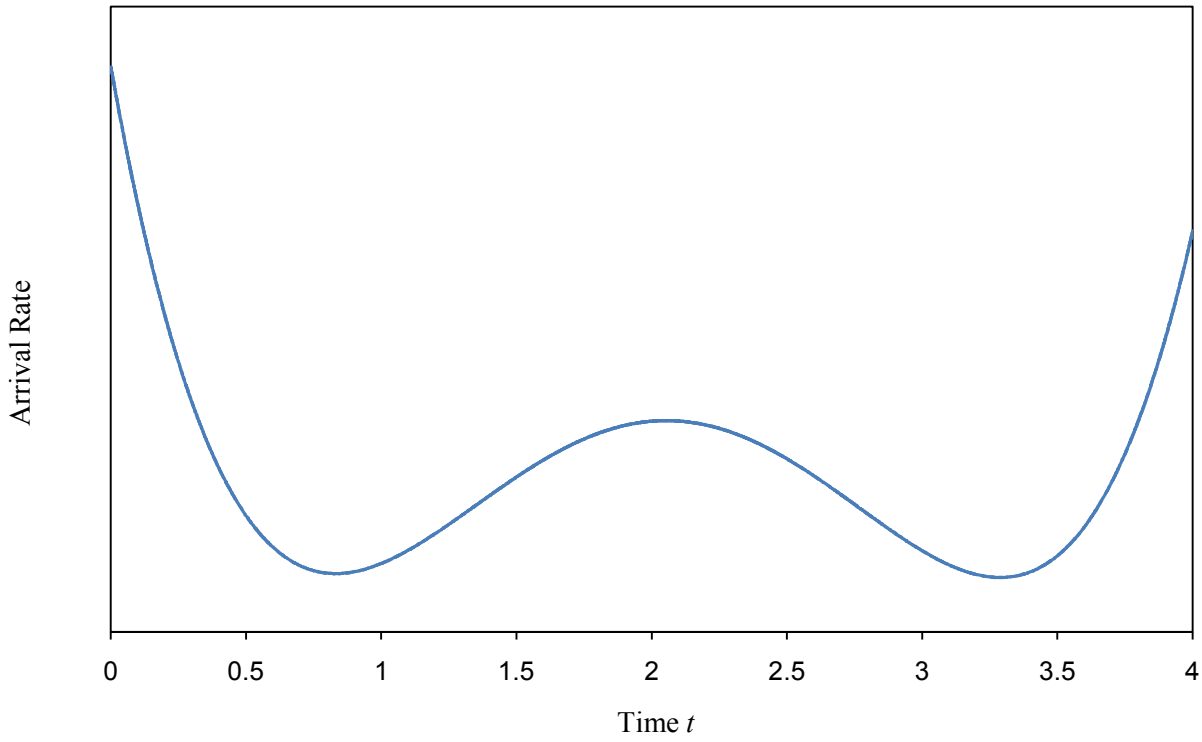

Figure 4.5 Estimated Rate Function for t ∈ [0,4]

4.2 Estimation of NHPPs Given Multiple Process Realizations

When modeling NHPPs in some situations there exists an opportunity to collect multiple realizations of the observed process. In these situations, Leemis (1991) demonstrates that the arrival times from multiple process realizations can be superimposed to fit the mean-value function of an NHPP. Here, this result is utilized to fit the semiparamteric mean-value function to multiple observed process realizations.

4.2.1 Methodology

In this section, the semiparametric model (1) is fit to arrival data originating from multiple process realizations. Suppose there are P realizations of NHPP available and there are

total $N_\ell(S)$ arrivals in ℓ^{th} realization $(\ell = 1,...,P)$ resulting in arrival times $\{t_{i,\ell} : i = 1,..., N_\ell(S)\}$. The estimate of the mean number of arrivals in $(0, S]$, $\mu(S)$ is given by,

$$\bar{N}(S) = \frac{1}{P}\sum_{\ell=1}^{P} N_\ell(S).$$

To fit the constrained polynomial, a consolidated data set is formed by superimposing the arrivals times $\{t_{i,\ell} : i = 1,...N_\ell(S);\ \ell = 1,...,P\}$ over all realizations. This data set is then sorted in ascending order so, $t_{(1)} \le t_{(2)} \le ... \le t_{(m)}$ where:

$$m = \sum_{\ell=1}^{P} N_\ell(S).$$

Let, $W_i = i/m$ denote the cumulative fraction of overall set of arrivals occurring up to time of $t_{(i)}$ of the i^{th} earliest arrival where $i = 1,...,m$. The data set of super imposed arrivals with corresponding cumulative fraction of arrivals is represented by

$$\{(Z_j, W_j)^{\mathrm{T}} : j = 1,...,m\}$$

where $Z_1,...,Z_m$ represent scaled arrival times in ascending order. To obtain normally distributed observation with constant variance, variance stabilization is performed over data set using a data transformation technique. Transformed data set of super imposed arrivals is of form

$$Y_i = \sin^{-1}\left[\sqrt{W_i}\right]$$

for $i = 1,...m$. The least square estimation method is applied to determine the polynomial degree r fit to the transformed data. A statistical model of form

$$\mathrm{E}[Y_i] = f_r(Z_i; \mathrm{C}_r) + \varepsilon_i$$

is fit to transformed data such that,

$$f(u;\mathrm{C}_r)=\begin{cases}\left(\dfrac{\pi}{2}\right)u, & \text{if } r=1\\ \displaystyle\sum_{k=1}^{r-1}C_k u^k+\left(\frac{\pi}{2}-\sum_{k=1}^{r-1}C_k\right)u^r, & \text{if } r>1,\end{cases}$$

for $u\in[0,1]$, where C_r is the coefficient vector

$$\mathrm{C}_r\equiv\begin{cases}\pi/2, & \text{if } r=1\\ [\mathrm{C}_1,\ldots,\mathrm{C}_{r-1}] & \text{if } r>1,\end{cases}$$

subject to the constraint $f_r^{'}(u;\mathrm{C}_r)\geq 0$ for all $u\in[0,1]$. Note that for $r\geq 2$ the nonnegativity constraint is equivalent to requiring the zeroes of the degree $(r-1)$ polynomial $f_r^{'}(u;\mathrm{C}_r)$ lies outside the interval $(0,1)$. To determine the appropriate degree for the statistical model, a modified likelihood ratio test is used. For successive values of *r*, the vector C_r is estimated via constrained least squares, yielding:

$$\tilde{\mathrm{C}}_r=\arg\min\sum_{i=1}^{m}\left[Y_i-f_r(Z;\hat{\mathrm{C}}_r\right]^2$$

for $r\geq 2$. The corresponding error sum of squares for the degree – *r* fit is:

$$\mathrm{SSE}_r=\sum_{i=1}^{m}[Y_i-f_r(Z;\hat{\mathrm{C}}_r)]^2$$

for $r\geq 2$; and for $r=1$, we take:

$$\mathrm{SSE}_1=\sum_{i=1}^{m}\left[Y_i-\frac{\pi i}{2(m+1)}\right]^2.$$

The corresponding total sum of squares is:

$$\mathrm{SST}=\sum_{i=1}^{m}(Y_i-\bar{Y})^2,$$

where

$$\bar{Y}=m^{-1}\sum_{i=1}^{m}Y_i,$$

is used in the first step of the modified likelihood-ratio procedure is appropriate. The response variance σ^2 for each postulated value of r,

$$\tilde{\sigma}^2 = \mathrm{SSE}_r / m \text{ for } r = 1, 2, ...$$

The associate likelihood function is the joint distribution of observations Y_i and has the form:

$$(2\pi e \tilde{\sigma}_r^2)^{-m/2};$$

and the resulting log-likelihood function for degree r is

$$\psi_r(\tilde{\mathrm{C}}_r; Y) = -\frac{m}{2}[\ln(2\pi) + 1 + \ln(\tilde{\sigma}_r^2)].$$

The degree r is determined using the following likelihood ratio test at the level of significance α (where $0 < \alpha < 1$). The estimate of r is:

$$\tilde{r} = \begin{cases} 1, & \text{if } \mathrm{SSE}_1 / \mathrm{SST} < 0.01 \text{ or } m \le 2, \\ \min\left\{ r : r \ge 2; -m \ln\left(\dfrac{\tilde{\sigma}_r^2}{\tilde{\sigma}_{r-1}^2} \right) \le \chi_{1-\alpha}^2(1) \right\} - 1, & \text{otherwise,} \end{cases}$$

where $\chi_{1-\alpha}^2$ (1) denotes $1-\alpha$ quantile of the chi-squared distribution with 1 degree of freedom. The existing LRT procedure is modified as explained in section 4.1.2 such that if value,

$$-m \ln\left(\frac{\tilde{\sigma}_2^2}{\tilde{\sigma}_1^2} \right) \le \chi_{1-\alpha}^2,$$

coefficient vector $\tilde{\mathrm{C}}_3$ is estimated with a maximum of likelihood estimates for variance being calculated. The test is repeated at to check:

$$-m \ln\left(\frac{\tilde{\sigma}_3^2}{\tilde{\sigma}_2^2} \right) \le \chi_{1-\alpha}^2.$$

A linear fit is estimated only if both conditions are satisfied. If at degree, $r = 3$, it is found that the fitting improves with additional coefficient then successively higher order fits are estimated until the further addition of coefficients does not improve the quality of fit per equation (6). The

basis for this modification is explained in experimentation Section 5.2 where underfitting is seen for test sets with a small number of arrivals. The modified algorithm of the estimation procedure is shown in Figure 4.2.

Given the results of likelihood ratio test algorithm described in Section 4.1.2 a degree $\tilde{r}$ polynomial is fit to points $[t_{(i)}, W_i]$ $i = 1, 2, ..., m$. The polynomial vector $\mathrm{B}_{\tilde{r}}$ is estimated by minimizing the function $R(t; r, \mathrm{B}_{\tilde{r}})$ by applying constrained least squares to original data (untransformed data), yielding

$$\tilde{\mathrm{B}}_{\tilde{r}} = \underset{\hat{\mathrm{B}}_{\tilde{r}}: R'(u;\tilde{r},\hat{\mathrm{B}}_{\tilde{r}})\geq 0}{\arg\min} \sum_{i=1}^{m} \left[W_i - R(t_{(i)}; \tilde{r}, \hat{\mathrm{B}}_{\tilde{r}}) \right]^2$$

provided $r \geq 2$; and if $\tilde{r} = 1$ then we take $\tilde{\mathrm{B}}_1 = \tilde{\beta}_1 = 1$ in equation (2) resulting in a linear mean-value function . The final estimator for mean value function (1) is:

$$\tilde{\mu}(t) = \bar{N}(S) R(t; \tilde{r}, \tilde{\mathrm{B}}_{\tilde{r}}) \text{ for all } t \in (0, S].$$

4.2.2 Example for Multiple Replications

To understand the effectiveness of modeling multiple reproductions of NHPP, an example is discussed in this section. Arrival times from various numbers of realizations serve as input for the fitting procedure. NHPP realizations are generated using over duration, $S = 4$ and each realization has $\mu(S) = 160$ arrivals. The polynomial function used for data generation is

$$R(s) = 0.892620125\, s \text{ -}1.190605473\, s^2 + 0.783775048\, s^3 \text{ -}0.215066796\, s^4 + 0.020873734\, s^5.$$

Initially a single realization is utilized for the fitting procedure. Figure 4.8 shows a quadratic fit to the cumulative arrivals from a single realization. This lack of fit appears to be due to inadequate information available about arrival pattern from the small data set. To attempt to improve the quality of fit, arrival times from two realizations are used as input for the fitting method. The number of arrivals for two realizations are $N_1(S) = 156$ and $N_2(S)$=166 respectively

and are plotted super imposed in Figure 4.6. Arrival data over the two realizations are arranged in ascending order then transformed using the variance stabilization technique. The likelihood ratio test is employed to estimate degree r for superimposed arrivals from two realizations. Figure 4.7 displays average of 156 arrivals from two NHPPs fitted by the mean-value function. Graph in figure 4.7 shows improvement in fitting mean-value function for two realizations over single realization. Polynomial coefficients for estimated mean-value function for increasing number of realizations are shown in table 4.1. Results in table 4.1 imply that, the estimated coefficient values are closer actual coefficient values with increasing number or realizations. Degree $r=5$ polynomial is fitted for all the cases of NHPP realizations $P>1$. The difference between the actual and estimated polynomial function decreases with greater value of P. The Mean Square Error (MSE) is computed as discussed in Section 5 to see the effect of increasing the number of realizations. MSE values for different realizations are shown in the table.

Table 4.1 Polynomial Coefficients of the Estimated Mean-Value Function for Varying Process Realizations

		Estimated Coefficients				
P	**Degree *r***	β_1	β_2	β_3	β_4	β_5
1	1	0.25				
2	5	1.01915263	-1.35254257	0.87469725	-0.23970228	0.02338597
3	5	0.99861027	-1.30065112	0.83464615	-0.22713855	0.02201767
4	5	0.96003742	-1.26082643	0.81504520	-0.22216773	0.02152844
5	5	0.92450261	-1.17984882	0.76586372	-0.21081195	0.02063687

Table 4.2 Mean Square Error for the Estimated Mean-Value Function for Varying Process Realizations

P	Mean Square Error
1	0.78352
2	0.00386
3	0.0019
4	0.0012
5	0.0011

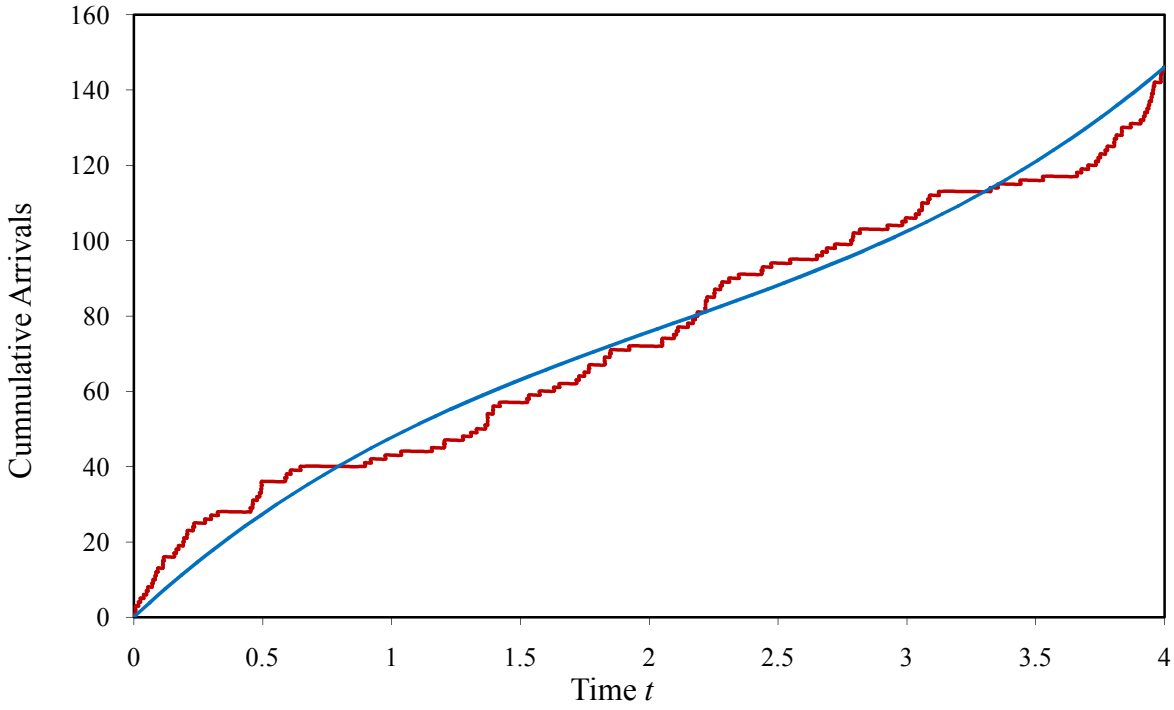

Figure 4.6 Estimated Mean-Value Function for Single Realizations

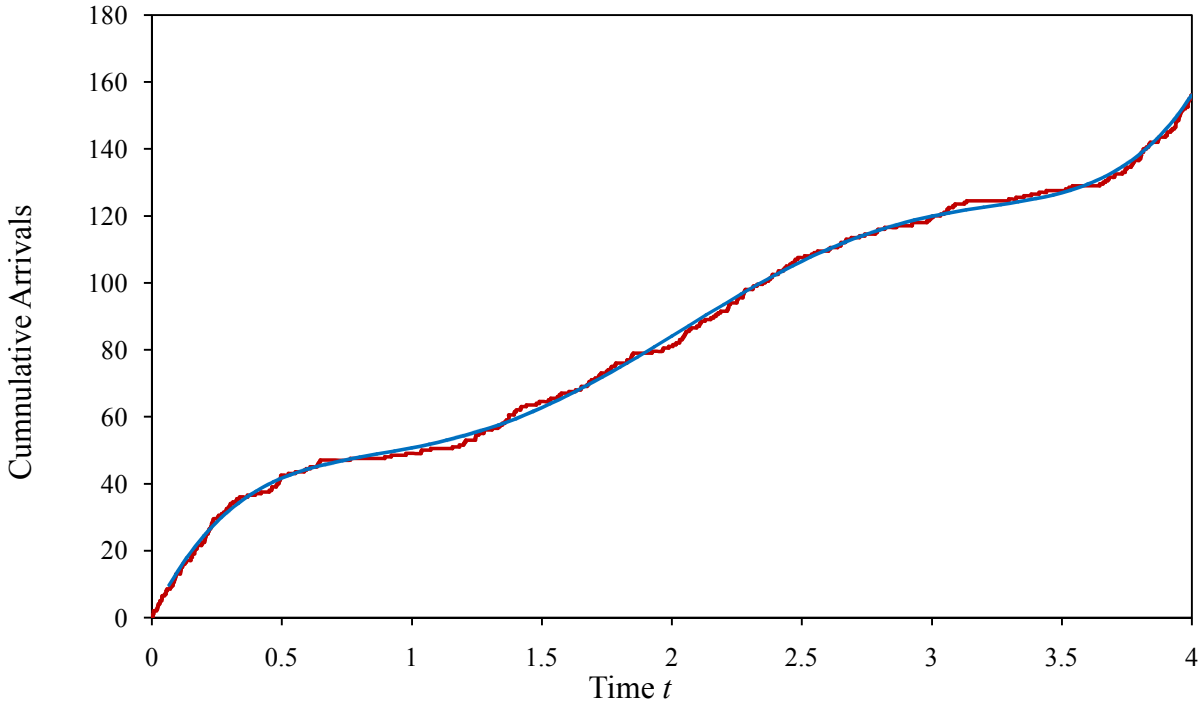

Figure 4.7 Estimated Mean-Value Function for Two Realizations

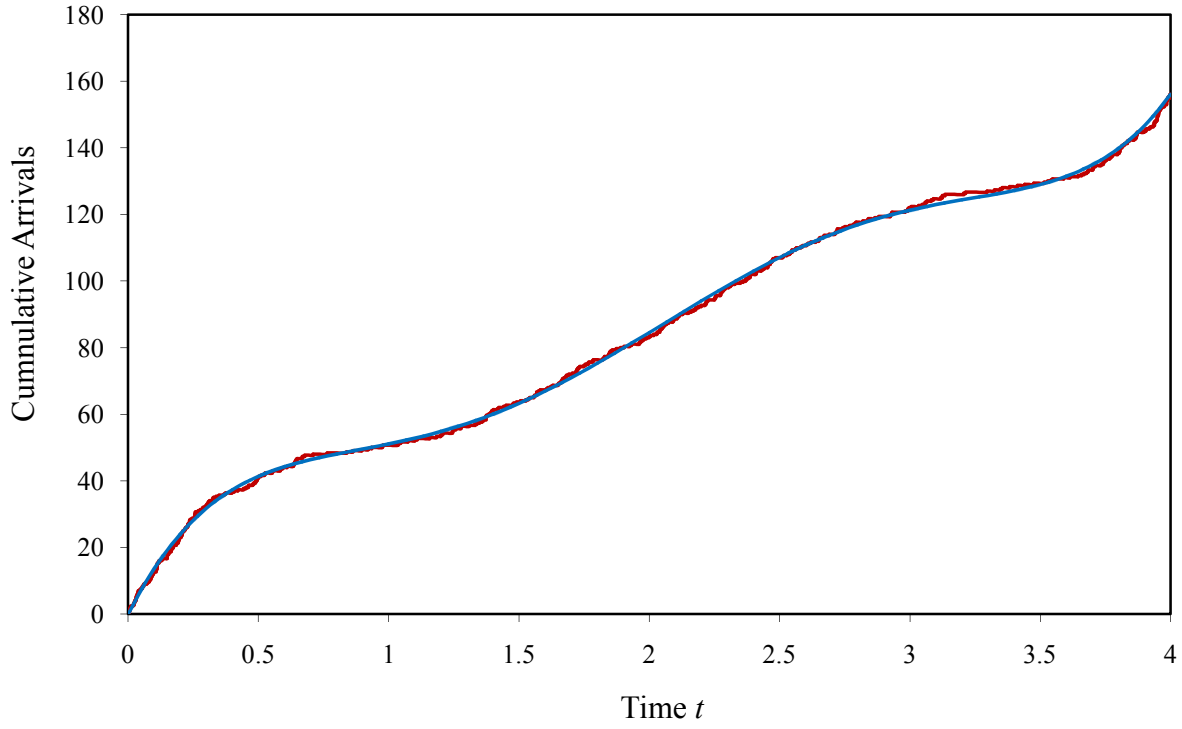

Figure 4.8 Estimated Mean Value Function for Three Realizations

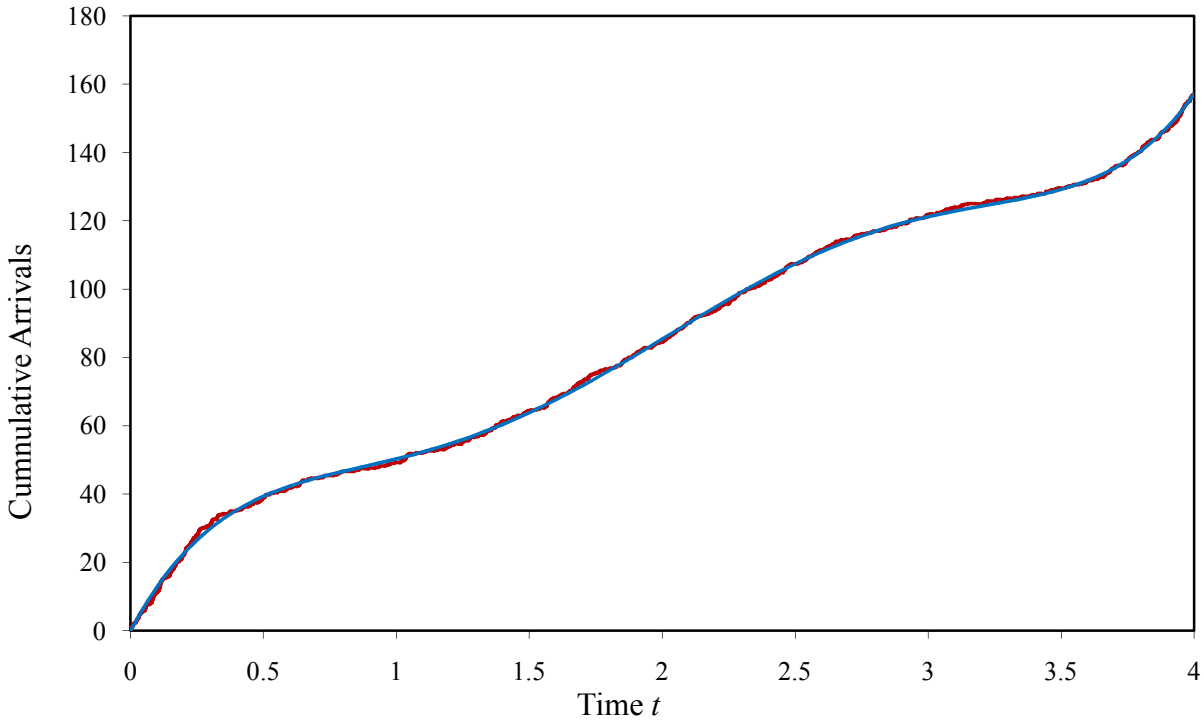

Figure 4.9 Estimated Mean Value Function for Four Realizations

4.3 Generation of Arrival Times for NHPPs with Semiparamteric Mean-Value Function

The estimation procedure fits a polynomial function of special form to cumulative arrivals. Given an NHPP with mean-value function of the form $\mu(t) = \bar{N}(S)R(t)$, an inversion-type algorithm is employed. An arrival time is generated by approximating an inverse of polynomial function. A bisection method is used to estimate the value of:

$$t = R^{-1}(x)$$

where x is cumulative arrival. The procedure for generation of arrival times is explained in Figure 4.10

Generation Method Algorithm

[1] Initialize fraction of cumulative arrivals $x = 0$

[2] Generate $u = \text{uniform}(0,1)$

[3] Estimate $\delta = \left(-\ln(1-u)\right) / N(S)$

[4] Set $x = x + \delta$

[5] Estimate arrival time t from cumulative arrival value x and polynomial coefficients using Bisection method to solve for $t = R^{-1}(x)$ and deliver t as next arrival time

[6] If $t < S$ go to [2] otherwise stop.

Figure 4.10 Procedure for Simulating NHPPs

5. Experimental Performance Evaluation

This research develops a method to fit a special form of polynomial function to cumulative arrivals from a single or multiple realizations of NHPPs. Statistical measures are utilized to evaluate the performance of the developed method in fitting the mean-value function to different types of NHPPs. A rigorous testing is conducted on the developed method, to verify the consistency of results for different test scenarios. The following sections discuss the numerical performance measures used for testing, test set up and test results.

5.1 Performance Measures

A set of experiments are conducted to evaluate the ability of the semiparametric model to fit both the underlying NHPP and the observed data with sufficient accuracy. These numerical measures are based on performance measures developed by Johnson et al. (1991), Kuhl et al. (1997) and Kuhl et al.(2006) to test the EPT, EPTMP- type rate functions, and the mean-value function, respectively and are used to verify the estimation method. Two sets of statistical performance measures are used. The first set of measures allow the comparison of the estimated mean value function with the actual function while the second set of parameters compare the estimation process with actual data. For completeness and reference, these performance measures are included here. To obtain measures of performance, the fitting procedure will be replicated multiple, K, times on data sets generated for each case.

In the following discussion, $\tilde{\lambda}_k(t)$ refers to the estimated rate function of k^{th} replication and $\tilde{\mu}_k(t)$ refers to the estimated mean-value function or the integrated function of k^{th} replication. As defined by Johnson et al. (1991), the *average absolute error* δ_k and the *maximum absolute error* δ_k^* in estimation of rate function for k^{th} replication are:

$$\delta_k = \frac{1}{S}\int_0^S \left|\tilde{\lambda}_k(t) - \lambda(t)\right| dt$$

and

$$\delta_k^* = \max\left\{\left|\tilde{\lambda}_k(t) - \lambda(t)\right| : 0 \le t \le S\right\}$$

for $k = 1,...,K$. In addition, the *average absolute deviation delta* Δ_k and *maximum absolute deviation* Δ_k^* are defined for mean value function as:

$$\Delta_k = \frac{1}{S}\int_0^S \left|\tilde{\mu}_k(t) - \mu(t)\right| dt$$

and

$$\Delta_k^* = \max\{\left|\tilde{\mu}_k(t) - \mu(t)\right| : 0 \le t \le S\}$$

for $k = 1,...,K$. Johnson et al. (1991) also develop aggregate performance measures for errors in estimating rate function for all the realizations. The sample mean of observations $\{\delta_k : k = 1,...,K\}$ is denoted by $\overline{\delta}$, and the sample coefficient of variation for all observations V_δ is computed as:

$$V_\delta = \frac{\left[\frac{1}{K-1}\sum_{k=1}^{K}(\delta_k - \overline{\delta})^2\right]^{1/2}}{\overline{\delta}}.$$

Maximum error values are also computed similarly from the observations $\{\delta_k^* : k = 1,...,K\}$ denoted by $\overline{\delta}^*$ and V_{δ^*}. Analogous performance measures for errors in estimation of the mean-value function are computed and are denoted as $\overline{\Delta}$ and V_Δ for the sample mean and coefficient of variation; the maximum values are represented by $\overline{\Delta}^*$ and V_Δ^* respectively. Normalized statistics reported by Kuhl et al. (1997) are also computed to facilitate comparison of results for different rate and mean value functions:

$$Q_{\delta} = \frac{\overline{\delta}}{\mu(S)/S} \text{ and } Q_{\delta^*} = \frac{\overline{\delta}^*}{\mu(S)/S}$$

and

$$Q_{\Delta} = \overline{\Delta} \Big/ \left[\frac{1}{S} \int_0^S \mu(s)dt \right] \text{ and } Q_{\Delta^*} = \overline{\Delta}^* \Big/ \left[\frac{1}{S} \int_0^S \mu(s)dt \right].$$

Apart from the statistics mentioned above, Kuhl et al. (1997) developed statistics to measure the ability of LRT algorithm to approximate each observed arrival process. On the k^{th} replication of given NHPP they let $\{t_{i,k} : i = 1, 2, ..., N_k(S)\}$ denote the arrival epochs observed in time interval $(0, S]$. The k^{th} replication of sum of square estimation errors and the mean squared estimation errors are given by

$$\mathrm{SS_E}(\tilde{\Theta})_k \equiv \sum_{i=1}^{N_k(S)} [\tilde{\mu}_k(t_{i,k}) - j]^2$$

and

$$\mathrm{MS_E}(\tilde{\Theta})_k \equiv \mathrm{SS_E}(\tilde{\Theta})_k \Big/ N_k(S).$$

Further, V_{SS_E} represents the sample coefficient of variation of the values $\{\mathrm{SS_E}(\tilde{\Theta})_k : k = 1, 2, 3....., K\}$, and $V_{\mathrm{MS_E}}$ represents the sample mean and sample coefficient of variation of the values $\{\mathrm{MS_E}(\tilde{\Theta})_k : k = 1, 2, 3....., K\}$.

The *average absolute error* and *maximum absolute error* occurring in the estimation of the mean-value function for a given number of replications are calculated as:

$$D_k = \frac{1}{N_k(S)} \sum_{i=1}^{N_k(S)} \left| \tilde{\mu}_k(t_{i,k}) - i \right|$$

and

$$D_k^* = \max \left\{ \left| \tilde{\mu}_k(t_{i,k}) - i \right| : 1 \le i \le N_k(S) \right\}$$

for $k = 1,...,K$. In addition $\bar{D}$ and $\bar{D}^*$ denote sample means of the observed values of error are calculated for $\{D_k : k = 1,2,...,K\}$ and $\{D_k^* : k = 1,2,...,K\}$, respectively. Kuhl et al. (2000) also formulate two types of aggregate performance measures to compare error values $\bar{D}$ and $\bar{D}^*$ across the experimental cases. The first type utilizes the grand average level of the empirical mean-value functions computed over all *K* replications to normalize the average performance measures $\bar{D}$ and $\bar{D}^*$ so that:

$$Q_D = \frac{\bar{D}}{(1/K)\sum_{k=1}^{K}(1/S)\int_0^S N_k(t)dt}$$

and

$$Q_D^* = \frac{\bar{D}^*}{(1/K)\sum_{k=1}^{K}(1/S)\int_0^S N_k(t)dt}.$$

The second type of aggregate performance measure is estimated by expressing each performance measure $\bar{D}_k$ and $\bar{D}_k^*$ observed in k^{th} replication as a percentage of average level of the empirical mean value function on that replication. Kuhl *et al.* (2000) average the resulting normalized statistics over all K replications, yielding:

$$H_D = \frac{1}{K}\sum_{k=1}^{K}\frac{D_k}{(1/S)\int_0^S N_k(t)dt}$$

and

$$H_{D^*} = \frac{1}{K}\sum_{k=1}^{K}\frac{D_k^*}{(1/S)\int_0^S N_k(t)dt}$$

In addition to these performance measures, graphs are plotted for the estimated mean-value function and estimated instantaneous rate function with a tolerance band to visually evaluate quality of estimates.

Suppose $\tilde{\mu}_{(k)}(t)$ is estimated mean-value function for k^{th} replication and there are K replications. Let $\tilde{\mu}_{(1)}(t) < \tilde{\mu}_{(2)}(t) < \ldots < \tilde{\mu}_{(k)}(t)$ for a fixed time $t \in (0, S]$ represent ordered estimates for the mean-value function for K replications. Then an approximate $100(1-\beta)$ tolerance interval for $\mu(t)$ is obtained as:

$$\left[\tilde{\mu}_{(\lceil K\beta/2 \rceil)}(t), \tilde{\mu}_{(\lceil K\{1-\beta/2\} \rceil)}(t) \right]$$

where $\lceil z \rceil$ denotes smallest integer greater than or equal to z. Similarly, the tolerance interval are determined for instantaneous rate function $\lambda(t)$ at a fixed time $t \in (0, S]$.

To evaluate the accuracy of the estimation procedure fit to multiple process realizations, we form a statistical performance measure on similar basis to there in Kuhl et al. (2006). The sum of square errors over P realizations for k^{th} replication is estimated as

$$\text{SS}_{\text{E}}(\tilde{\Theta})_k = \sum_{i=1}^{m_k} \left(\tilde{\mu}_k(t_{i,k}) - \frac{i}{P} \right)^2$$

where

$$m_k = \sum_{p=1}^{P} N_{pk}(S)$$

is number of arrivals in P realizations on the k^{th} replication. The mean of sum of square errors is defined as:

$$\text{MS}_{\text{E}}(\tilde{\Theta})_k = \frac{\text{SS}_{\text{E}}(\tilde{\Theta})_k}{m_k}.$$

Let $\overline{SS}_E$ and V_{SS_E} respectively represent the sample mean and the sample coefficient of variation of the observed values $\left\{\mathrm{SS}_{\mathrm{E}}\left(\tilde{\Theta}\right)_k : k = 1,2,3,....,K\right\}$ such that:

$$\overline{SS}_E = \frac{1}{K}\sum_{k=1}^{K} SSE_k \quad \text{and} \quad V_{SS_E} = \frac{1}{\overline{SS}_E}\frac{\sqrt{\sum_{k=1}^{K}\left(SSE_k - \overline{SS}_E\right)^2}}{K-1}.$$

Similarly $\overline{MS}_E$ and V_{MS_E} respectively represent the sample mean and sample coefficient of variation of the observed values $\left\{MS_E\left(\tilde{\Theta}\right)_k : k = 1,2,...,K\right\}$ such that:

$$\overline{MS}_E = \frac{1}{K}\sum_{k=1}^{K} MSE_k \quad \text{and} \quad V_{MS_E} = \frac{1}{\overline{MS}_E}\frac{\sqrt{\sum_{k=1}^{K}\left(MSE_k - \overline{MS}_E\right)^2}}{K-1}.$$

On the k^{th} replication of a given test process, the average absolute error and the maximum absolute error incurred in estimating empirical mean value function are given by:

$$D_k \equiv \frac{\sum_{i=1}^{m_k}\left|\tilde{\mu}_k(t_{i,k}) - \frac{i}{P}\right|}{m_k}$$

and

$$D_k^* \equiv \max\left\{\left|\tilde{\mu}_k(t_{i,k}) - \frac{i}{P}\right| : i = 1,...,m_k\right\}$$

respectively, for $k = 1,...,K$. $\bar{D}$ and $\bar{D}^*$ represent the sample means of observed values $\left\{D_k : k = 1,...K\right\}$ and $\left\{D_k^* : k = 1,...K\right\}$ respectively defined as:

$$\bar{D} = \frac{\sum_{k=1}^{K} D_k}{K} \text{ and } \quad \bar{D}^* = \frac{\sum_{k=1}^{K} D_k^*}{K}.$$

Similarly, sample standard error and sample standard deviation is estimated for each replication.

5.2 Experimentation using a Single Realization of the Target Process

The numerical performance measures listed in section 5.1 are calculated for various test cases to determine the accuracy of estimation method. The following sections discuss the various test cases used for experimental evaluation and results of the experiments.

5.2.1 Experimental Setup

Experimentation on the fitting method is conducted by fitting mean-value function to arrival sets from various NHPP realizations. The generation method is used to generate the test data in the form of arrival times from the given set of polynomial coefficients. For each test case, the fitting method is used to fit mean-value function for hundred different data sets generated by the generation method. A maximum degree of $r = 10$ is fitted to the NHPP arrivals. The confidence interval for chi-square test is set to 90% for all the cases. The estimated mean-value function by the fitting method is compared with the underlying mean-value function using numerical performance measures discussed in section 5.1. The efficiency of the fitting method is also gauged by evaluating the fitted function against actual arrival data. The two types of performance measures serve as goodness-of-fit indicator for the fitting method. The experimentation is conducted using six different test cases. The polynomial coefficients utilized to generate the arrival data for the NHPP is shown in table 5.1. Table 5.1 also displays all other details pertaining to the test cases like duration length, S and total number of arrivals over the observation duration, $N(S)$.

Table 5.1 Input Polynomial Coefficients for the six Experimental Cases

Case	$N(S)$	S	Degree r	Input Coefficients					
				β_1	β_2	β_3	β_4	β_5	β_6
1	2000	4	5	0.892620125	-1.190605473	0.783775048	-0.215066796	0.020873734	
2	1000	1	3	1.62999919	-2.69999442	2.06999523			
3	3500	3.7	5	0.060520647	0.22894165	-0.068123488	-0.005186734	0.00297733067	
4	1500	10	3	0.284523809	-0.0561507936	0.003769841			
5	2000	4	6	0.72817869	-1.313493126	1.14226802	-0.428388746	0.071200209	-0.00420983218
6	4800	45	5	0.019711543	0.000485151	-0.0000743467	0.0000026824	-0.0000000277	

5.2.2 Performance Measures for Experimental Cases

The data is generated using the different polynomials of degree r shown in table 5.1 and a mean-value function is fitted to the data using the fitting method for hundred replications. Table 5.2 compares the degree of the fitted men-value function and the original degree of underlying mean-value function for hundred replications executed for each test case. As observed in the table, the fitted degree matches closely with the original degree.

Over-fitting is observed in cases where a large amount of data is available. For example, in experimental cases, 3 and 6 where 3500 and 4800 arrivals are generated respectively, a degree $r = 10$ polynomial is fitted for most of the replications. It is advisable to limit the over-fitting tendency of the fitting method by providing a lower value for the maximum degree r .

The second way to limit the overfitting tendency of the model is by using a lower value of α . For example in the experimental case 3, degree $r = 10$ is fitted to all the 100 replications of the NHPP processes when the α value is set at 0.1. If the value is reduced to 0.01 then it can be observed that the over fitting phenomenon is reduced from the conducted experimentations. For the fitted 100 replications of the NHPPS, with reduced α value, degree $r = 8$ is fitted to 13 replications, degree $r = 21$ is set to 21 replications and degree $r = 10$ is fitted to rest of the replications. The graph is plotted for the lowered α value in figure 5.6 for the 90% tolerance band for the mean-value function. If the graph for $\alpha = 0.1$ is compared with the $\alpha = 0.01$, then it can be observed that the fitting in the two models is not significant.

Table 5.2 Comparison between the Original and Fitted Degrees

Case	Original Degrees	Degree Fitted							
		3	4	5	6	7	8	9	10
1	5			94					6
2	3	70		21		3	16		
3	5								100
4	3	90			1	3		1	5
5	6			14	68	15			5
6	5			25	5				70

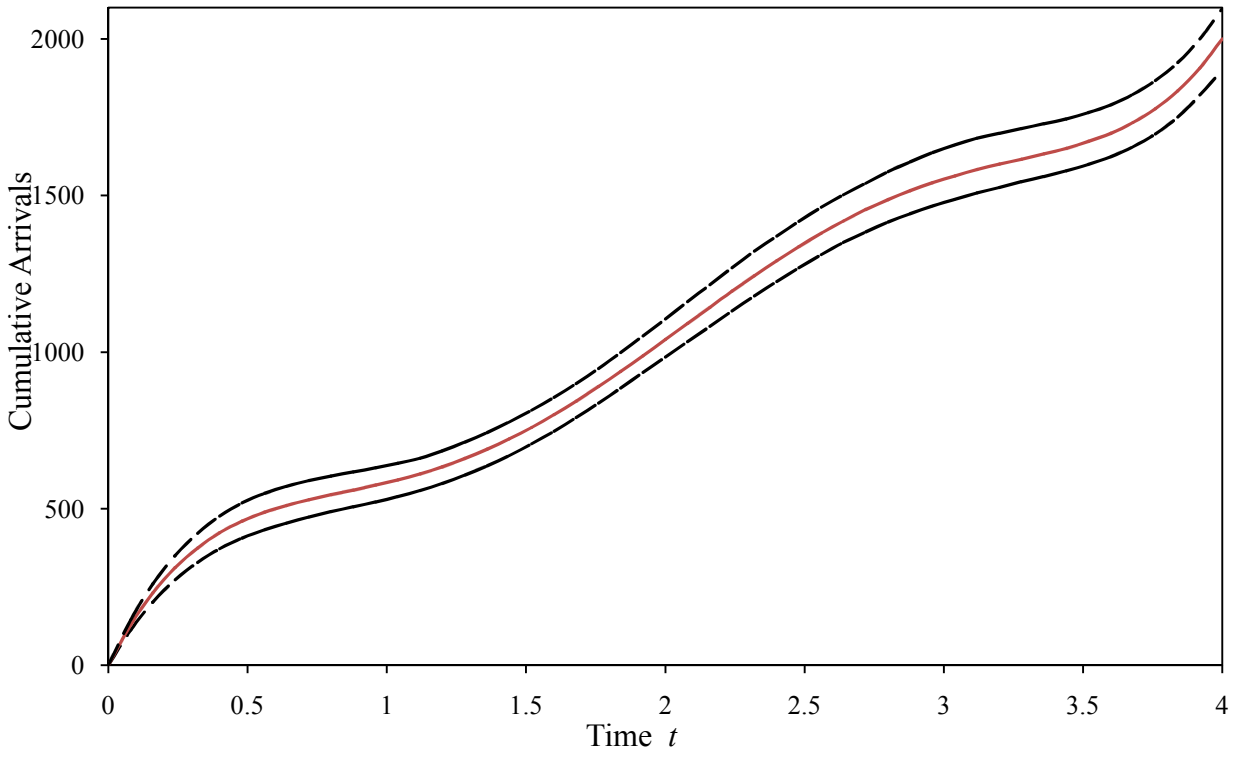

Figure 5.1 90% Tolerance Intervals for μ(t),t ∈ [0,4] in Case 1

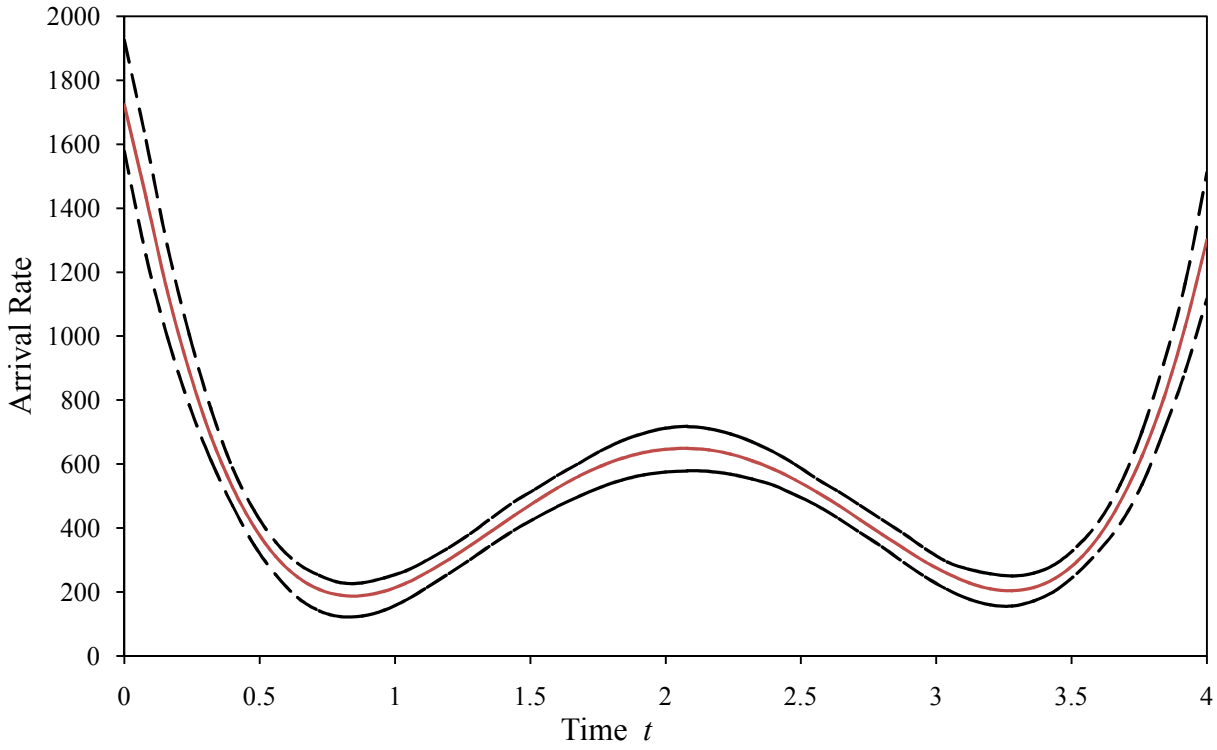

Figure 5.2 90% Tolerance Intervals for λ(t),t ∈ [0,4] in Case 1

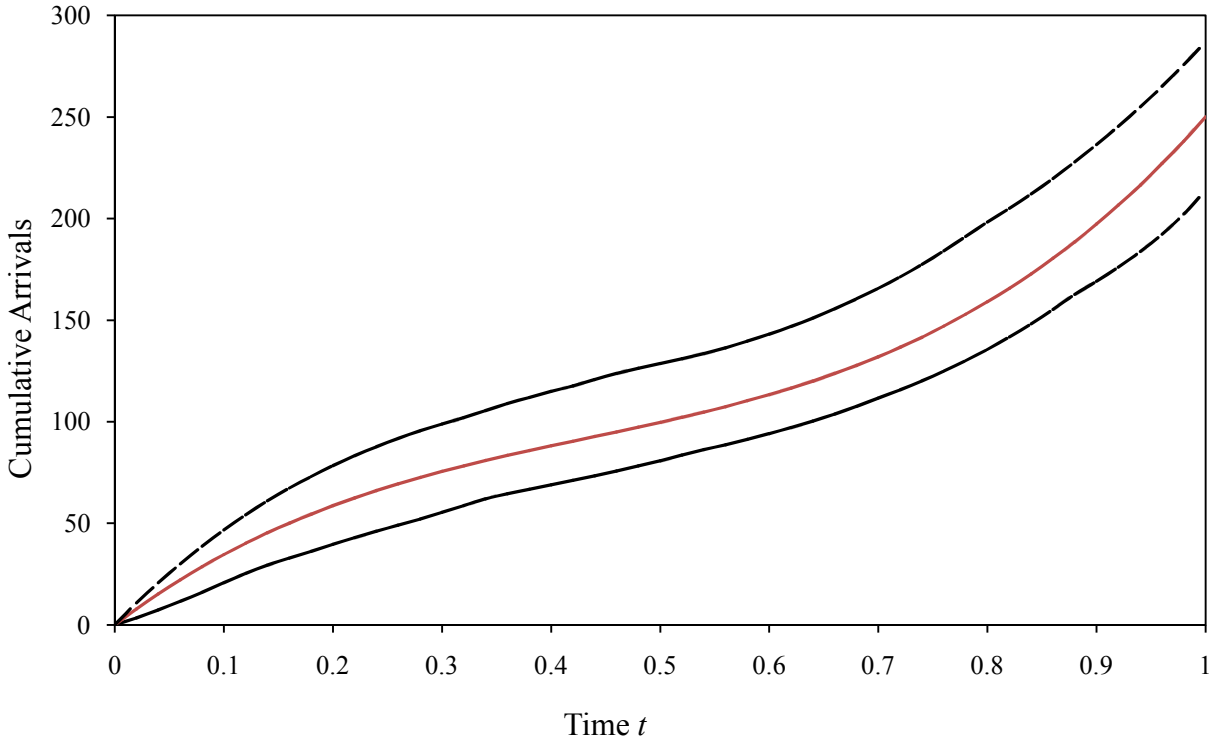

Figure 5.3 90% Tolerance Intervals for μ(t),t ∈ [0,1] in Case 2

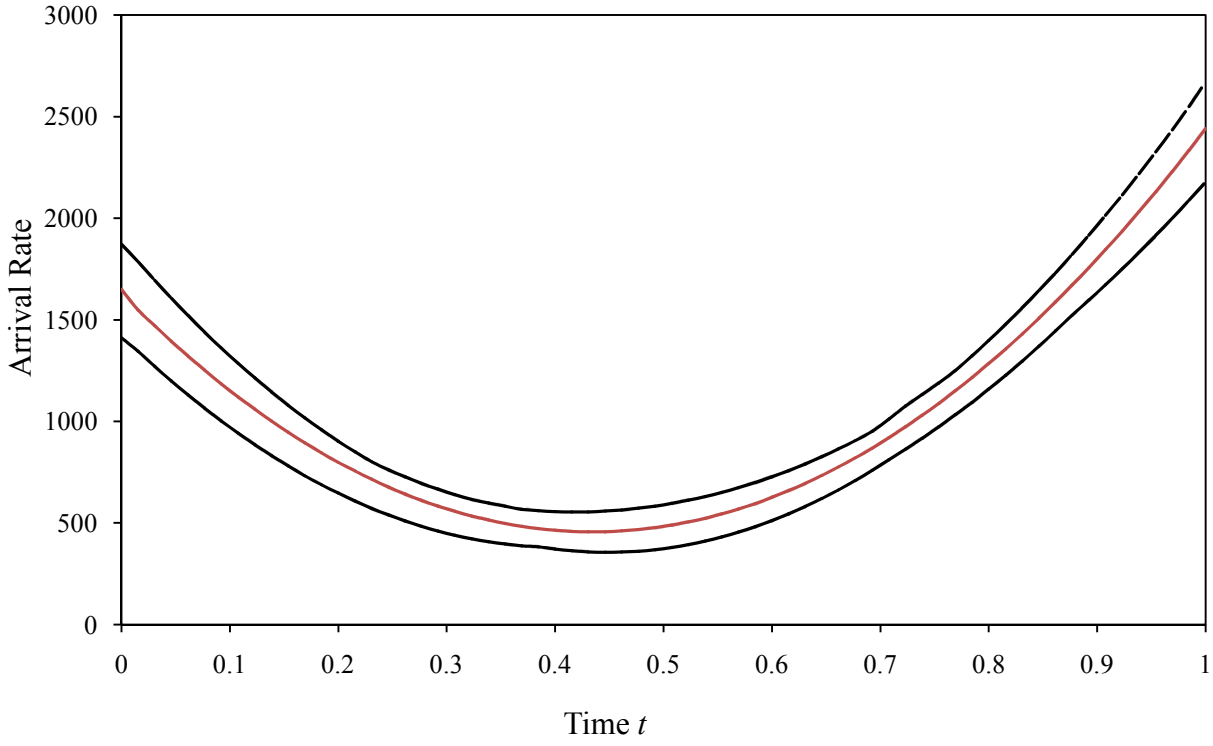

Figure 5.4 90% Tolerance Intervals for λ(t),t ∈ [0,1] in Case 2

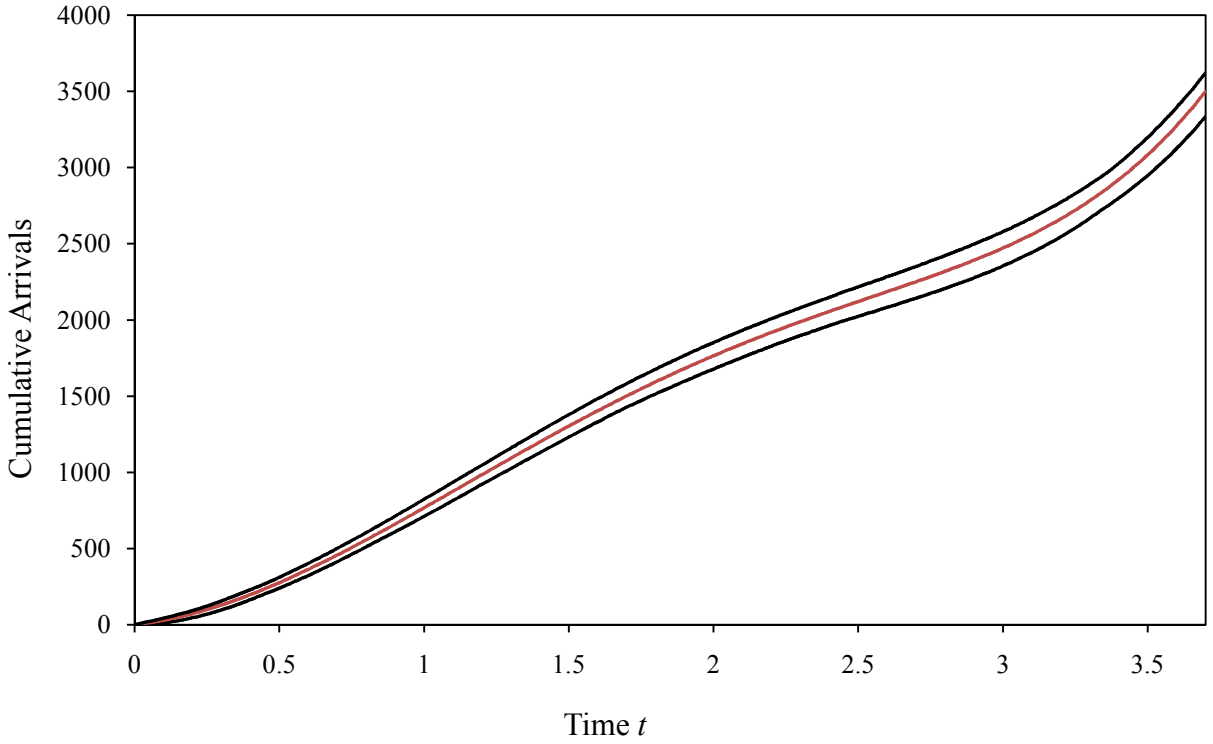

Figure 5.5 90% Tolerance Intervals for μ(t),t ∈ [0,3.7] in Case 3

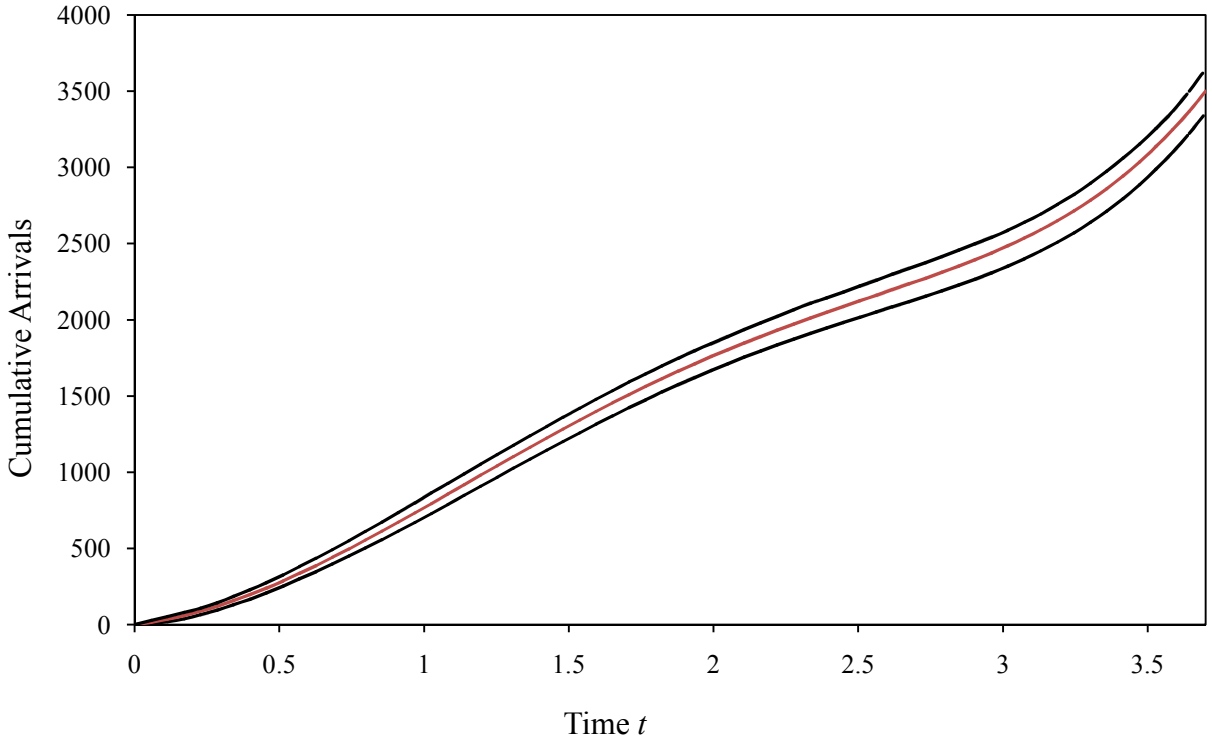

Figure 5.6 90% Tolerance Intervals for μ(t),t ∈ [0,3.7] in Case 3 for $\alpha = 0.01$

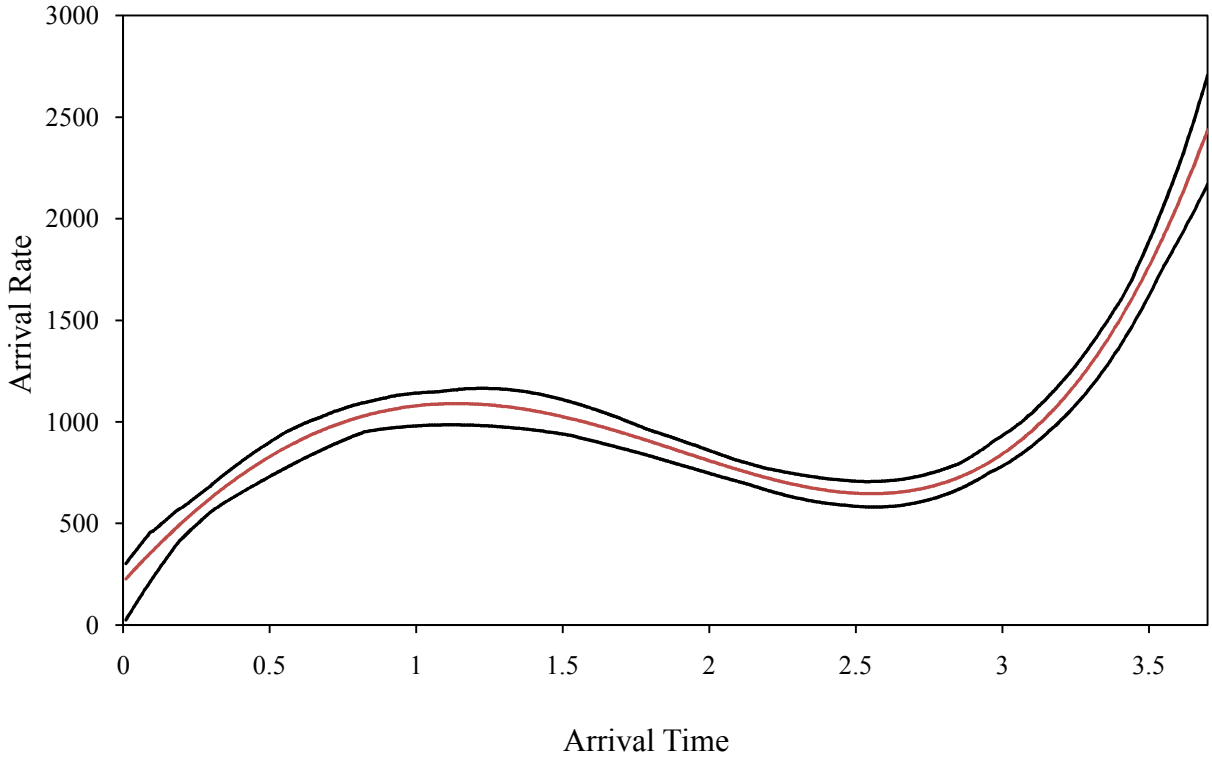

Figure 5.7 90% Tolerance Intervals for λ(t),t ∈ [0,3.7] in Case 3

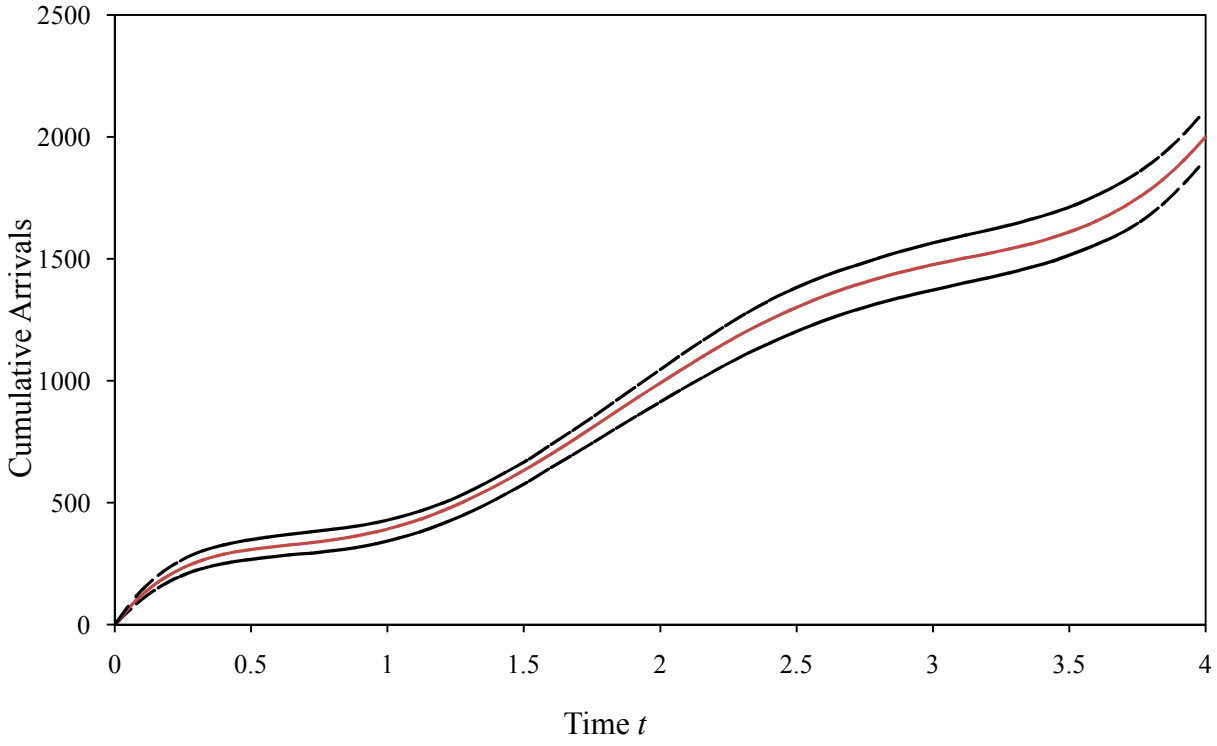

Figure 5.8 90% Tolerance Intervals for μ(t),t ∈ [0,4] in Case 4

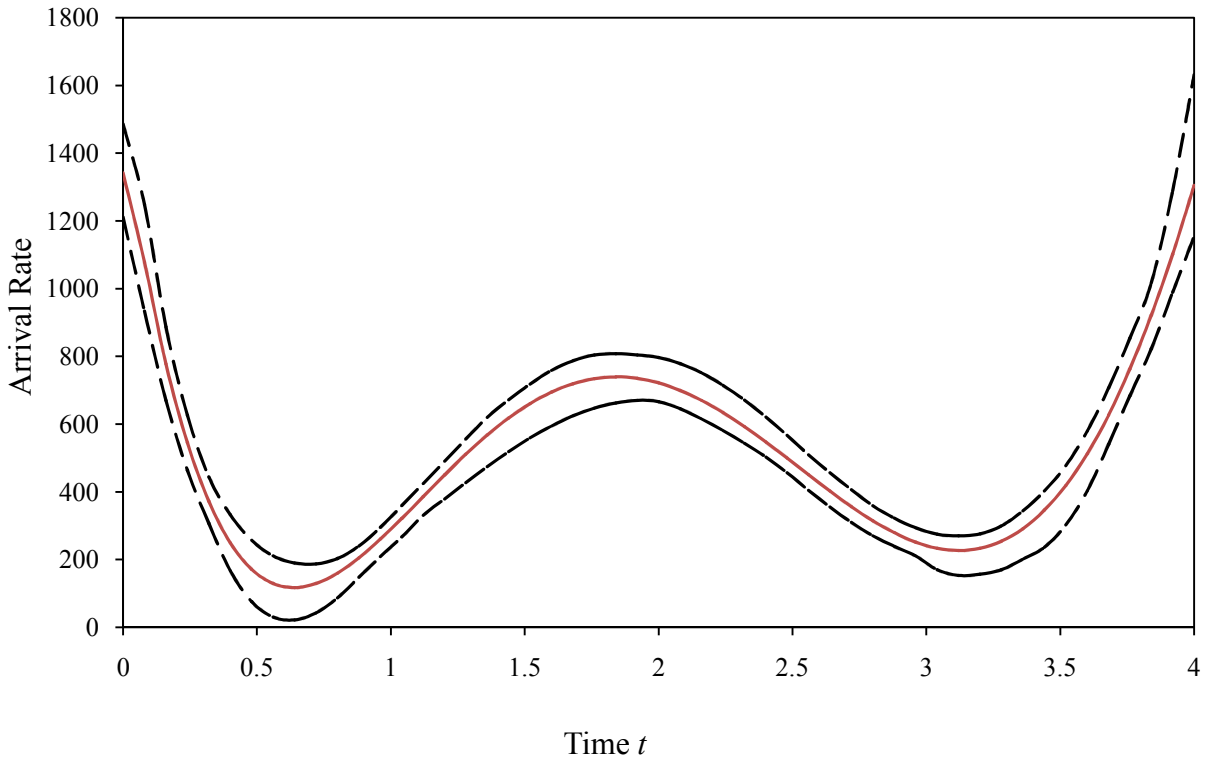

Figure 5.9 90% Tolerance Intervals for λ(t),t ∈ [0,4] in Case 4

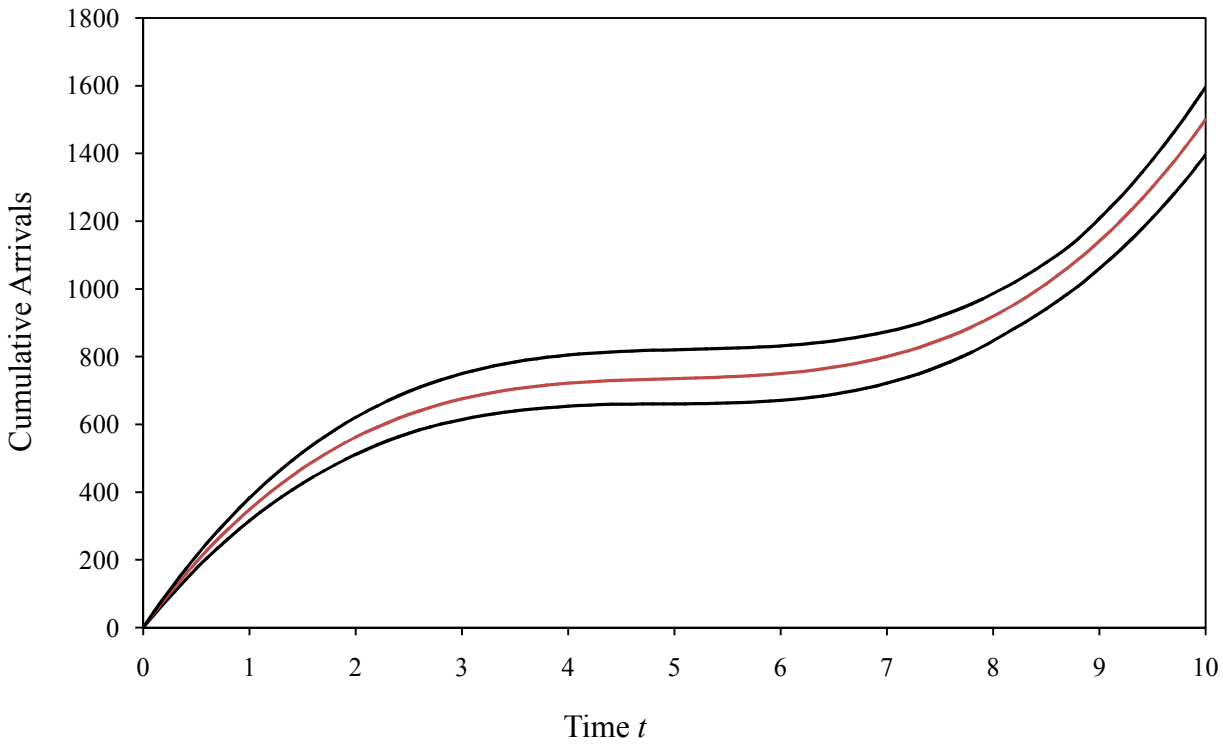

Figure 5.10 90% Tolerance Intervals for μ(t),t ∈ [0,10] in Case 5

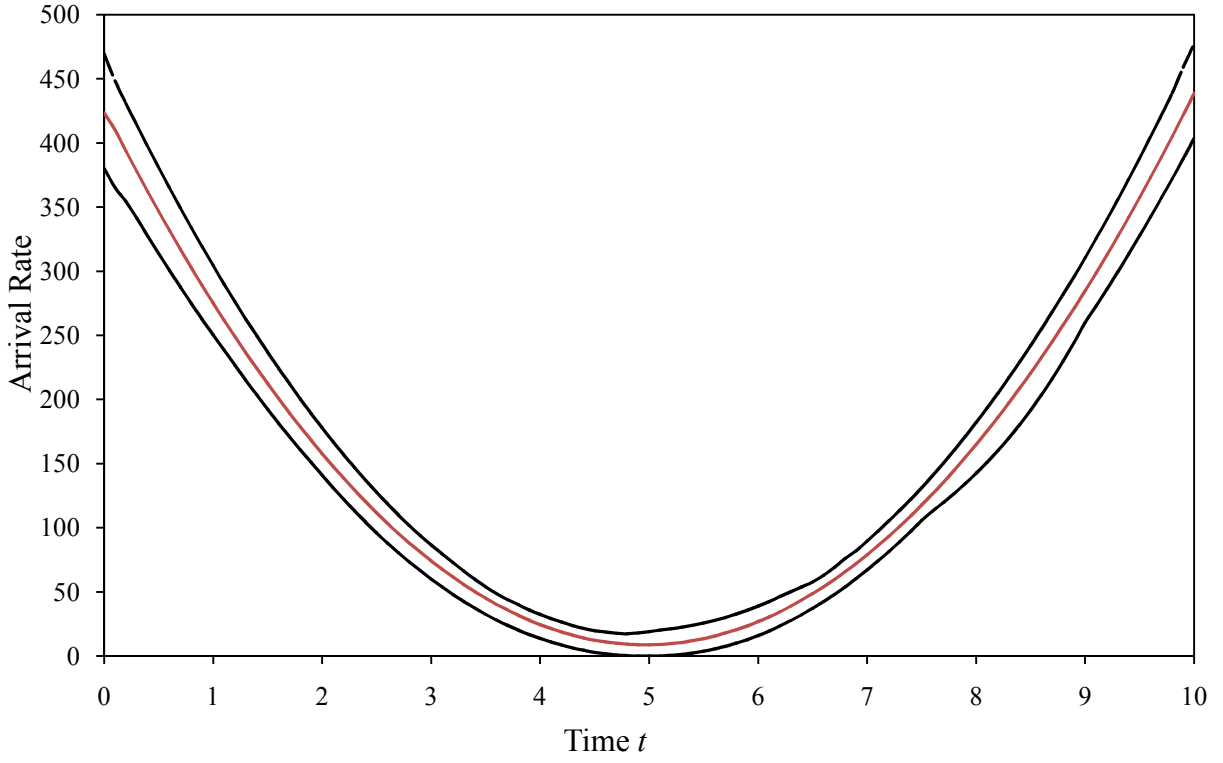

Figure 5.11 90% Tolerance Intervals for λ(t),t ∈ [0,10] in Case 5

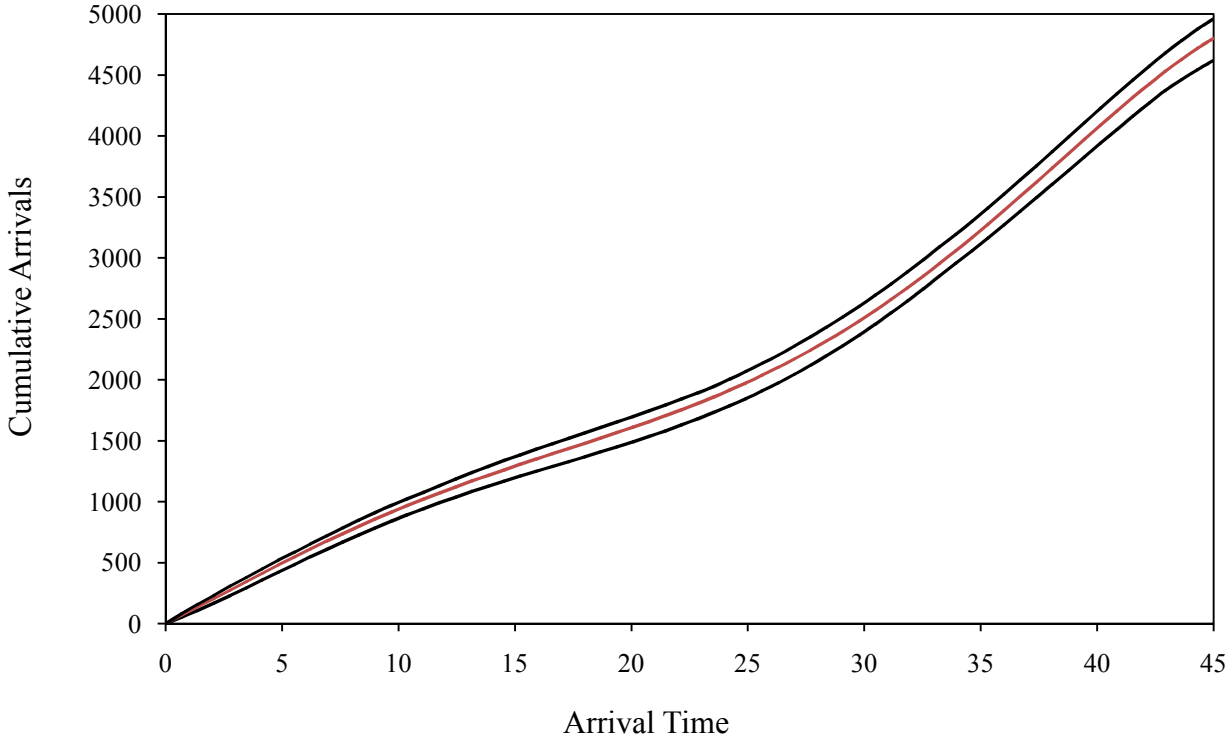

Figure 5.12 90% Tolerance Intervals for μ(t),t ∈ [0,45] in Case 6

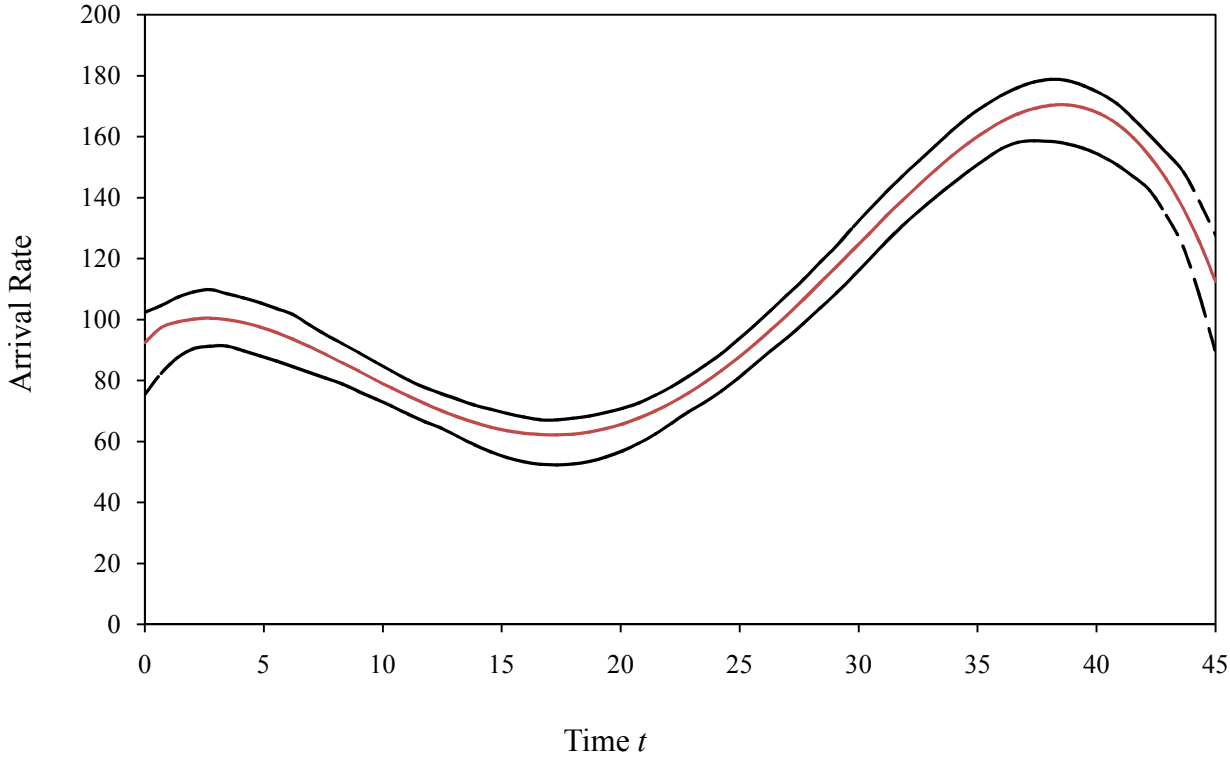

Figure 5.13 90% Tolerance Intervals for λ(t),t ∈ [0,10] in Case 6

To visually demonstrate the accuracy of fitting methods, graphs are plotted for 90% tolerance interval over the mean-value and the rate function for the experimental cases. The upper and the lower limits of the interval signify that 90 out of 100 replications lie between the tolerance limits. Figures 5.1 through 5.13 shows the graphs plotted for 90% tolerance interval for the mean-value and the rate function. The graphs are plotted for all the experimental cases. The dotted lines in the graphs represent the upper and the lower limit of mean-value and rate function. The continuous line represents the underlying mean-value and rate function. It can be observed from the graphs that the fitting method accurately estimates the mean-value function and that the accuracy of the estimation method increases with availability of data. For example, for experimental cases 3 and 6, where a large amount of arrival data is available $\left(N_3(S) = 3500, N_6(S) = 4800\right)$, a narrow tolerance interval is observed.

The numerical performance measures listed in section 5.1 are estimated for all the experimental cases. Table 5.3 shows the performance measures that compare the fitted mean-value function with the underlying mean-value function. Consistent error values are seen across all the cases. For example, variance in *average absolute error* V_δ is approximately same (0.3~0.4) for all the test scenarios. Table 5.4 shows the computed performance measures that compare the fitted mean-value function with the cumulative arrival data. A consistent pattern in error values is seen for all the test scenarios. Smaller values for *mean square error* conclude that the fitting method accurately estimates the mean-value function for the NHPPs. The computed numerical performance measures demonstrate the capability of fitting method is estimating the mean-value function accurately. The results show that the fitting method provides reliable results for all the different test scenarios.

Table 5.3 First Set of Statistical Performance Measures for 100 Replications.

Measure	1	2	3	4	5	6
δ	21.438	45.9650	29.542	6.7257	24.4720	2.7531
V_δ	0.3639	0.3881	0.3384	0.4239	0.4355	0.3512
Q_δ	0.0428	0.0459	0.0312	0.0448	0.0489	0.0258
$\overline{\delta}^*$	88.311	123.32	129.18	21.066	109.90	9.6938
V_δ^*	0.5156	0.4858	0.5068	0.5055	0.6152	0.5138
Q_δ^*	0.1767	0.1233	0.1365	0.1404	0.2198	0.0908
$\overline{\Delta}$	26.624	16.984	28.713	26.331	23.494	32.441
V_Δ	0.5872	0.5662	0.5865	0.6127	0.5640	0.6229
Q_Δ	0.0993	0.0400	0.0681	0.3520	0.0955	0.0424
$\overline{\Delta}^*$	49.071	32.368	60.629	44.7740	48.203	69.042

Table 5.4 Second Set of Statistical Performance Measures for 100 Replications

Measure	1	2	3	4	5	6
SS_E	55.8224	42.7774	88.3786	71.1191	103.182	121.474
V_{SS_E}	0.5147	0.6537	0.5219	0.7401	0.4650	0.7812
MS_E	0.0279	0.0433	0.0252	0.0474	0.0519	0.0253
V_{MS_E}	0.5153	0.6615	0.5198	0.7385	0.4662	0.7621
$\overline{D}$	5.3480	5.0801	6.1850	6.2937	6.7585	8.1403
$\overline{D}^*$	18.8792	16.5483	22.9147	20.5987	22.1706	29.5187
Q_D	0.0050	0.0118	0.0039	0.0086	0.0071	0.0039
Q_{D^*}	0.0179	0.0383	0.0148	0.0281	0.0231	0.0143
H_D	5.11E-5	1.51E-4	4.53E-5	1.94E-4	6.62E-5	3.81E-5
H_{D^*}	2.11E-4	5.02E-4	1.45E-4	6.05-4	2.41E-4	1.24E-4

5.3 Multiple Realizations for Single Resolution

In some cases there exists an opportunity to collect data from multiple process realizations of NHPPs. The developed fitting method can exploit the availability of multiple process realizations in estimating the mean-value function. To test the estimation procedure for multiple process realizations, experimentation is conducted using experimental cases. The numerical performance measures in section 5.1 are computed for multiple process realization. The following sections discuss the experimental set up and the results for test scenarios.

5.3.1 Experimental Setup for Multiple Process Realizations

The experimental set up for the multiple process realization is similar to that of the single process realization. The test data in form of arrivals from multiple process realizations of NHPPs is generated using the generation method. The polynomial coefficients used to generate the arrival data are shown in the table 5.5. The average number of arrivals in each process realization is $\bar{N}(S)$ and observation duration S for NHPPs is also shown in table 5.5. The number of process realizations generated for the first experimental case is $P_1 = 1,3,8,15$ and for the second experimental case is $P_2 = 1,3,5,8$. For each test scenario with multiple process realizations, hundred replications are generated and estimation method is used to fit a mean-value function to the multiple process realizations.

Table 5.5 Input Polynomial Coefficients for Multiple Process Realizations

				Input Coefficients				
Case	$\bar{N}(S)$	***S***	***r***	β_1	β_2	β_3	β_4	β_5
1	150	4	5	0.8926201	-1.190605473	0.783775048	-0.2150667	0.02087373
2	50	1	3	1.62999919	-2.69999442	2.06999523	-	-

5.3.2 Performance Measures for Multiple Process Realizations

The experiments are conducted for fitting the mean-value function to multiple process realizations. Table 5.6 shows the degree r of the fitted mean-value function for multiple process realizations of NHPPs. It can be observed from the table, that the accuracy in estimation of degree of the polynomial increases with increasing number of process realizations. For example in first experimental case a degree $r = 3$ is estimated for 28 replications while the correct degree $r = 5$ is estimated for 45 replications for single process realization. With increase in number of process realizations, $P = 3$, the correct fit of degree $r = 5$ is estimated for 77 replications. Similar effect is seen in second experimental case where the estimation of correct degree r improves with additional process realizations. From the conducted experimentation, it can be concluded that the estimation of degree r of polynomial improves with additional observed process realizations and the effect is evident in cases with sparse arrivals over single process realization.

Table 5.6 Comparison between Original and Fitted for varying Process Realizations

Case	Original Degrees	Realizations *P*	Degree Fitted							
			1	2	3	4	5	6	7	8
1	5	1			28	13	45	9	1	4
		3			6	1	77	7	5	4
		8			2		83	3	6	6
		15					90		7	3
2	3	1	2		79	4	8	5	3	1
		3			81		10		6	3
		5			84		9		4	3
		8			87		5		5	3

Figure 5.14 to 5.21 represents the graphs plotted for the 90% tolerance interval for the mean-value and the rate function. The tolerance band represents the range of 90 replications out of 100 replications, for the mean-value and rate function. The dotted lines in the graphs represent the interval limit while the continuous line represents the underlying function. It can be observed from the graphs that the tolerance interval narrows with increase in number of process realizations which proves that as P increases, accuracy in estimation improves.

The numerical performance measures are computed for the experiments to understand the effect of availability of additional process realizations on estimation procedure. Table 5.7 represents the first set or performance measures to compare the estimated function with the underlying function. Table 5.8 displays the second set of performance measure which compares the estimated function with the cumulative arrivals. The numerical measures are consistent with the graphs for tolerance bands and indicate that the error values decrease with increase in number of process realizations. For example, it can be seen from the table 5.7 that for the first experimental case, *average absolute error* reduces from 10.376 to 1.337 for $P=1$ to $P=15$ process realizations. Similarly, it can be seen in table 5.8 that for the second experimental case, the *mean square error* decreases from 0.141 to 0.005 for $P=1$ to $P=8$ process realizations. From the experimentation conducted for the multiple process realizations of NHPPs, it can be concluded that the estimation of the mean-value function improves with increasing number of process realizations.

Table 5.7 Second Set of Statistical Performance Measures for Multiple Realizations

	Case							
	1				2			
	Realizations for Case 1				Realizations for Case 2			
Measures	**1**	**3**	**8**	**15**	**1**	**3**	**5**	**8**
δ	10.376	6.625	3.418	1.337	14.765	7.762	5.452	4.106
V_{δ}	0.377	0.725	1.189	0.381	0.462	0.657	0.556	0.386
Q_{δ}	0.332	0.212	0.109	0.042	0.295	0.155	0.109	0.082
$\overline{\delta}^{*}$	51.012	32.195	15.708	5.607	52.996	28.292	19.010	13.472
V_{δ}^{*}	0.417	0.802	1.336	0.591	0.568	0.813	0.744	0.601
Q_{δ}^{*}	1.632	1.030	0.502	0.179	1.059	0.565	0.381	0.269
$\overline{\Delta}$	7.263	4.577	2.758	1.736	3.699	2.216	1.693	1.324
V_{Δ}	0.503	0.472	0.596	0.610	0.620	0.581	0.641	0.645
Q_{Δ}	0.434	0.273	0.164	0.103	0.174	0.104	0.079	0.062
$\overline{\Delta}^{*}$	14.975	9.769	5.638	3.150	7.913	4.485	3.415	2.562

Table 5.8 First Set of Statistical Performance Measures for Multiple Realizations

	Case							
	1				2			
	Realizations for Case 1				Realizations for Case 2			
Measures	**1**	**3**	**8**	**15**	**1**	**3**	**5**	**8**
SS_E	51.965	70.119	85.884	5.538	7.001	4.731	2.847	2.323
V_{SS_E}	0.706	1.321	2.287	0.522	1.177	1.952	2.711	0.661
MS_E	0.419	0.186	0.085	0.002	0.141	0.031	0.011	0.005
V_{MS_E}	0.697	1.317	2.281	0.515	1.142	1.901	2.557	0.651
D	4.644	2.292	1.187	0.493	2.149	0.941	0.598	0.503
$\overline{D}^{*}$	12.294	6.834	3.944	1.825	4.982	2.703	1.951	1.727
S_D	2.211	3.568	2.019	0.003	2.3	0.706	0.141	0.009
V_D	1.487	1.889	1.421	0.055	1.516	0.841	0.376	0.095

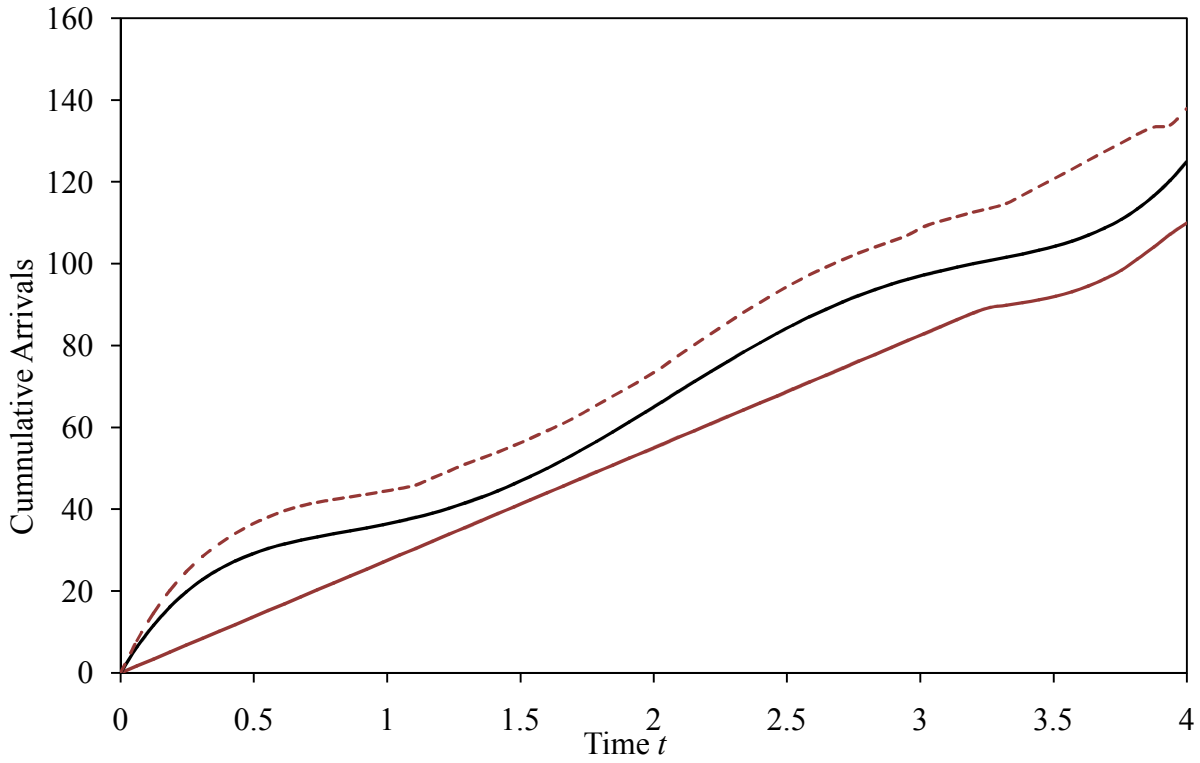

Figure 5.14 90% Tolerance Intervals for μ(t),t ∈ [0,4] in Case 1, 3 Realizations

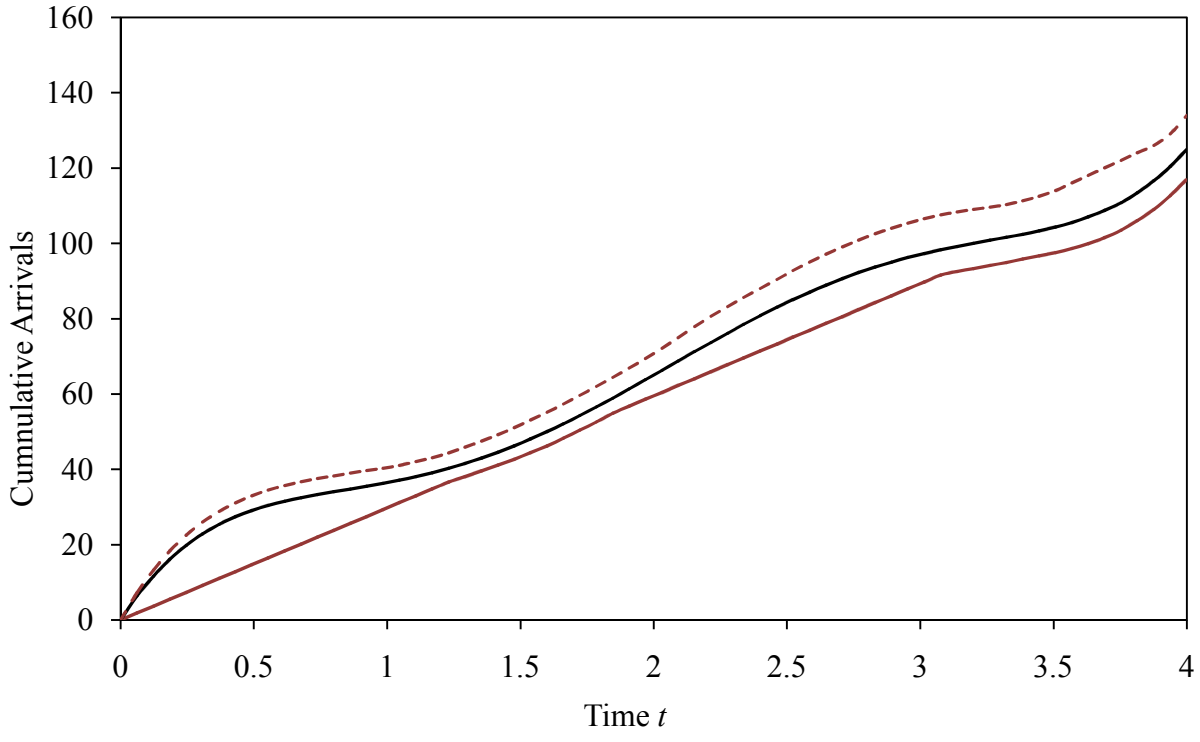

Figure 5.15 90% Tolerance Intervals for μ(t),t ∈ [0,4] in Case 1, 8 Realizations

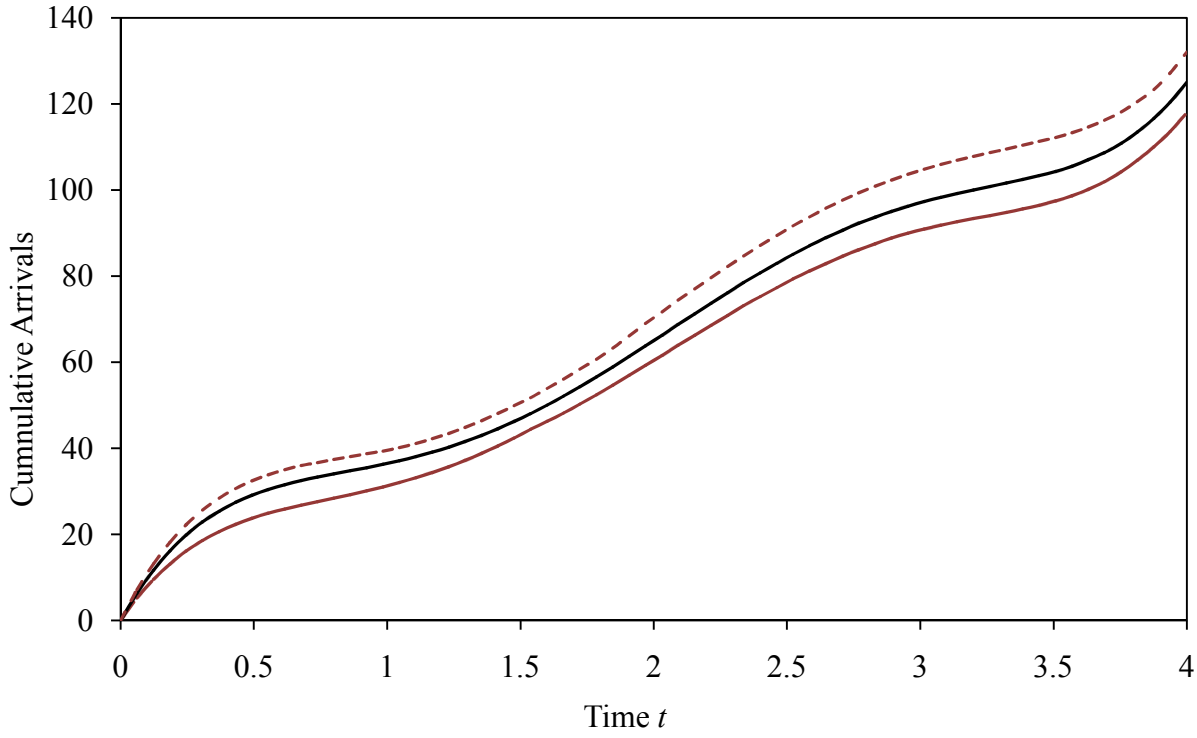

Figure 5.16 90% Tolerance Intervals for μ(t),t ∈ [0,4] in Case 1, 15 Realizations

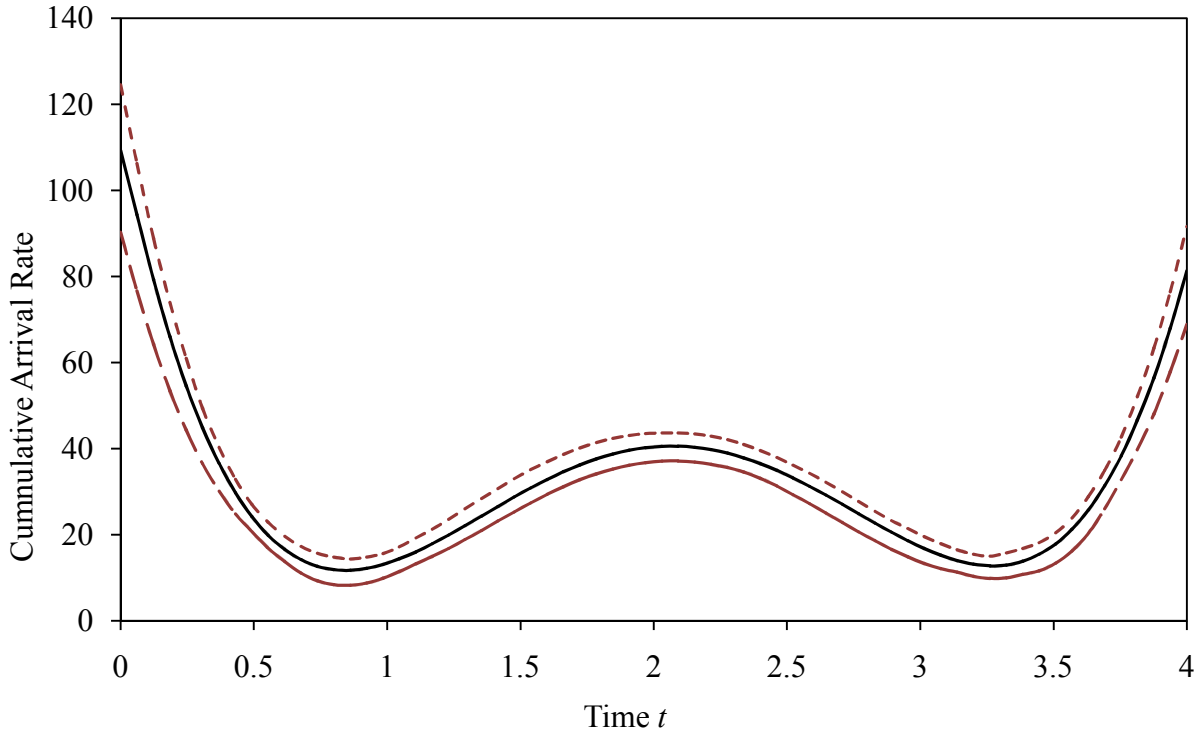

Figure 5.17 90% Tolerance Intervals for λ(t),t ∈ [0,4] in Case 1, 15 Realizations

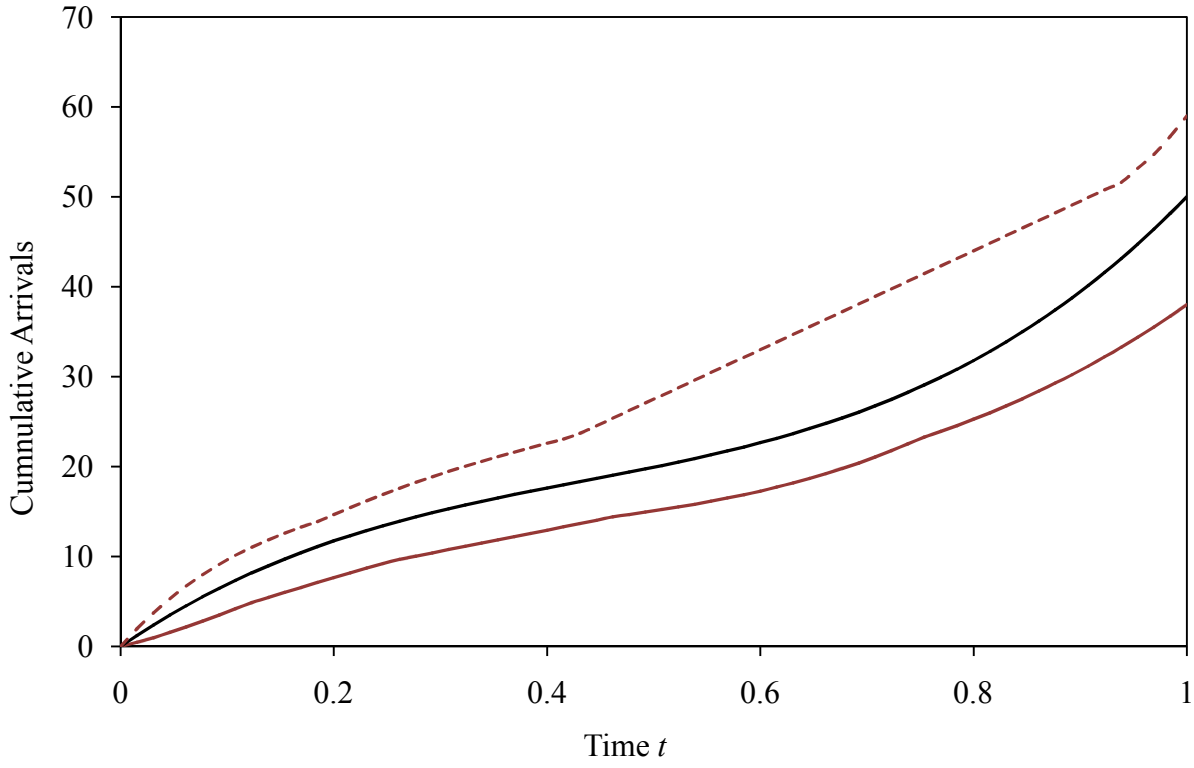

Figure 5.18 90% Tolerance Intervals for μ(t),t ∈ [0,1] in Case 2, 3 Realizations

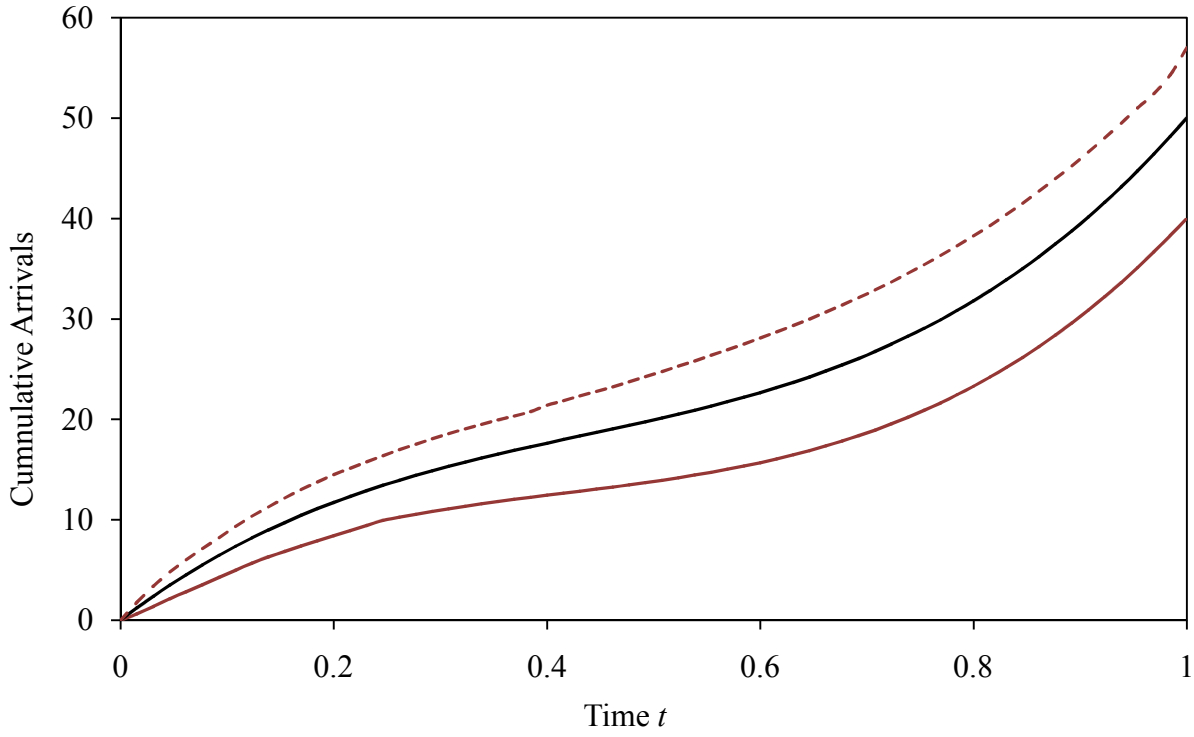

Figure 5.19 90% Tolerance Intervals for μ(t),t ∈ [0,1] in Case 2, 5 Realizations

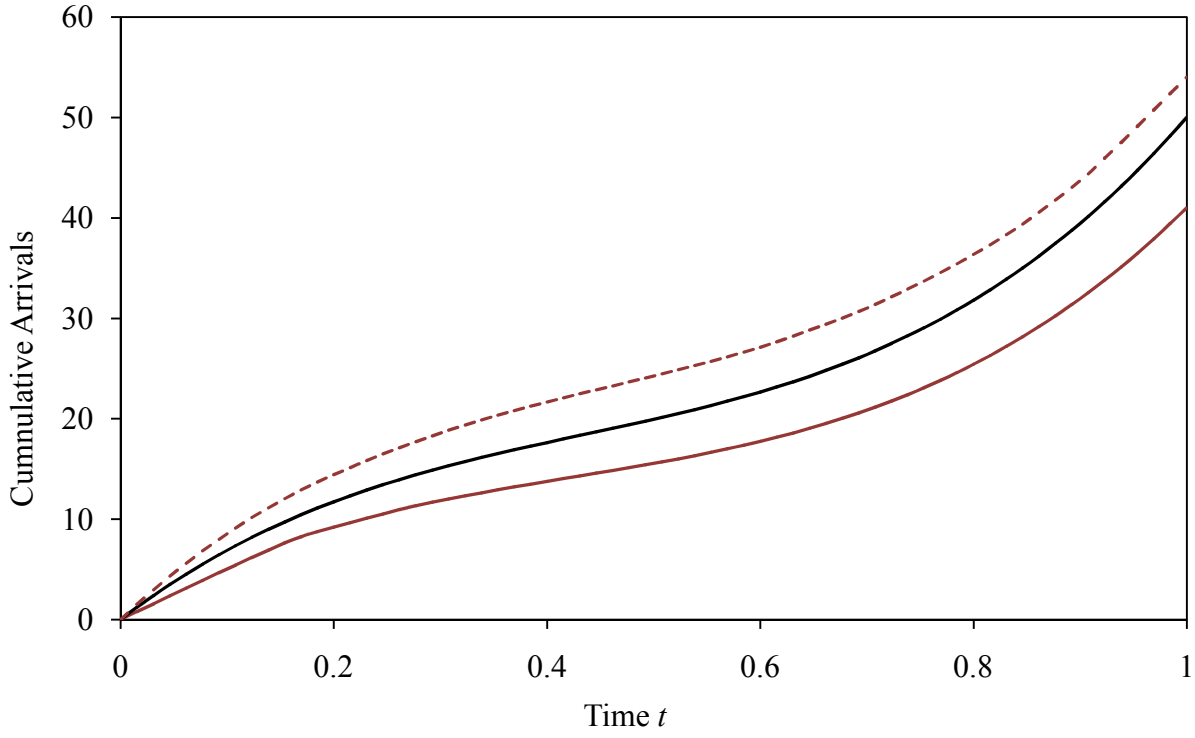

Figure 5.20 90% Tolerance Intervals for μ(t),t ∈ [0,1] in Case 2, 8 Realizations

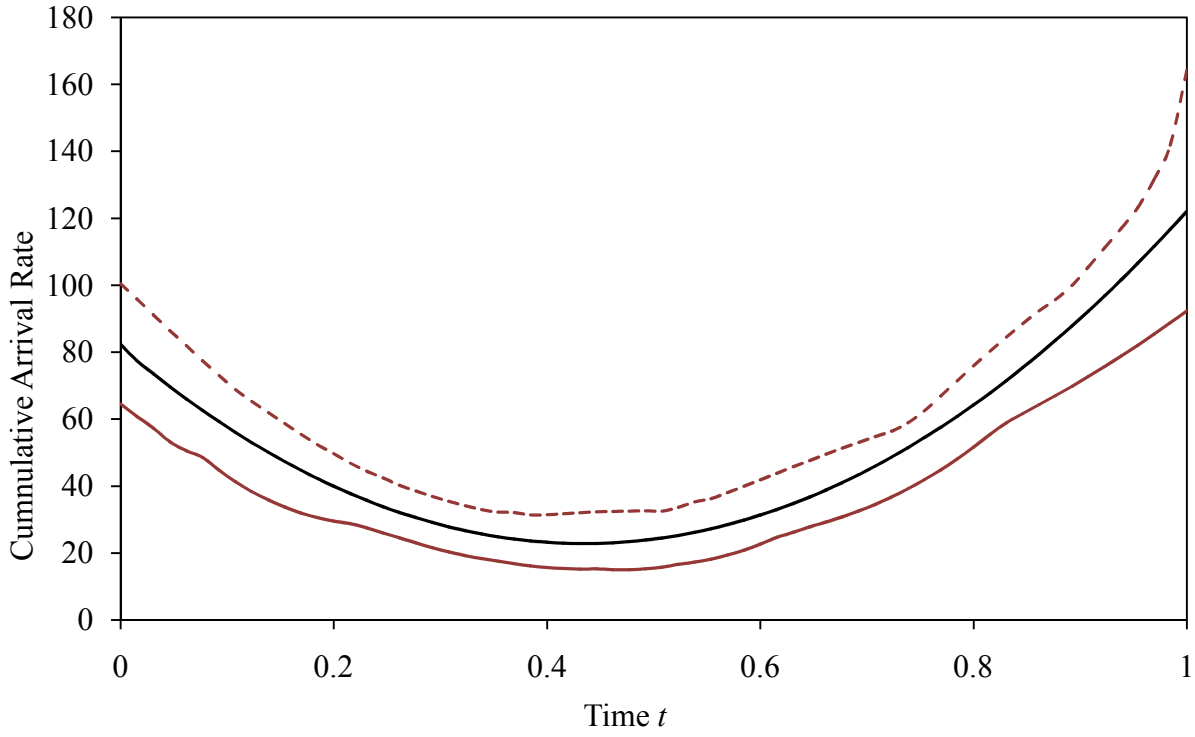

Figure 5.21 90% Tolerance Intervals for λ(t),t ∈ [0,1] in Case 2, 8 Realizations

6. MultiResolution Procedure to Fit an NHPP to Multiple Process Realizations

Advantages of modeling mean value function from multiple realizations for an acyclic NHPP is seen in Section 5.2. The same advantage can be extended to cyclic NHPPs where data observed for a single realization is inadequate. If data from multiple process realizations of an NHPP is observed, then better estimates for the mean-value function can be achieved by fitting the function to all realizations rather than a single process realization. The flexibility of modeling the mean-value function from single or multiple realizations is achieved here by adapting the multiresolution procedure developed by Kuhl et al. (2006).

6.1 Background

Kuhl et al. (2006) devise an automated multiresolution procedure to model nonhomogeneous Poisson processes with nested cycles and long term trend. A component cycle in the NHPP is referred as a *resolution*; hence, the name multiresolution. The multiresolution procedure estimates the mean-value function using the cumulative arrivals at each resolution. The literature review discusses the multiresolution procedure in detail. Recall that at each resolution, a polynomial function of form:

$$\hat{R}_i(s) \equiv \begin{cases} s/b_i & \text{if } r = 1 \\ \sum_{k=1}^{r-1} \beta_k (s/b_i)^k + \left(1 - \sum_{k=1}^{r-1} \beta_k\right)(s/b_i)^r & \text{if r} > 1 \end{cases} \quad (5)$$

is fitted to cumulative data points. The original procedure uses arrivals from single NHPP and divides the process into its component cycles. The accuracy of the estimated mean-value function depends on the amount of data available. To improve the fit for sparse datasets, the multiresolution procedure is extended to utilize multiple realizations. The mean-value function is

estimated after combining arrivals from all realizations. It is assumed that multiple realizations represent processes drawn from same population; meaning all of the observed NHPP realizations have the same cyclic components and the same duration length. A detailed discussion of this method is conducted in the section below. An example is included to illustrate advantages of numerous realizations. The statistical performance measures are estimated at different levels of realization to understand the effects of more than single realization on the resultant fitted mean-value functions.

6.2 Multiresolution Methodology

The multiresolution procedure developed by Kuhl et al. (2006) estimated the mean-value function of NHPPs with nested cycles and long term trend. This research adapts the multiresolution procedure to allow for multiple process realizations, $P \geq 2$. The arrival times are denoted as $\left\{t_{i,\ell} : i = 1, ..., N_\ell(S)\right\}$ for ℓ^{th} and $N_\ell(S)$ is the total number of arrivals in ℓ^{th} realization. The arrivals from all realizations are combined together to obtain a arrival set of m arrivals where:

$$m = \sum_{\ell=1}^{P} N_\ell(S),$$

over observation interval $(0, S]$ and average number of arrivals over $P \geq 2$ realizations is:

$$\bar{N}(S) = \frac{1}{P} \sum_{\ell=1}^{P} N_\ell(S).$$

The combined arrivals are then arranged in ascending order, such that $t_{(1)} \leq t_{(2)} \leq ... \leq t_{(m)}$. The overlapped arrivals from multiple realizations are then distributed over different resolutions; refer multiresolution procedure by Kuhl et al. (2003). The cumulative arrivals in each resolution

LRT-Based Multi-Resolution Estimation Procedure

[0] Initialize the resolution index $i \leftarrow 0$.

[1] If $i > p$ then deliver the fitted mean-value function $\hat{\mu}(t) = \hat{\mu}(t; \tilde{\beta}_0, ..., \tilde{\beta}_p)$ using the estimated regression coefficients $\{\tilde{\beta}_0, ..., \tilde{\beta}_p\}$ then stopping. If $i \le p$, then take the composite of the arc- sine and square-root transformations of the original arrival data within the basic resolution-*i* cycle $[0, b_i]$.

[2] Test the adequacy of the degree-1 polynomial $\hat{R}_i(\bullet)$ as an estimator of $R_i(\bullet)$.

(*a*) If $\text{SSE}_1/\text{SST} < 0.01$ or $m_i < 2$, then set $r \leftarrow 1$ and $\tilde{\beta}_i \leftarrow 1$ for the original (untransformed) responses at resolution i*;* set $i \leftarrow i+1$; and go to [1].

(*b*) If $\text{SSE}_1/\text{SST} \ge 0.01$ and $m_i \ge 2$, then set $r \leftarrow 2$ and go to [3].

[3] Compute the OLS estimator $\tilde{\text{C}}_r$ of the coefficient vector C_r for the transformed responses at resolution *i*.

(a) Use starting value $\hat{\text{C}}_r = \left[\tilde{\text{C}}_{r-1}, 0\right]^{\text{T}}$ to obtain $\tilde{\text{C}}_r$; for example, to compute $\tilde{\text{C}}_2$, start with $\hat{\text{C}}_2 = \left[\tilde{\text{C}}_1, 0\right]^{\text{T}} = \left[\pi/2, 0\right]^{\text{T}}$.

(b) Compute $\tilde{\text{C}}_r$ using least square estimates

(c) Compute maximum likelihood estimate for $\tilde{\sigma}_r^2 = \text{SSE}_r / m_i$

[4] If $-m \ln\left(\frac{\sigma_2^2}{\sigma_1^2}\right) \le \chi_{1-\alpha}^2$ then

(4a) and $-m \ln\left(\frac{\sigma_3^2}{\sigma_2^2}\right) \le \chi_{1-\alpha}^2$ then deliver liner fit $r = 1$.

(4b) and $-m \ln\left(\frac{\sigma_3^2}{\sigma_2^2}\right) > \chi_{1-\alpha}^2$, then reject linear fit and continue to next r.

[5] Using the fitted degree $\tilde{r}$ compute the OLS fit to for the original (untransformed) data; and saving the resulting $\tilde{r}$-dimensional estimator $\tilde{\beta}_i$ of the vector of regression coefficients in estimator $R_i(\bullet)$. Set $i \leftarrow i+1$ and go to [1].

Figure 6.1 Modified Algorithm for Multiresolution Procedure

are transformed using arc-sine conversions. The degree r for the polynomial function is estimated for the transformed data using modified likelihood ratio test. After estimating the degree r of the polynomial, the vector $\tilde{\mathrm{B}}_r$ is estimated for the original data. The steps in estimation of mean-value function are shown in figure 6.1.

6.3 Illustrative Multiresolution Example

An example is considered for an NHPP with multiple nested cyclic components over an observation duration of $S = 100$ days and have $j = 2$ cyclic components. The period of the cyclic components are $j_1 = 10$ days and $j_2 = 1$ day with the average number of arrivals over each process realization, $N_\ell(S) = 100$. The polynomial coefficients of the target process for the three resolutions that used to generate realizations of the NHPP are shown in table 6.2.

Arrivals from multiple processes are sorted in ascending order. Arrival times in the resulting data set are segregated into different resolutions following the process detailed in Kuhl et al. (2006).

Table 6.1 Polynomial Coefficients for Multiresolution Procedure

		Degree		Input Coefficients				
Case	***S***	***r***	**Res**	β_1	β_2	β_3	β_4	β_5
1	28	1	0	0.01				
		2	1	0.281048	-0.1047879	0.02163275	-0.0018185	0.0000522
		4	2	3.5704805	-19.04968	50.161603	-55.057099	21.374703

To illustrate the fitting procedure applied to a multiple observed process realizations, graphs are plotted at each resolution for $P = 3$ realizations. The mean-value function is fit to

arrivals from 3 realizations with average of 150 arrivals over each realization. Estimated functions at various resolutions for transformed data and original data for 3 realizations are:

Transformed resolution 0:

$$\Gamma_0(s) = 0.0157079637\ s$$

Transformed resolution 1:

$$\Gamma_1(s) = 0.518102062\ s - 0.146161869\ s^2 + 0.0188308103\ s^3 - 0.0007824848\ s^4$$

Transformed resolution 2:

$$\Gamma_2(s) = 9.4605464158\ s - 52.185687518\ s^2 + 130.155920559\ s^3 - 141.4926654008\ s^4 + 55.632682314\ s^5$$

Estimated function for original arrivals is

Resolution 0:

$$R_0(s) = 0.01\ s$$

Resolution 1:

$$R_1(s) = 0.2237828878\ s - 0.0541579601\ s^2 + 0.0081181054\ s^3 - 0.000394014\ s^4$$

Resolution 2:

$$R_2(s) = 4.6059783424\ s - 25.836464026\ s^2 + 66.7573994979\ s^3 - 72.1112699394\ s^4 + 27.584356126\ s^5.$$

Figure 6.2 through 6.7 shows the fit for the transformed data and the cumulative fraction of arrivals at each resolution. Figure 6.8 represents the mean-value function plotted against the original cumulative arrivals for 3 realizations.

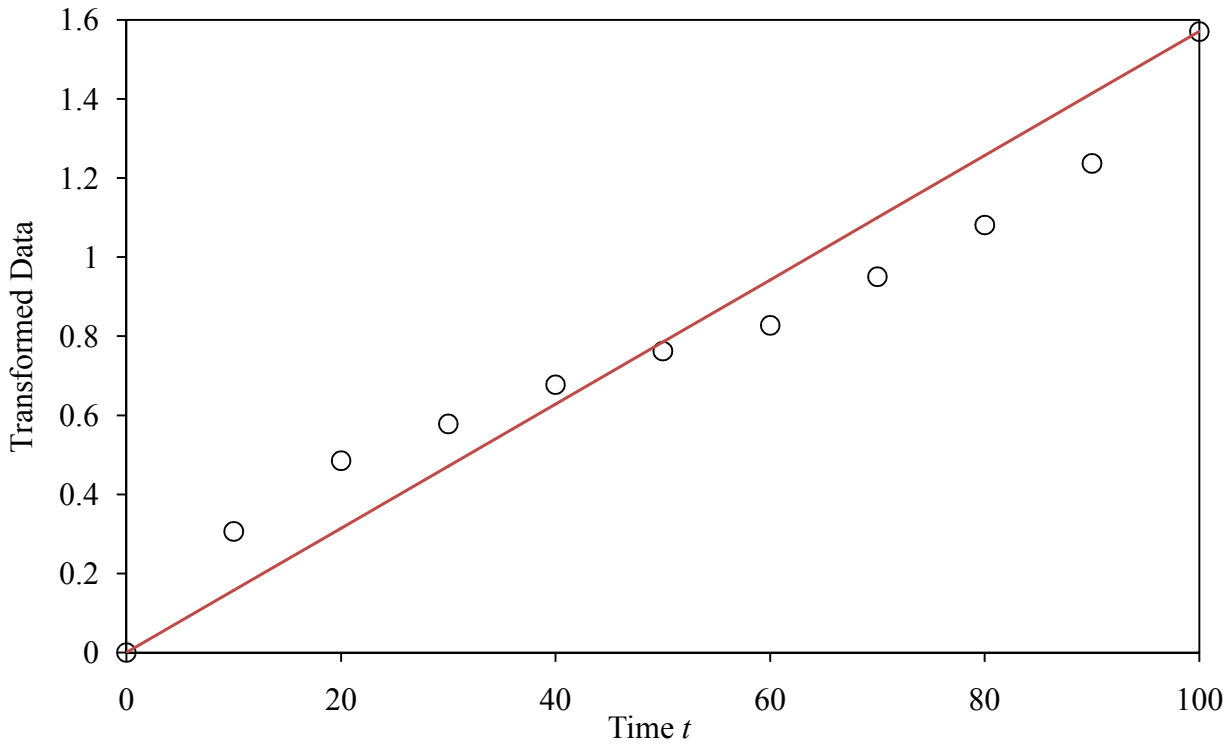

Figure 6.2 Transformed Function vs. Transformed Data at Resolution 0

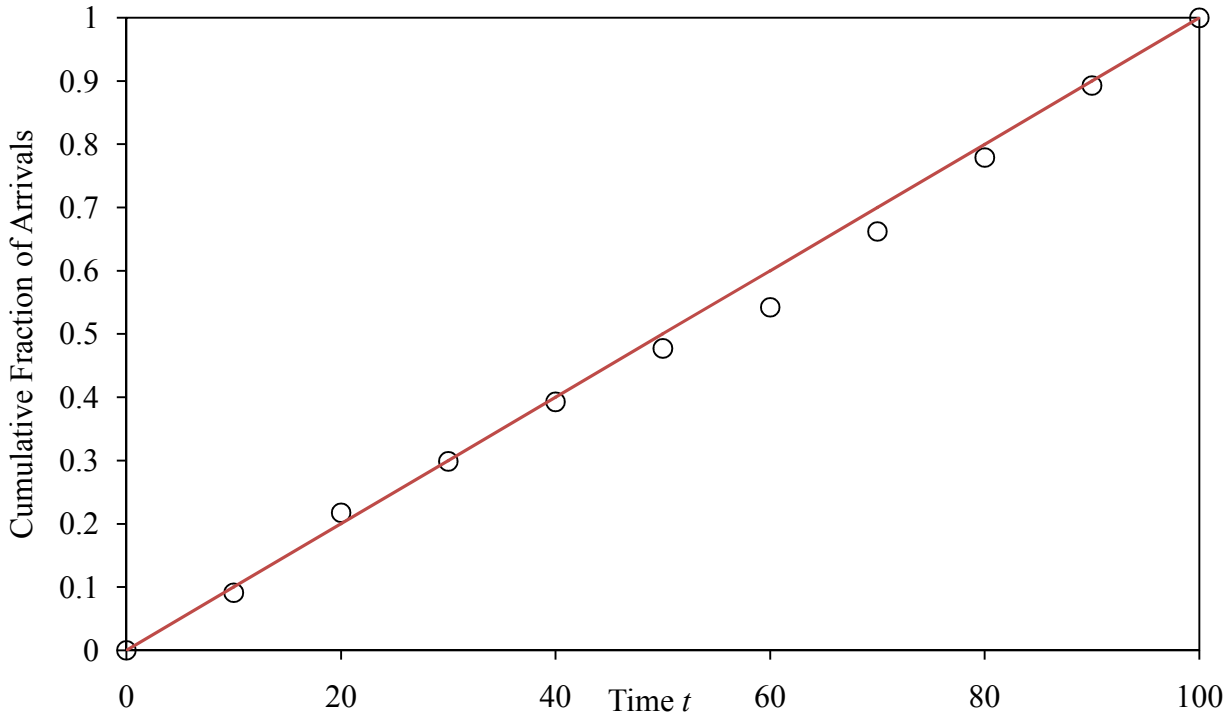

Figure 6.3 Fitted Function vs. Original Data at Resolution 0

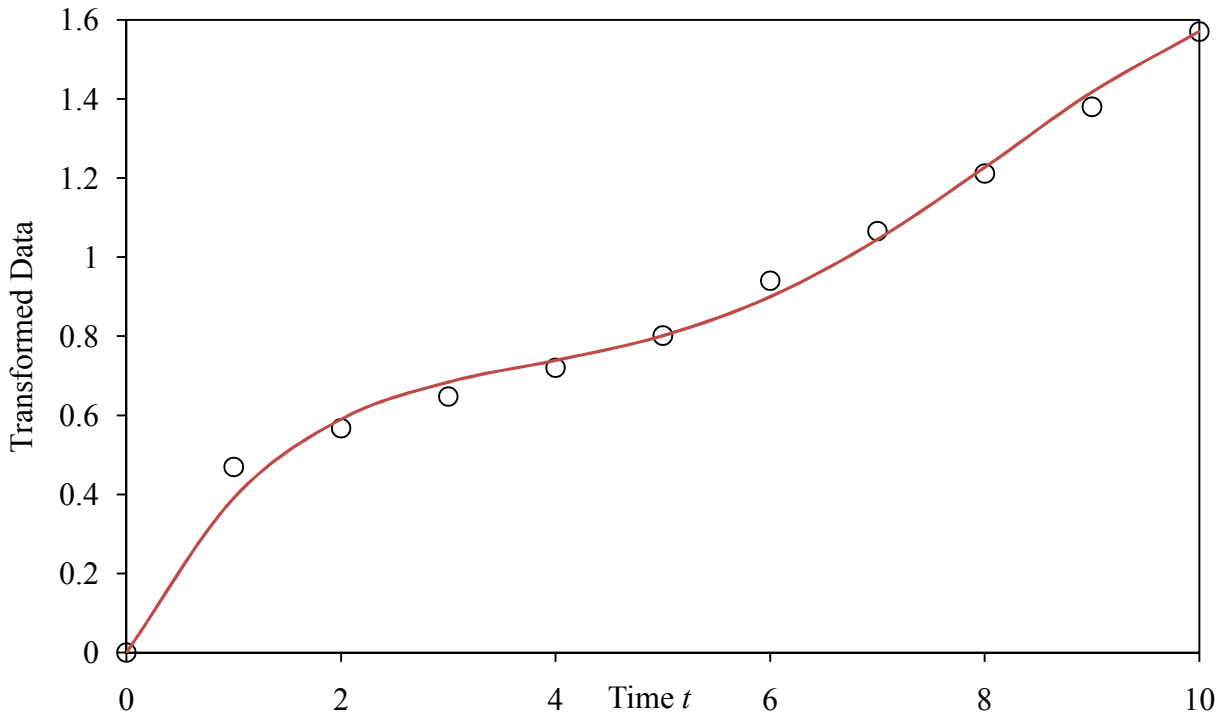

Figure 6.4 Transformed Function vs. Transformed Data at Resolution 1

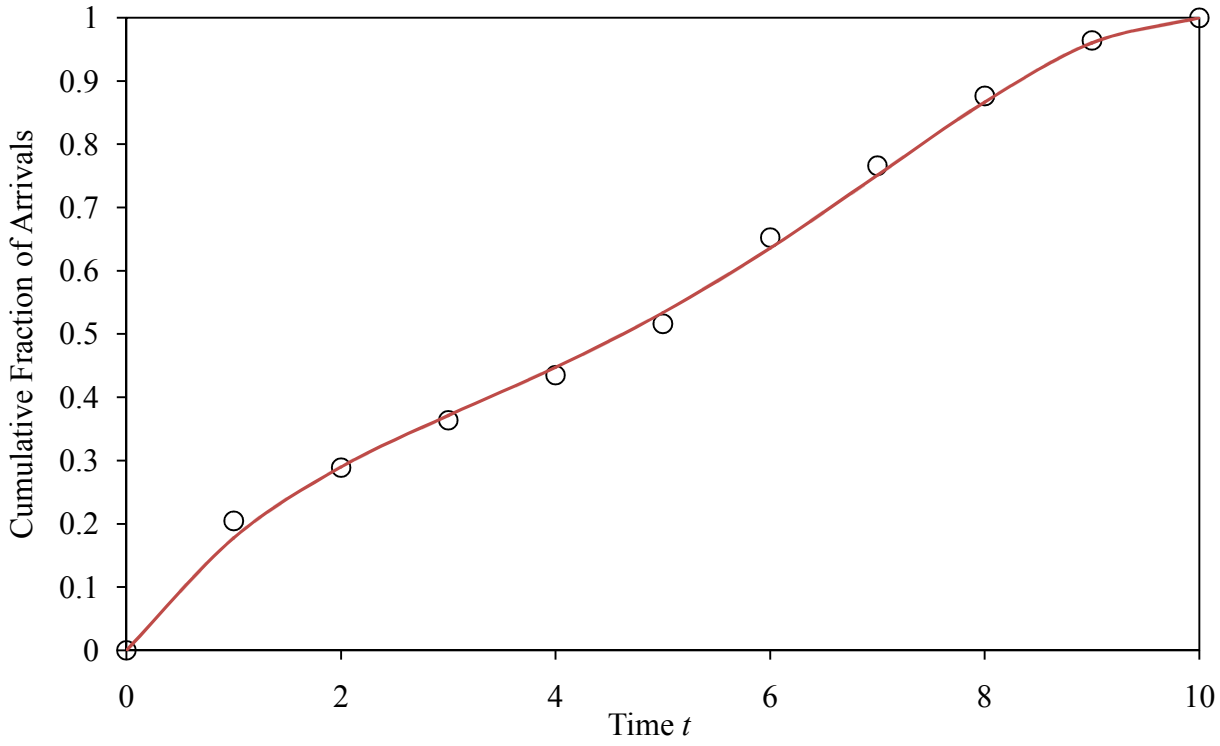

Figure 6.5 Fitted Function vs. Original Data at Resolution 1

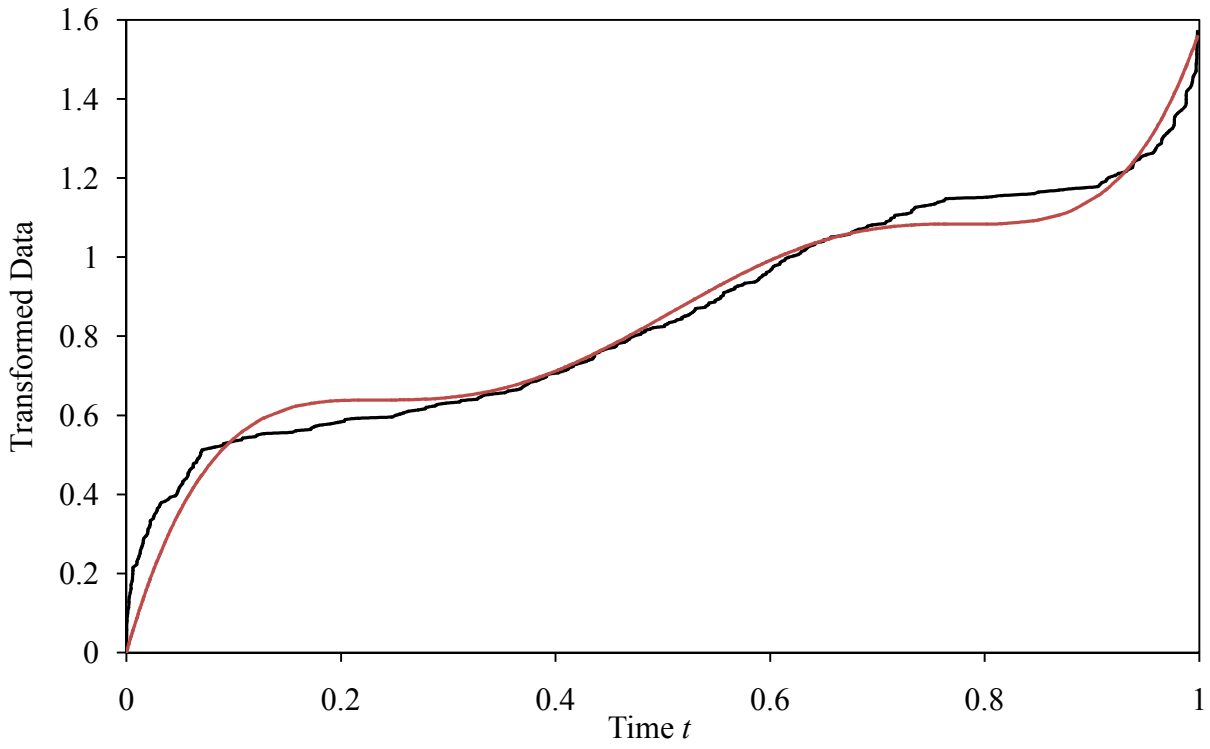

Figure 6.6 Transformed Function vs. Transformed Data at Resolution 2

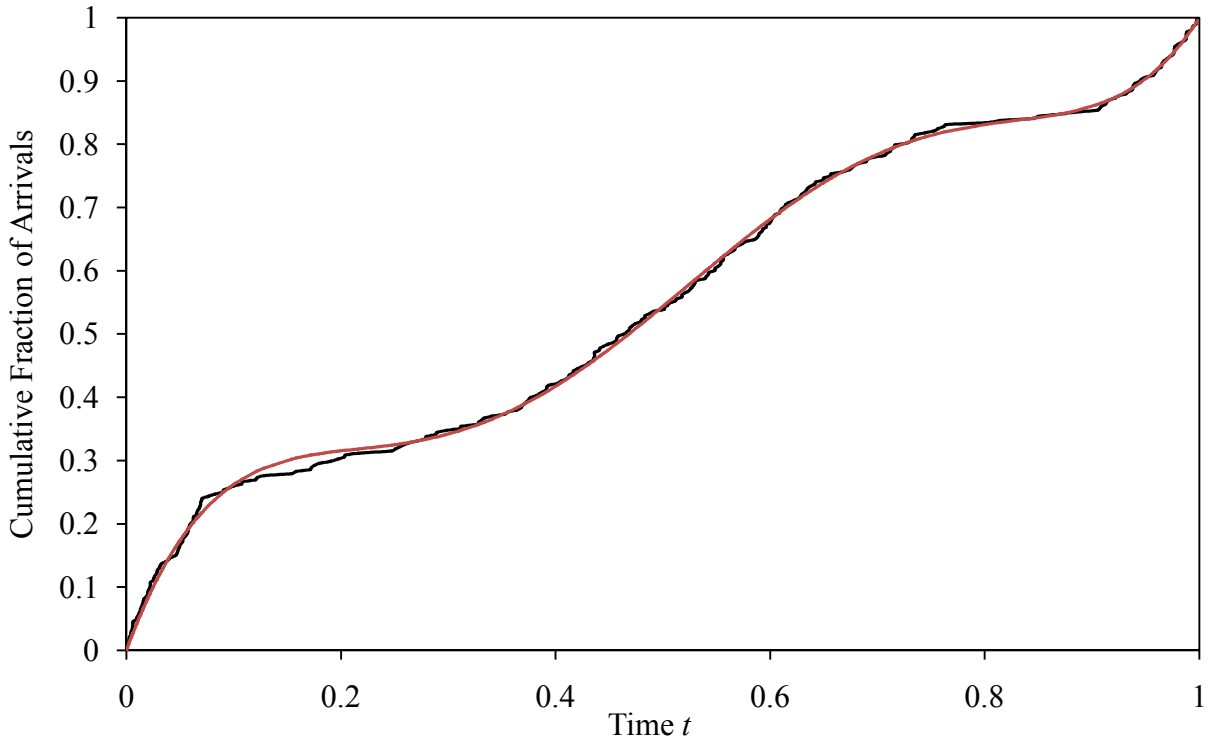

Figure 6.7 Fitted Function vs. Original Data at Resolution 2

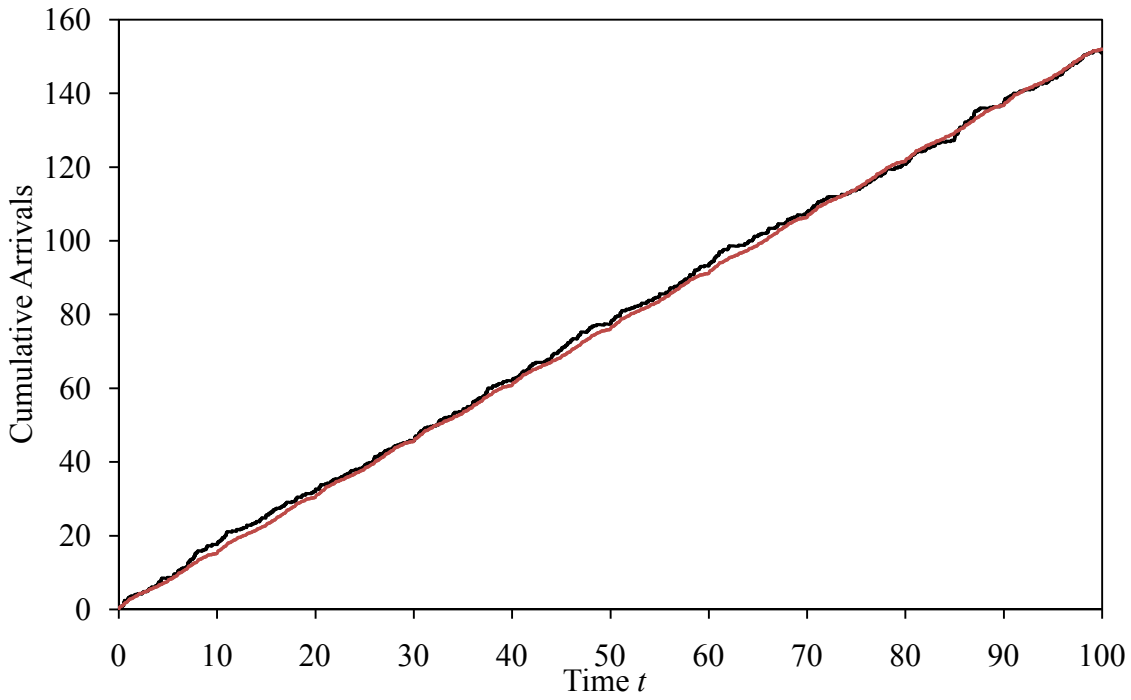

Figure 6.8 Estimated Mean Value Function for Original Data t ∈ [0,100]

6.4 Experimentation and Presentation of Results

The fitting procedure is executed for 100 replications for cyclic NHPPs with varying number of realizations. The experimental set up is similar to the one discussed in Section 5.1. The generation program is used to generate the arrival times for 100 replications while the fitting program determines the mean-value function at each resolution. Results from the experimental two cases are shown in the section below. Table 6.2 compares the fitted degree of the mean-value function with the actual polynomial function used to generate data. The statistical performance measures are estimated for multiple realizations and are listed in the table 6.3 and table 6.4 respectively. Graphs are plotted for 90% confidence interval for instantaneous rate function and mean value function from figure 6.9 to 6.20 for both the cases and various realizations to graphically demonstrate improvements in fitted function. The confidence intervals graphs for instantaneous rate function are plotted only for a single cycle meaning for duration of

7 for Case 1 and for duration of 10 for case 2. Shorter duration graphs are plotted to provide a detailed view of change in the rate function. Graphs for 90% confidence interval for the instantaneous rate function over entire duration for both the cases are included in Appendix C for the benefit of readers.

Table 6.2 Polynomial Coefficients for Multiresolution Procedure

		Degree		Input Coefficients				
Case	***S***	***r***	**Res**	β_1	β_2	β_3	β_4	β_5
1	28	1	0	0.01				
		2	1	0.258203	-0.0164780			
		4	2	1.83200	-0.890900	-0.967333	1.02624	
2	100	1	0	0.01				
		5	1	0.281048	-0.1047879	0.02163275	-0.0018185	0.0000522
		5	2	3.5704805	-19.04968	50.161603	-55.057099	21.374703

Table 6.3 Comparison Between the Original and Fitted Degrees for Varying Realizations

Case	*P*	*Res*	Original Degrees	Degree Fitted							
				1	2	3	4	5	6	7	8
1	1	0	1	100							
		1	2	11	78	11					
		2	4	15	7	4	71	3			
	3	0	1	100							
		1	2	2	80	16	2				
		2	4			15	75	2	8		
	8	0	1	100							
		1	2		83	5	12				
		2	4			11	78				
2	1	0	1	100							
		1	5	28			66	6			
		2	5	39		4	10	41	6		
	8	0	1	100							
		1	5	5			93	2			
		2	5	9				83	4	4	
	15	0	1	100							
		1	5				98	2			
		2	5					90	4	6	

Table 6.4 First Set of Statistical Performance Measures for Multiple Realizations

	Case					
	1			2		
	Realizations for Case 1			Realizations for Case 2		
Measures	**1**	**3**	**8**	**1**	**8**	**15**
SS_E	7.3885	7.6375	7.8562	21.734	18.335	22.587
V_{SS_E}	0.751	0.7523	0.7301	0.7792	0.7361	0.9077
MS_E	0.1244	0.0424	0.0165	0.1426	0.0152	0.0101
V_{MS_E}	0.7718	0.7479	0.7385	0.7792	0.7338	0.9094
$\overline{D}$	0.5817	0.1432	0.0465	0.6911	0.0426	0.0232
$\overline{D}^*$	6.2106	3.9419	2.6843	10.765	3.9677	3.2893
S_D	0.1412	0.0048	6.82E-4	0.2669	7.02E-4	1.73E-4
V_D	0.376	0.069	0.026	0.516	0.026	0.013

Table 6.5 Second Set of Statistical Performance Measures for Multiple Realizations

	Case					
	1			2		
	Realizations for Case 1			Realizations for Case 2		
Measures	**1**	**3**	**8**	**1**	**8**	**15**
δ	0.5863	0.3499	0.2297	0.5655	0.2461	0.1815
V_δ	0.3413	0.3405	0.2223	0.3389	0.5942	0.0416
Q_δ	0.2735	0.1633	0.1072	0.3771	0.1647	0.1209
$\overline{\delta}^*$	2.9530	1.7441	1.1583	4.2855	1.6631	1.1015
V_δ^*	0.4163	0.4503	0.4204	0.4532	0.7656	0.2406
Q_δ^*	1.3781	0.8134	0.5405	2.857	1.1087	0.7343
$\overline{\Delta}$	3.3112	2.0584	1.1711	5.0357	1.9295	1.2318
V_Δ	0.6714	0.6385	0.8453	0.7261	0.7351	0.6514
Q_Δ	0.1033	0.0642	0.0365	0.0666	0.0255	0.0163
$\overline{\Delta}^*$	6.1987	3.9439	2.2905	10.175	3.9855	2.5976

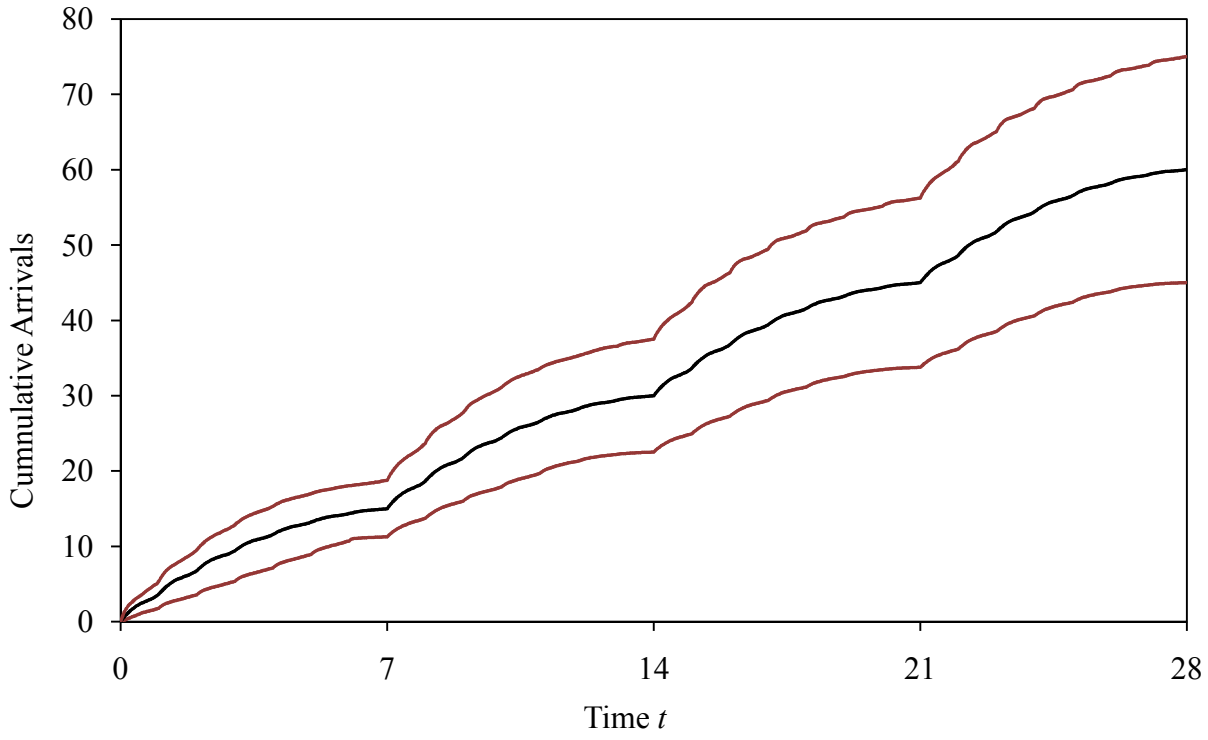

Figure 6.9 90% Tolerance Intervals for μ(t),t ∈ [0,28] in Case 1, Single Realization

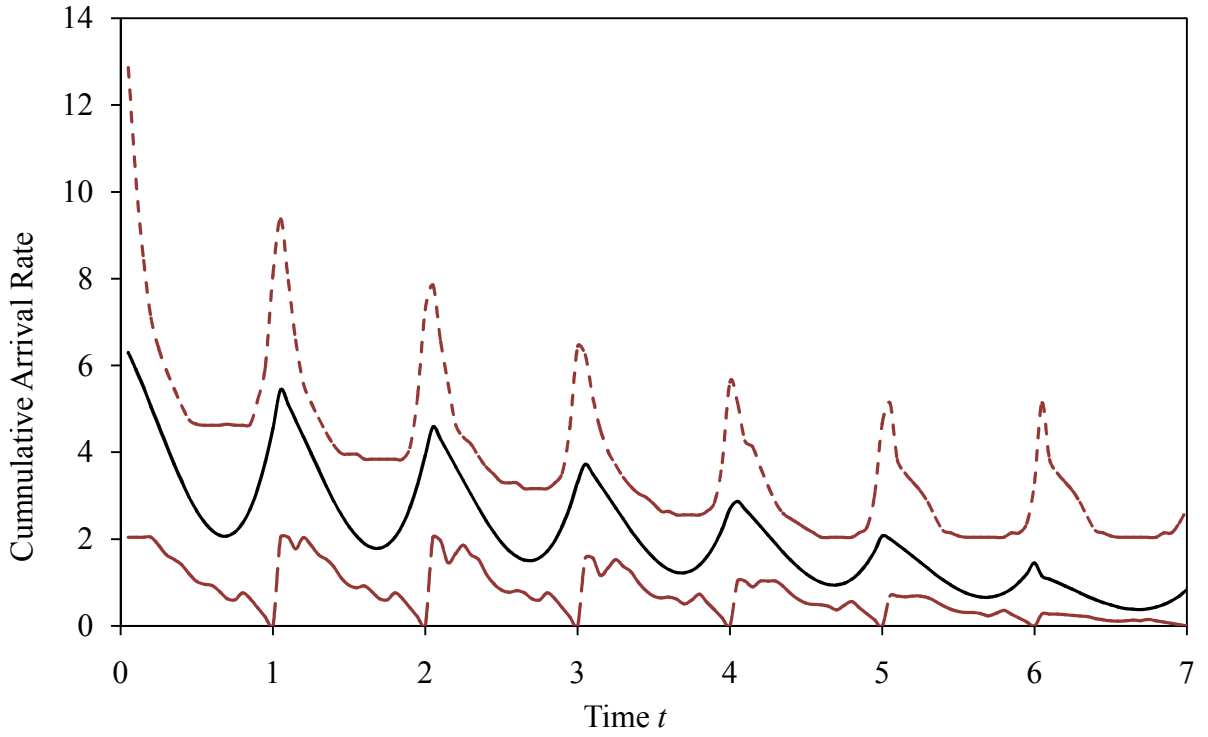

Figure 6.10 90% Tolerance Intervals for λ(t),t ∈ [0,28] in Case 1, Single Realization

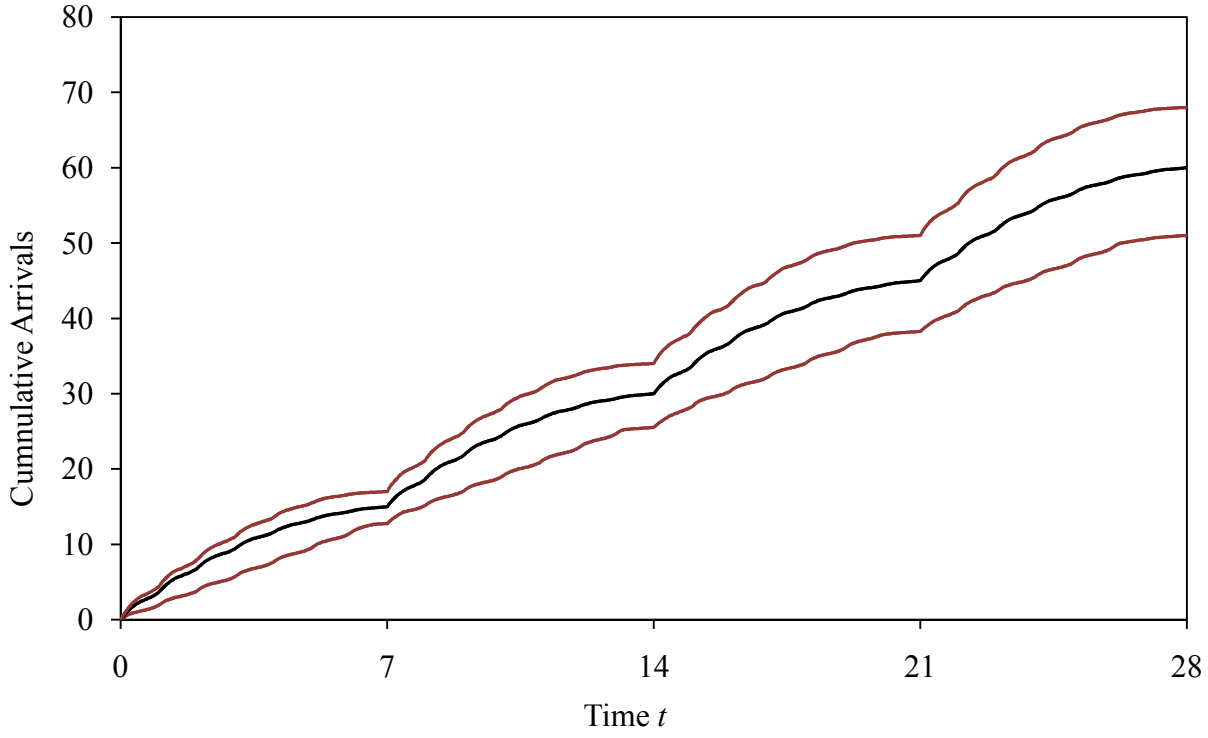

Figure 6.11 90% Tolerance Intervals for μ(t),t ∈ [0,28] in Case 1, 3 Realizations

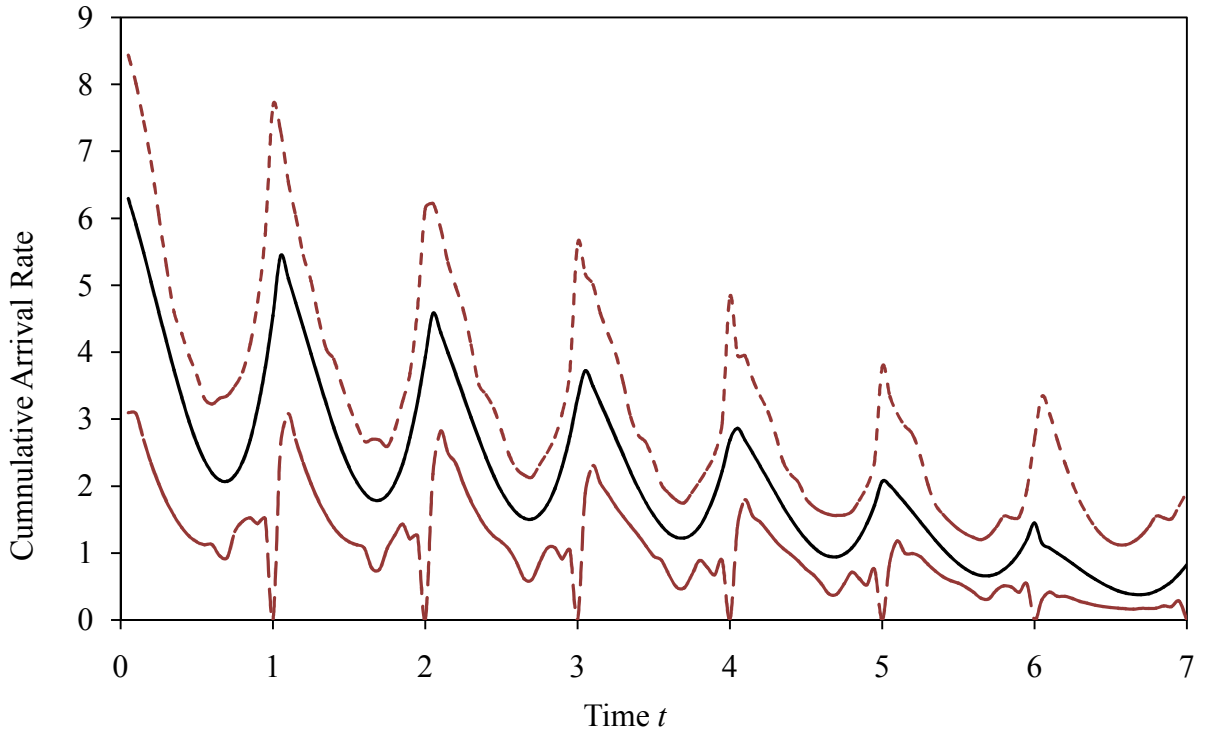

Figure 6.12 90% Tolerance Intervals for λ(t),t ∈ [0,28] in Case 1, 3 Realizations

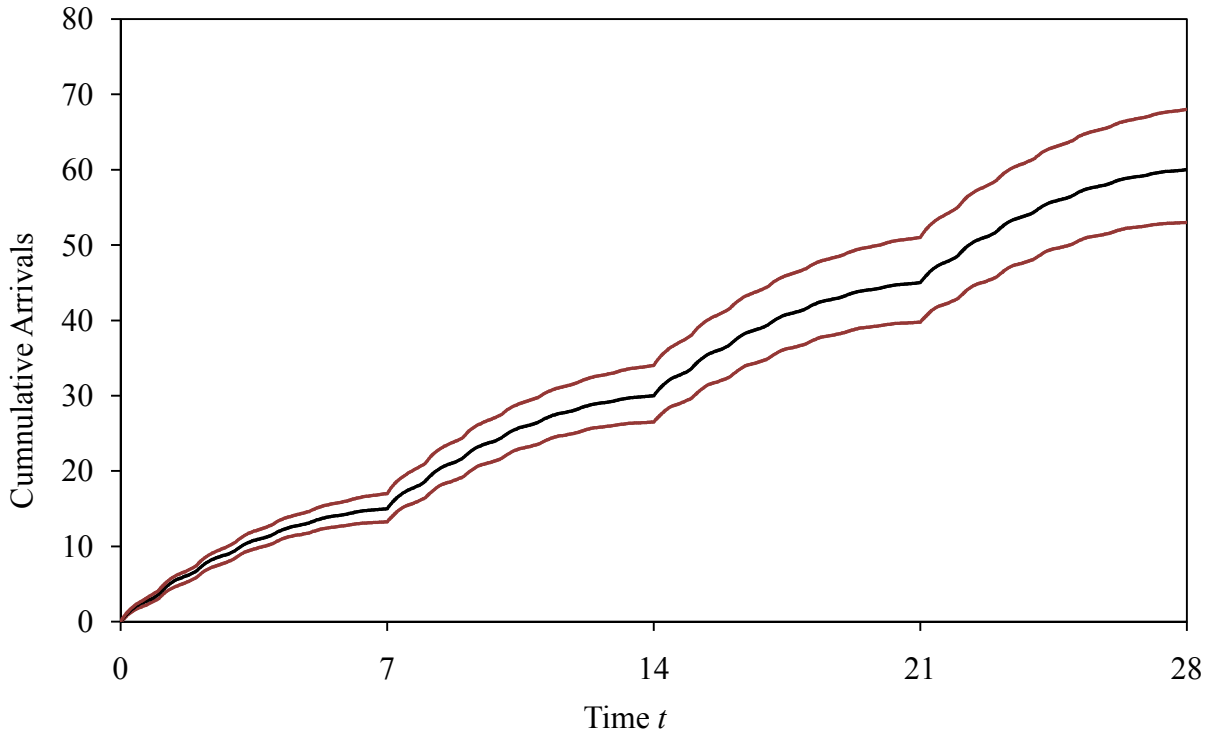

Figure 6.13 90% Tolerance Intervals for μ(t),t ∈ [0,28] in Case 1, 8 Realizations

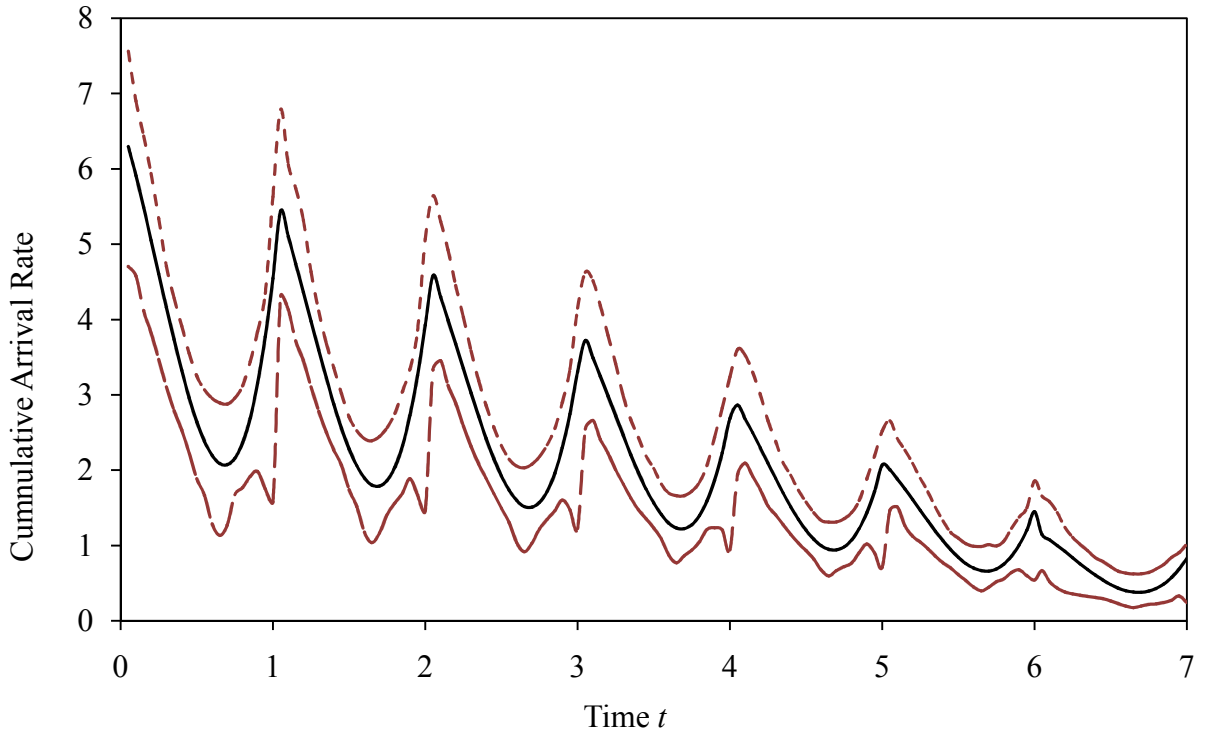

Figure 6.14 90% Tolerance Intervals for λ(t),t ∈ [0,28] in Case 1, 8 Realizations

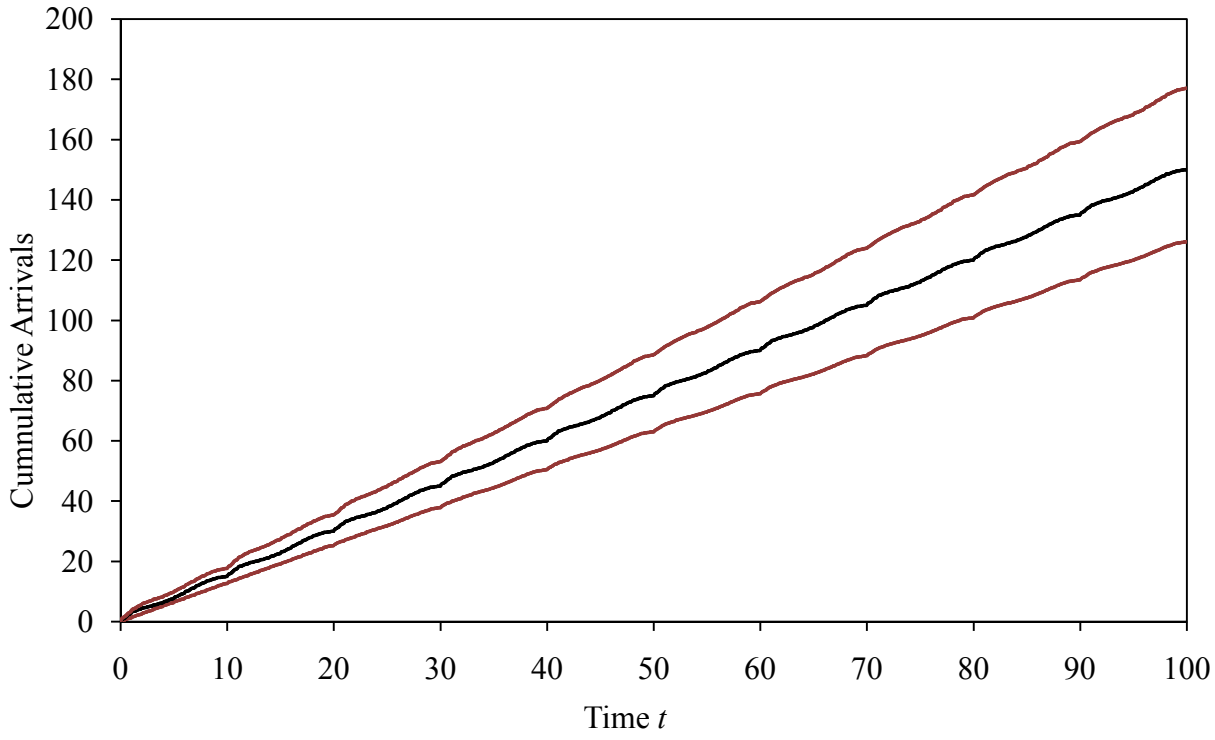

Figure 6.15 90% Tolerance Intervals for μ(t),t ∈ [0,100] in Case 2, Single Realization

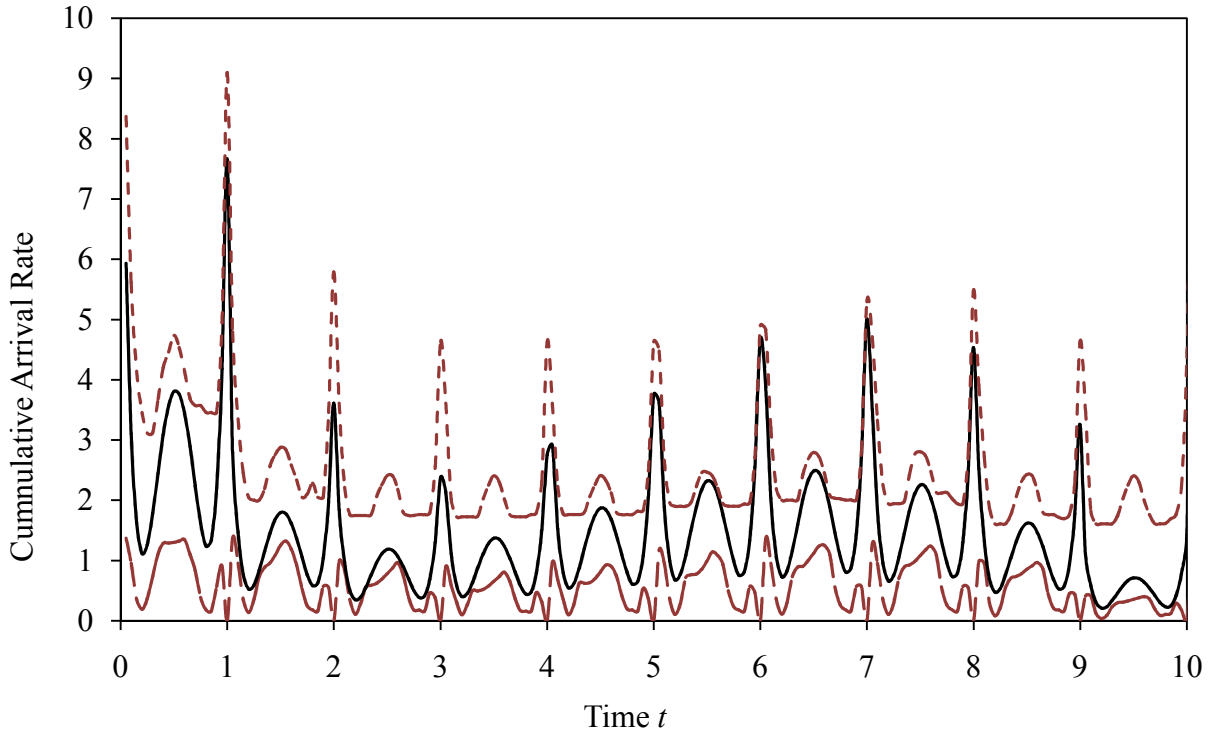

Figure 6.16 90% Tolerance Intervals for λ(t),t ∈ [0,100] in Case 2, Single Realization

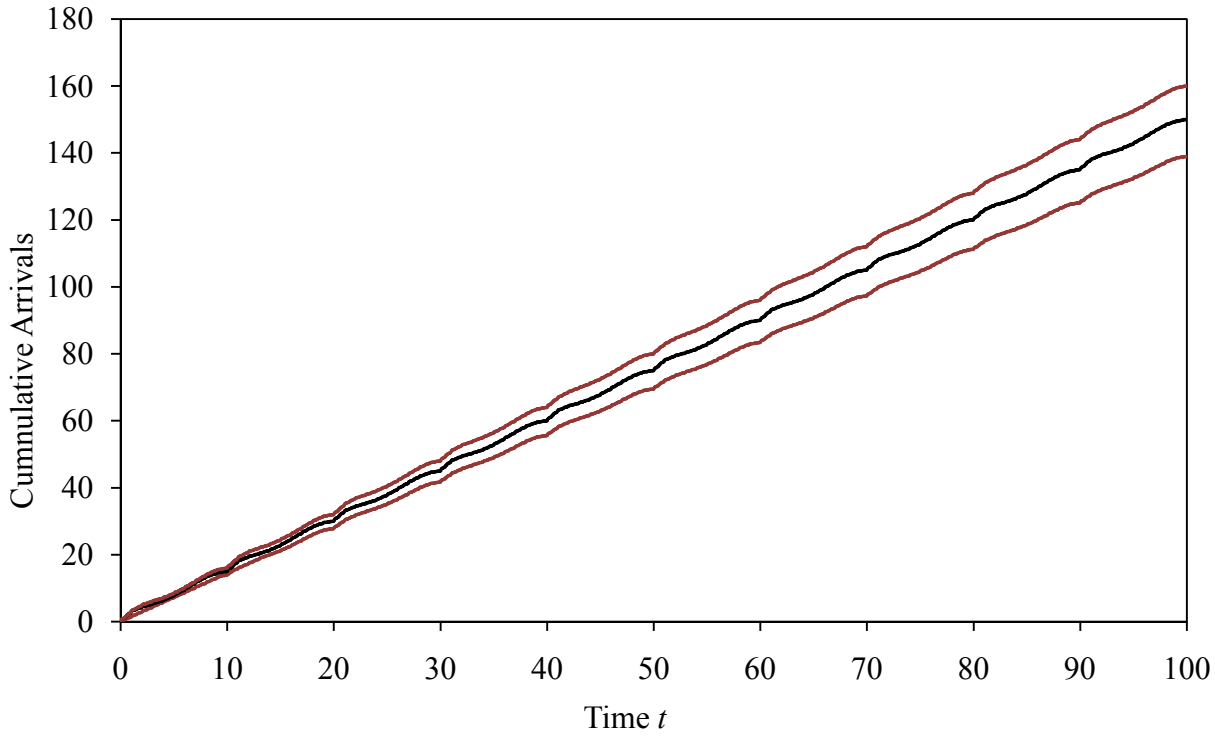

Figure 6.17 90% Tolerance Intervals for μ(t),t ∈ [0,100] in Case 2, 8 Realizations

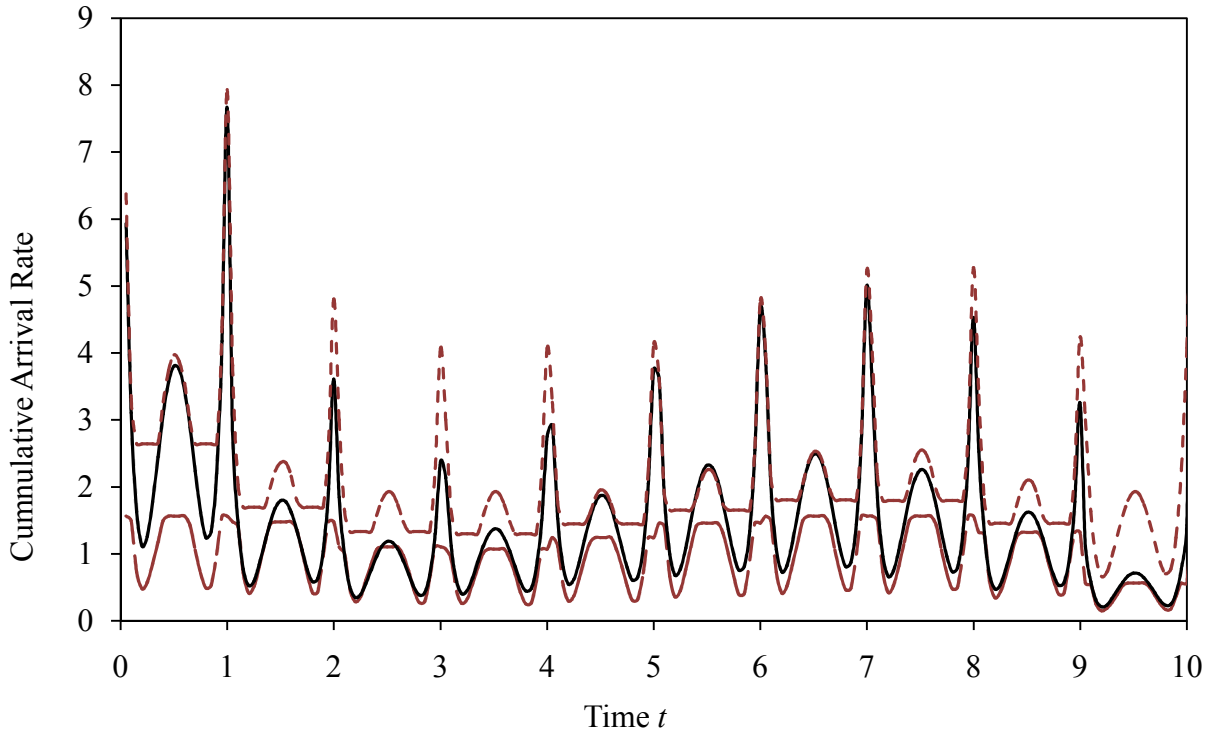

Figure 6.18 90% Tolerance Intervals for λ(t),t ∈ [0,100] in Case 2, 8 Realizations

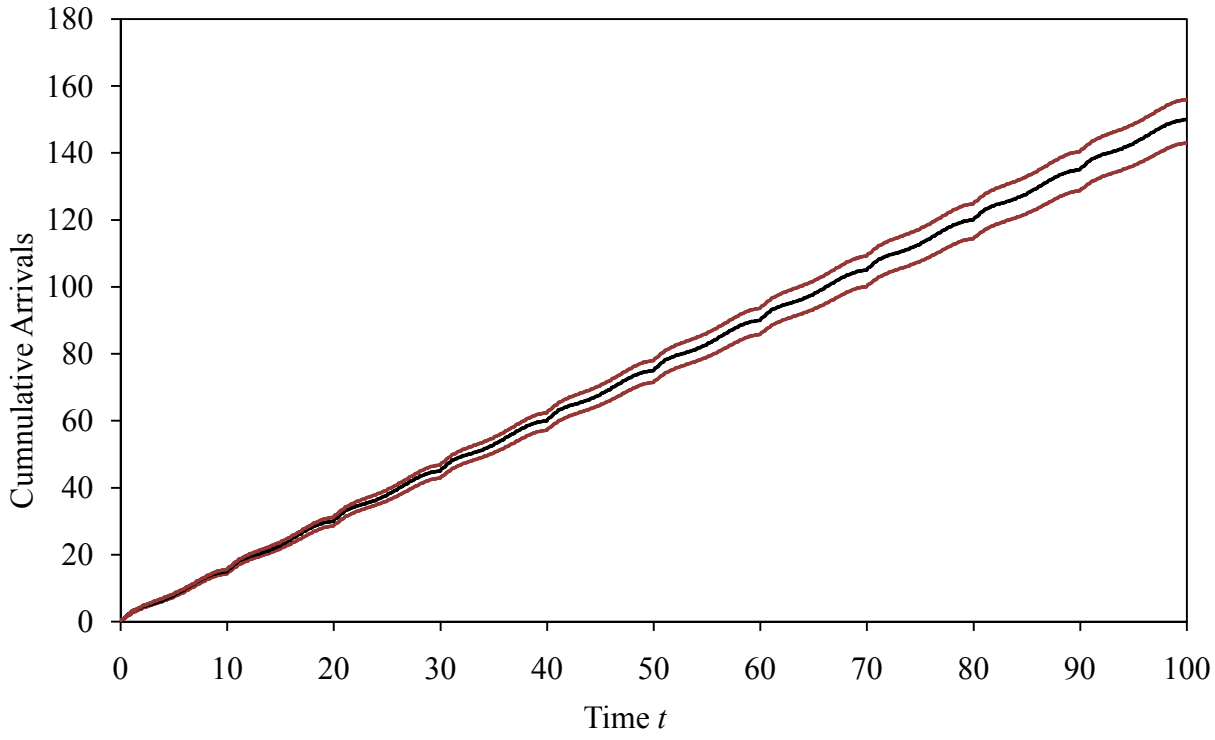

Figure 6.19 90% Tolerance Intervals for μ(t),t ∈ [0,100] in Case 2, 15 Realizations

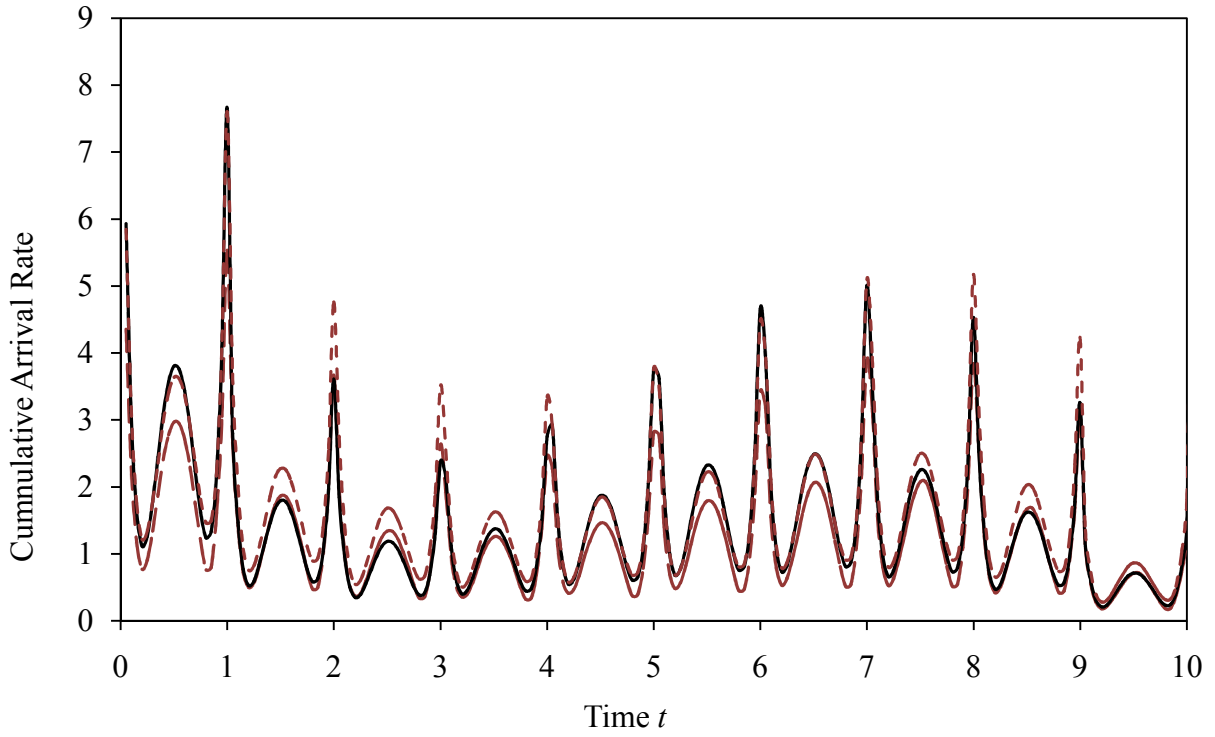

Figure 6.20 90% Tolerance Intervals for λ(t),t ∈ [0,100] in Case 2, 15 Realizations

6.5 Discussion of Results

Estimating the mean-value function of NHPP from multiple resolutions increases the accuracy of estimated function. As observed for multiple realizations for single resolution, it is seen that the error values diminish with an increasing number of realizations. For example, the mean square error in the first case reduces from 0.124 to 0.016 from single realization to 8 realizations. The trend of improved performance measure values with an increased number of realizations is recorded for both cases and for all measures. Also, plotted graphs improve with increased arrivals and distance between interval end points reduces.

7. Object-Oriented Implementation

The Object-oriented programming (OOP) is a powerful concept in software development due to the advantages such as modularity, data security, and reusability. The *object-oriented programming* is a method of programming based on a hierarchy of well-defined and cooperating classes. Java is considered a fully object-oriented programming language with the model for this research developed on the Java platform. The following discussion details the advantages and properties of the "Object Oriented Programming". From now on, the concepts of OOP are explained using Java because all the properties demonstrated by Java are typical properties of any OOP language.

7.1 Introduction to Java

Java models the real-world objects and its problems. In a real world, there are many objects of the same kind, for example, cars and trees. Java models things of the same kind as objects with their properties defined by a blueprint or a prototype called *classes*. In OOP, the objects are called as *instance of class*. The objects in the real world also have a *state* and a *behavior*. For example, a car can have a state such as make, color, model, and so on. Behavior of a car can be seen in terms of running, parked etc. Within Java, the state and the behavior of the objects are modeled as variables and methods, respectively and form the crux of the Java language. The basic properties of OOP are data encapsulation, inheritance, and polymorphism. *Data encapsulation* is the wrapping or binding of the data and the methods together in a structure known as *class*. The data is thus made invisible to the user and can only be accessed using the methods performing operation on data. Encapsulation means hiding the internal state of an object along with all the interactions with internal state being carried out using methods. Data encapsulation is a fundamental principle of the Java language and it is of great importance when

data security becomes a prime aspect of an application. Since the data cannot be directly modified in Java language like other procedural languages, data is considered secured. The *data encapsulation* also helps in adding modularity to the application as data sets are separated from each other and interact only through the methods.

The second powerful property exhibited by Java is called *inheritance* and is explained by the word itself. A child inherits properties of its parents; similarly in Java, a sub-class 'inherits', 'is derived from' or 'extends' the properties of its super class or the parent class. In the real world, we have dogs and cats of type mammal. Thus, the mammals become a super class defining the common properties of all the mammals and dogs inherit the properties of mammals. Since Java allows you to inherit properties of a super class, a great deal of code reusability is possible. When you want to create a new class requiring the code from an existing class, then the new class can simply be derived from the existing class. By extending super class, the methods and properties of a super class become available to a sub-class. In this way, the code from a super class is reused by all of its sub-classes, which save a lot of effort and time.

The third pillar of Java is *polymorphism*, which means different forms of the same entity. According to polymorphism, the same parameters are passed to different methods resulting in different behavior. Behavior is dependent on the receiver of the parameters. Polymorphism enables code reusability and imparts modularity. In Java, multiple methods can have the same name, but can accept different arguments. The execution of a particular method depends on the type of arguments passed in a method call.

The fitting method is implemented in Java language and uses the *inheritance*, *data encapsulation* and *polymorphism*. The features are used extensively to optimize program code and to reduce execution time by increasing reusability.

7.2 Hardware Requirements

The fitting method is implemented in Java programming language which needs Java platform for execution. The Java platform is a complete environment enabling software application development and deployment made available by the Sun Microsystems. It allows users to develop and deploy Java applications on the desktop systems and the servers. Java programs are executed within a program called *JVM*. Rather than running directly on the native operating system, the Java program is interpreted by the JVM for the native operating system as described by Nourie and Palwan (2007). The Java Standard Development Kit (SDK) is used for program development, which comes with a Java run-time environment and a JAVA Application Program Interface (API).

7.3 Program Architecture

The goal of this research is to automate the fitting method for NHPPs with and without cyclic components using a polynomial form of mean-value function. The research also develops a method to realize an NHPP from estimated mean-value function. This goal is achieved by writing a program in Java programming language, which receives input, processes the data, and delivers a fitted function as the output.

The aim of this program is to automate the fitting procedure detailed in the algorithm shown in figure 4.1. The fitting method estimates a mean-value function for NHPPs with and without periodic cycles. A *cyclic component* is referred to as *resolution* and same terminology persists in Java program. In the absence of any cyclic component, the entire NHPP process forms a single resolution. An application program is written for multiresolution and single resolution NHPPs. Two main tasks performed by this application are:

1) Fitting method to fit single or multiple resolutions in NHPPs.

2) Generation method to realize NHPP for single or multiple resolutions.

The program is structured so as the fitting and the generation procedures are carried at all the resolutions. This facilitates reuse of the code as same operations are executed for each resolution. The concepts of *polymorphism*, *inheritance*, and *data encapsulation* are exploited throughout implementation of the method. A software model is developed and designed in form of various java classes that perform specific tasks. The java classes are organized in a manner such that all the related java classes are stored in the same directories known as a *package*. The software model uses a total of six directories containing the classes required by the fitting and the generation method. Figure 7.1 details different packages and demonstrates the dependencies and the inheritance between different packages. The dependencies indicate if one package depends on one or more classes from the other packages. Various dependencies between packages are of form; *import*, *derived*, *call*, *implement*, and so on. Explanations about various dependency rules are present in the appendix B for reference. A brief discussion is included in this section to describe various classes, packages, along with the reason for their design. Some of the packages only contain interfaces, serving as a format of executing certain common functions. The interfaces are implemented by other classes where function-specific details are included. A better understanding of code design is achieved in the discussion below.

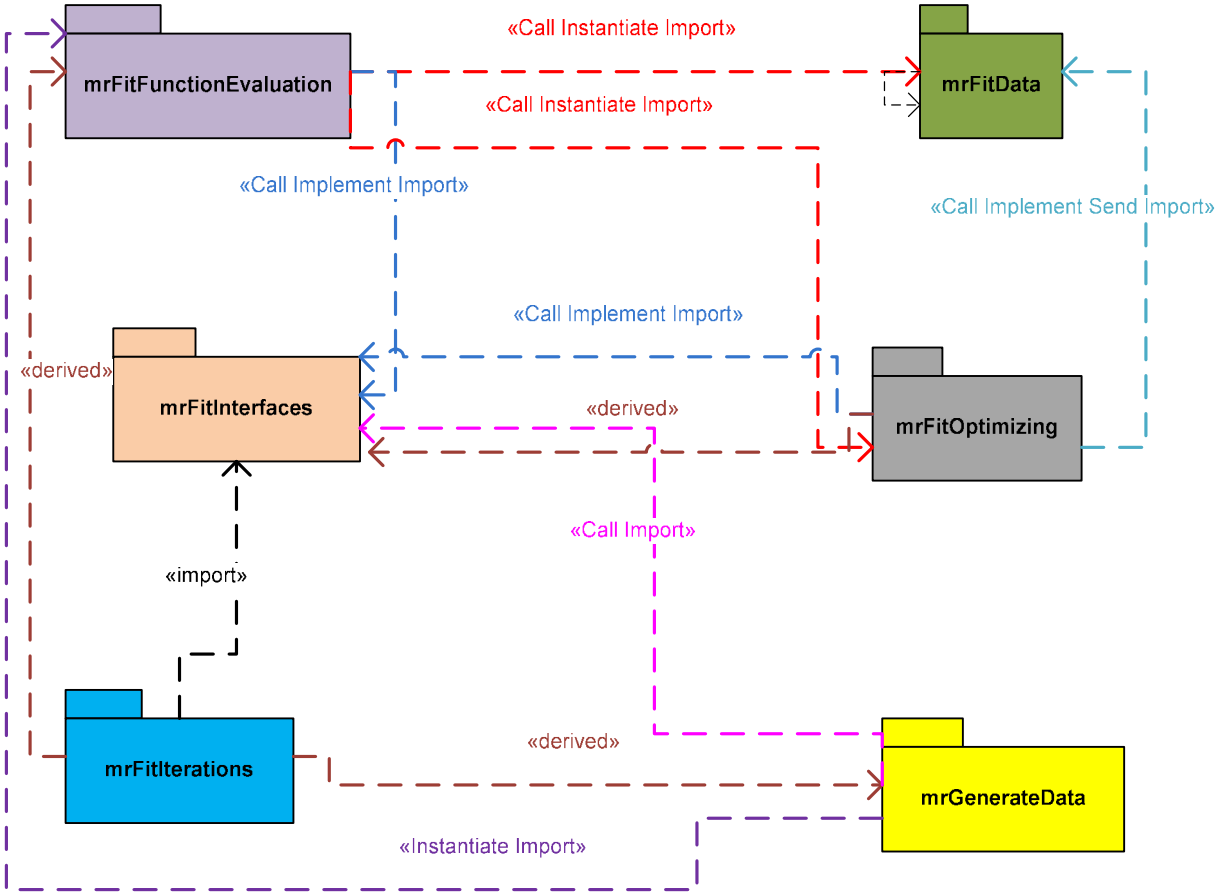

Figure 7.1 Package Diagram for Java Program

Package "mrFitData": As the name suggests, this directory contains the classes to store the various input and output parameters. These classes are referred to as *data object* from here on. The *data objects* are passed as arguments for various methods that operate on them. The sole purpose of their existence is to store data in form of the member variables whose values are altered by the methods. An attempt is made to decouple the data and the methods operating on the data to implement *data encapsulation* and provide data security. Apart from the member variables, the data objects have the *accessors* and the *mutators*, which facilitate initializing and extracting the values of the variables. In common terminology, they are called as the *getter and setter methods* of a data object.

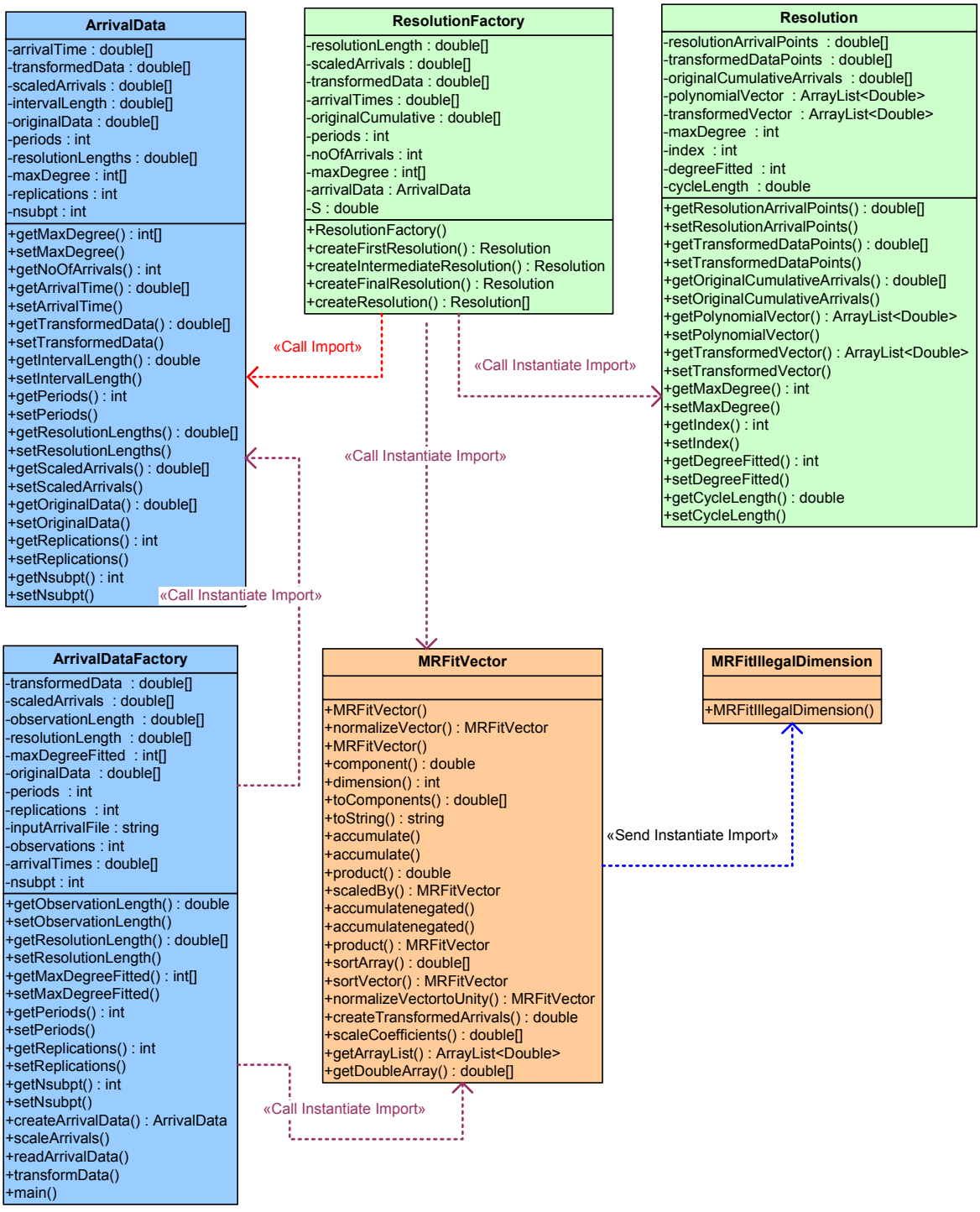

Figure 7.2 Class Diagram for "mrFitData"

The “ArrivalData” and “Resolution” are two data objects used as source of input for the application. The “ArrivalData” stores information regarding the entire NHPP realization while the “Resolution” object stores values pertaining to a single resolution. For example, the “ArrivalData” contains information such as the total number of arrivals $N(S)$ observed over the duration S and the “Resolution” object contains data pertaining to that individual resolution such as polynomial coefficients of the estimated mean-value function fitted to that resolution.

Other types of classes in the “mrFitData” package are the factory classes, which are creators of the *data objects*. The “ArrivalDataFactory” and “ResolutionFactory” are factory classes generating “ArrivalData” and “Resolution” *data objects* respectively. The “ResolutionFactory” class accepts the arrival times over entire duration. It separates the arrivals over the entire duration length into different cycles or sub-intervals if the NHPP contains any periodic cycles. Other than separating arrivals for cyclic components, it populates information for each resolution such as maximum degree, which is fitted at each resolution, length of resolution, and so on. It creates an array of resolution objects, the first resolution representing the lowest level.

“MRFitVector” is a utility class performing various services on an array data structure. This class is used by many other classes to perform common functions for data structures; for example, sorting the values in an array in ascending or descending order. The figure 7.2 is a class diagram for the package “mrFitData” and displays all the associations and the dependencies amongst various classes of the package. It also lists the member variables and the properties of all the classes present in the “mrFitData” package.

Package “mrFitfunctionEvaluation”: This package contains the classes performing the fitting operation on cumulative arrivals that are separated in various resolution objects.

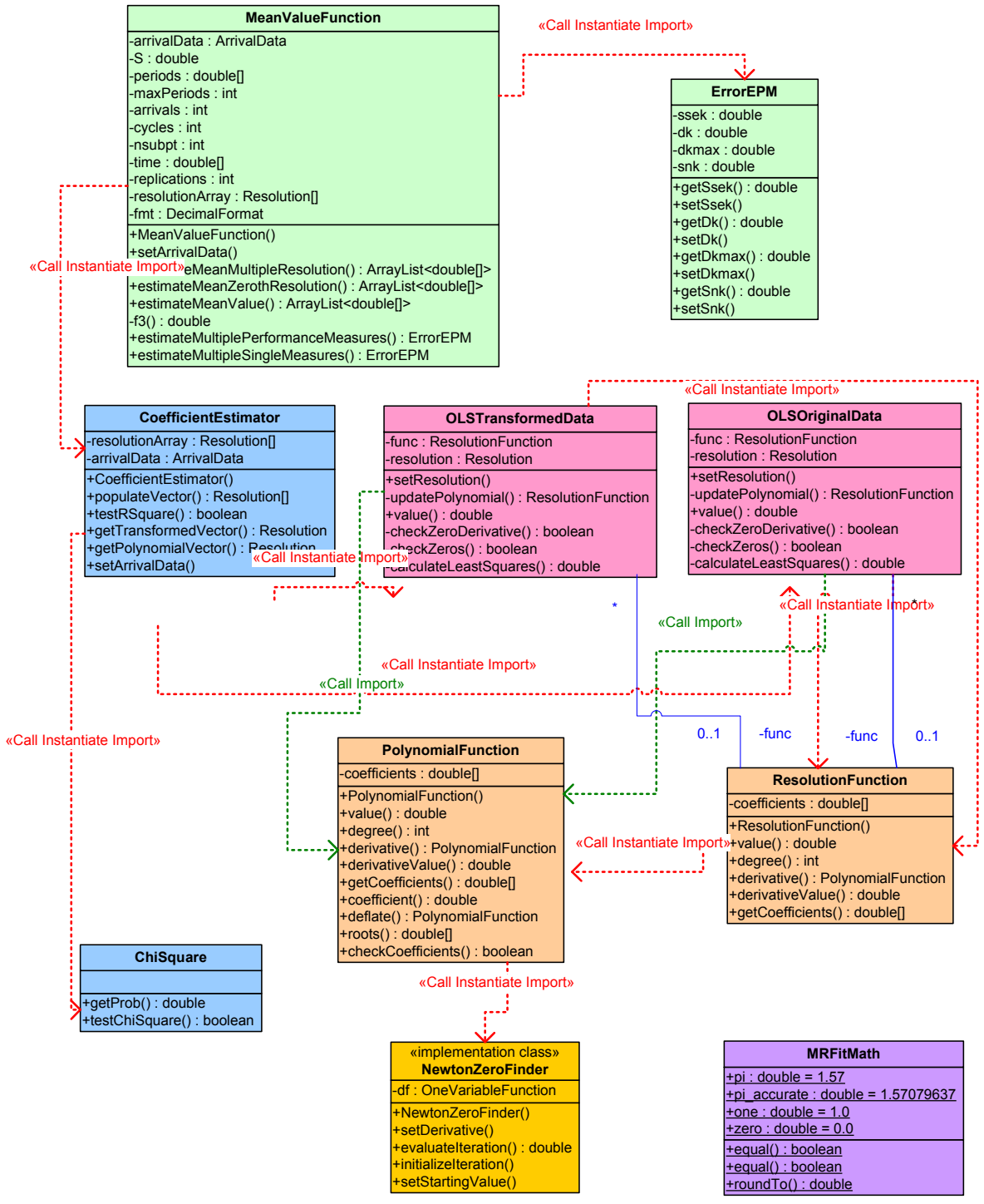

Figure 7.3 Class Diagram for "mrFitFunctionEvaluation"

Figure 7.3 represents the class diagram for the package "mrFitFunctionEvaluation". The "CoefficientEstimator" class uses the likelihood ratio test to determine the degree r of the

polynomial function that is best fitted to the arrival data. The "CoefficientEstimator" automates the fitting process to estimate the polynomial function that fits to the cumulative arrivals. The likelihood ratio test uses ordinary least squares to compare the fitted function for two consecutive iterations. The ordinary least squares value for the transformed data is estimated by the class "OLSTrasnformedData" and for the actual cumulative arrival data is estimated by the class 'OLSOriginalData". The likelihood ratio test uses the chi-square test to determine quality of the current fit against an earlier one. The chi-square test is performed by the "ChiSquareTest" class and mathematical implementation of this class is provided by a package from Jakarta community. Estimation of degree r is executed for each resolution for the transformed and the original data. The estimated polynomial coefficients are stored in the respective resolution objects.

The class "MeanValueFunction" calculates the value of the mean-value function for various arrival times using the coefficients of the polynomial function estimated by the "CoefficientEstimator". It also plots the step function for the actual cumulative arrivals and the transformed data which serves as visual aid to user in determining the accuracy of the estimated mean-value function.

The 'ResolutionFunction' is a class representing the special form of polynomial function explained in equation (2) of form:

$$f(s) = \beta_1 s + \beta_2 s^2 + \beta_3 s^3 + \ldots + \beta_n s^n \ ,$$

where $[\beta_1, \beta_2, \beta_3, .. \beta_n]$ is a vector of coefficients. The 'PolynomialFunction' class represents the polynomial of form of derivative of the resolution function. The figure 7.3 is a class diagram for the package 'mrFitEvaluation' listing various classes and the interactions between the classes.

Apart from the classes listed in the section, the package 'mrFitEvaluation" contains additional classes that handle various underlying mathematical operations required to derive results.

Package "mrFitInterfaces": All the interfaces are stored in the directory 'mrFitInterfaces'. Two interfaces are declared in the application to represent single-variable function of the form $f(x)$ and multi-variable function of the form $f(x, y, ..)$. The interfaces define a method $f(x)$ that returns the value of the function for a variable x.

Package "mrFitIterations": Classes of the package "mrFitIterations" specify rules for iterative processes. The rules include setting up initial value for the iteration, determining the step size or the incremental value for successive iterations, processing the iteration and checking for convergence. Iterative processes are greatly used throughout this application for estimating roots of equations.

Package "mrFitOptimizing": The fitting method estimates the polynomial coefficient so as to minimize the error value in equation (5) explained in section 4. To minimize the sum of square errors, Nelder-Mead simplex algorithm is used. The package "mrFitOptimizing" contains classes that find optimum values of polynomial coefficients.

Classes of packages; "mrFitInterfaces", "mrFitIterations" and 'mrFitOptimizing" are derived from "Object-Oriented Implementation of Numerical Methods" by Besset. H. Didier after some modification specific to needs of this project. Figure 7.4 represents the class diagram for the package "mrFitOptimizing".

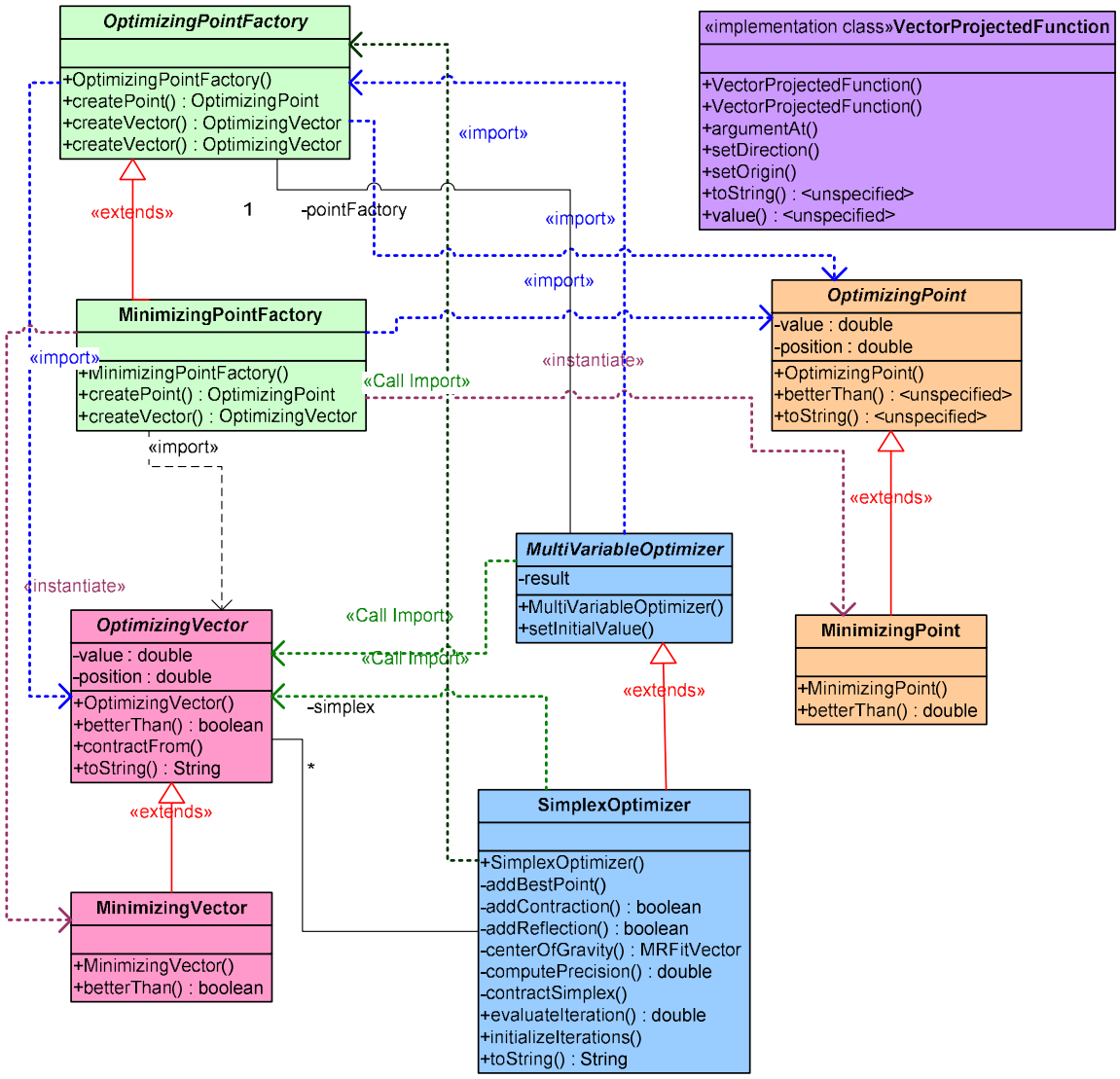

Figure 7.4 Class diagram for "mrFitOptimizing"

Package "mrGenerteData": The second objective of this research is to realize an NHPP from the mean-value function estimated by the fitting process. The generation method estimates the arrival times for an NHPP with or without periodic cycles from the polynomial coefficients that model each resolution. The algorithm for an NHPP realizations is explained in the figure 4.11 and the classes specific to the generation method are stored in the package "mrGenerateData". The class diagram for the package "mrGenerateData" is shown in the figure

7.5. The “InputGenerateData” is a *data object* for storing all the input variables describing the NHPP and the “InputGenerateFactory” is the factory class to create the “InputGenerateData”. The class “GenerateData” estimates arrival times over the duration of an NHPP from polynomial coefficients for single or multiple resolutions. The output of the “GenerateData” method is in form of an array that contains the arrival times for an NHPP realization. In the presence of multiple resolutions, the arrivals times are created after combining the effects of nested cycles.

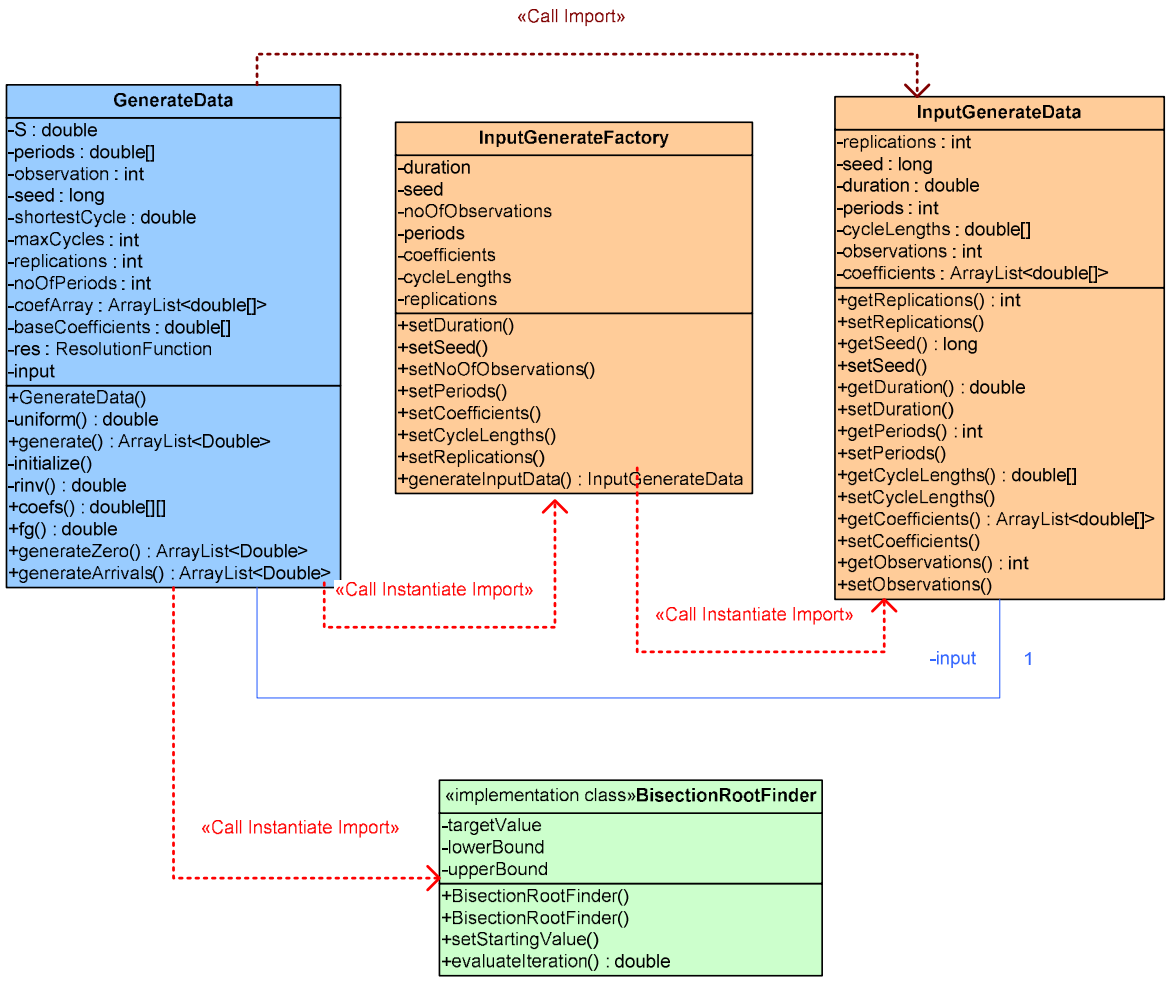

Figure 7.5 Class Diagram for “mrGenerateData”

7.4 Input and Outputs

The fitting method developed in this research fits a polynomial function of degree r to the cumulative arrivals from a single or multiple process realizations of NHPPs. A typical model receives the input, processes the data and delivers the output. The fitting model works in a similar manner and shown in the figure 7.6.

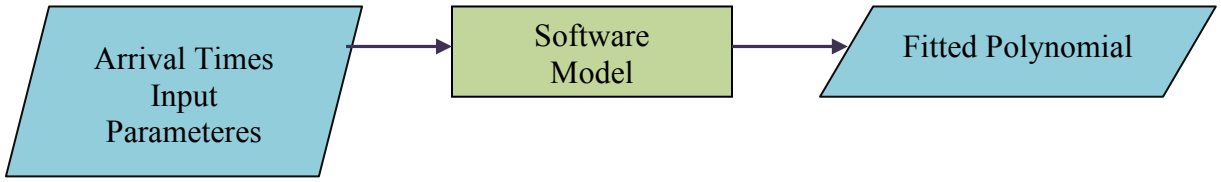

Figure 7.6 Fitting Model for NHPP

The arrival times of a nonhomogeneous Poisson process are stored in a text file, 'mrsim.in' and are read line wise by the application. The arrival times are positive floating point numbers. The other input parameters necessary to obtain the the results from software model are mentioned in the table 7.1.

Table 7.1 Inputs for Fitting Program

Inputs	Description
Length *S*	Length of the observation interval over which the arrivals are observed. This can be a double value.
Number of Resolutions	The 'resolution' refers to the number of periodic cycles observed over the observation length *S*. If there are no periodic cycles then this value is 0 else a positive integer is entered.
Length of Cycles	The length of each periodic cycle needs to be entered. The values should satisfy the criteria for the multiresolution procedure explained in section 3.3.
Maximum degree Fitted	The maximum degree till which the likelihood ratio test should be conducted. If there is a positive improvement seen in fitted model till maximum degree, execution stops at mentioned maximum degree. A positive integer needs to be entered.
Number of Realizations	If the data is collected from multiple process realizations then the corresponding positive integer is entered or else the value is 1.
Number of sub points	The number of points for which the fitted mean value function will be computed and plotted. The points are equally spaced in observation interval *S*. A positive integer should be entered.

The generation program realizes a nonhomogeneous poisson process from polynomial coefficients. The input parameters for the generation program are listed in the table 7.2.

Table 7.2 Input Parameters for Generation Program

Inputs	Description
Length *S*	The length of the observation interval over which the arrivals are observed. This can be a double value.
Number of Resolutions	Number of resolution refers to number of periodic cycles observed over length *S*. If there are no cycles then this value is 0 else a positive integer is entered.
Length of Cycles	The length of each periodic cycle needs to be entered. The values should satisfy the criteria for the multiresolution procedure explained in section 3.3.
Random Seed	The random seed is used to generate same sequence of random numbers for each execution. It a big positive number less than 9999999999.
Number of Realizations	More than a single process realization can be generated using by setting this input parameter. A positive integer more than 0 should be used.
Polynomial Coefficients	This is a mean value function in form of a polynomial function of degree *r*. The polynomial function should be of form explained in equation (2) in section 4. A polynomial function needs to be entered for each periodic cycle in case of multiple resolutions.

7.5 Development of Web Interface

The internet has become a common platform of communication in today's world and its easy accessibility and ubiquitous nature needs to be considered while designing any application. Current research leverages the advantages provided by the web-based applications to develop a common interface over which the fitting and generation programs can be executed by multiple users over a single server. A web interface helps users to use the program without actually fulfilling the hardware requirements necessary for its execution. The web interface developed in this research accepts or receives the inputs over the internet and provides the output over the same platform. The actual execution of the program runs in the background and is invisible to the user. As mentioned in the section 7.2., the Java platform is used to develop the software model; hence, a Java-based web application interface is developed to have a common platform for the entire application. A *web application* is defined as an application, which is accessed via web browser and *Java web application* refers to a collection of servlets, html pages, classes, and other resources that can be bundled and executed on web servers. A web application is developed in this research using the Java platform. The components constituting the developed web application are:

- Servlets
- Servlet Container
- Static HTML Pages
- Java Server Pages and Java Script
- Deployment descriptor
- Java Applets.

The various elements taking part in building the web application are explained in the following section. The functions performed by the various elements of the web application are also explained briefly.

Java Servlets:

The Java servlet allows the existing Java application functionalities to be extended over the internet. A *Java servlet* is in reality a Java program with *.java extension that runs on a server, resides in a servlet container and dynamically processes requests and generates responses. A *servlet container* is a program that loads, initializes, and executes servlets and acts as a layer between the servlet and the web. The requests processed by servlet, specific to our projects are HTTP requests sent over the internet through web browsers by users. The servlets receive the HTTP request objects, processes the request object depending upon the program logic, and then generate a response in form of an HTTP response object. A typical servlet resides in servlet container for its entire life cycle. The life cycle of a typical servlet can be described by Harbourne(2004) :

1. The servlet is initialized by servlet container at beginning.
2. The servlet components receive requests over the web. The container actually receives the request, transparently maps the request to the appropriate component instance, and passes the component properly-formatted request and response objects.
3. The servlet processes the request, normally with the help of either the business tier logic (EJB's) or by retrieving information directly from the database or enterprise information tier.
4. A response is returned to the client tier.
5. The container is responsible for destroying any servlet instances that it has created.

As mentioned in the second point, the servlet container maps concerned servlet objects for specific requests sent by the users. In the application, there are two types of requests sent by the user; the request to fit a mean-value function to single or multiple process realizations and a request to generate arrival times for the NHPPs. Two servlets are written to cater these requests; "FitArrivalDataServlet" and "GenerateArrivalDataServlet". The information entered by the user about the inputs for software program is received by servlets as HttpRequest objects. The input parameters discussed in the table 7.1 and 7.2 are then extracted from request objects and the data objects are generated by the factory classes. In the case of fitting data; "MeanValuefunction" object is used to determine the polynomial function and for generating the "GenerateData" object for realizing NHPP. A response is then displayed to the user in the form of HTML pages and applets through Java Server Pages (JSP) technology.

Servlet Container:

A *servlet container* is essentially the component of a web server interacting with the servlets. The servlet container is responsible for managing the life cycle of servlets and mapping a URL to a particular servlet.

HTML Pages:

HTML stands for *Hyper Text Markup Language* and is extensively used to display content over web pages. It is not a programming language, but a markup language defining the syntax for displaying text and content over internet using tags. The web browsers such as Mozilla Firefox, Internet Explorer, etc. read the HTML documents and display them to the user as web pages. Browsers do not display HTML syntax or tags to viewers, but views itself as

defined by HTML. HTML defines pages using HTML tags, which are HTML key words surrounded by angle brackets (e.g. <html>). The HTML pages are used to receive inputs from users over web page using HTML forms. An HTML form has elements such as text fields, radio buttons, and drop-down list allowing a user to input information. The designed web application uses HTML forms to receive inputs for the fitting and generation methods.

Apart from taking form field inputs, the HTML pages are also used to display static content such as pictures and text labels helping a user to understand the web page. All the HTML pages are stored in a web content folder for the application project.

Java Server Pages:

The *Java Server Pages* (JSP) technology is introduced by Sun Micro systems to enable the display of dynamic content over the web pages. The JSP is an extension of Java servlet technology and separates web content generation from presentation. In simple terms, JSP is a HTML page which has liberty to use Java code for setting dynamic content. As mention by Goodwill (2000), a JSP engine is just another servlet mapping to the extension *.jsp. When a JSP is initially displayed by a web server, it compiles a file with an *.jsp extension and creates a servlet object out of it which is stored in memory. When the request for that particular JSP file repeats, the web server verifies whether the JSP file is modified. In the case of the same JSP file, the web server invokes the existing servlet objects present in memory. If the JSP file changed during the period between two requests, then the page is recompiled to generate new servlet file. Fulfillment of request for JSP file is shown in the figure 7.7.

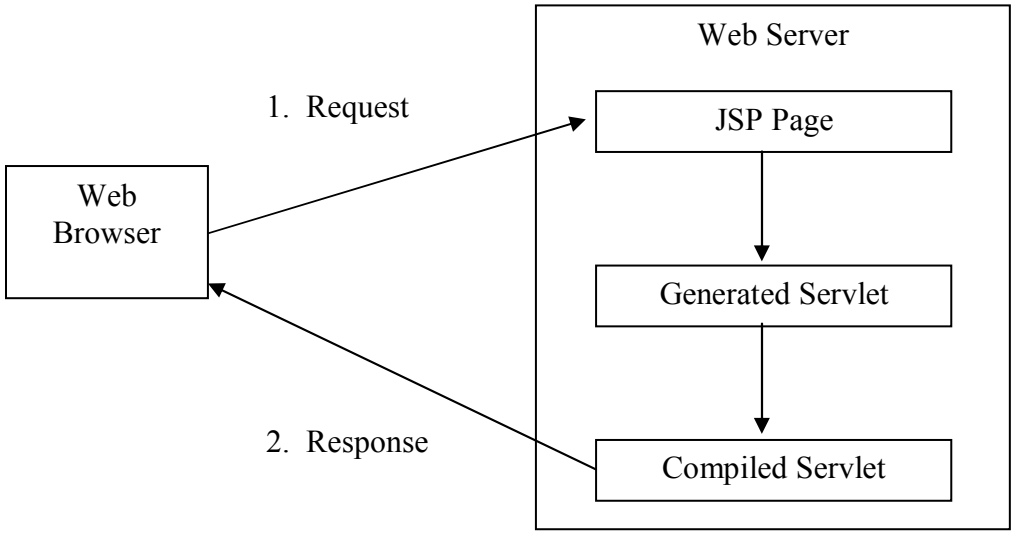

Figure 7.7 HTTP Request and Response

All the web pages are designed as Java Server Pages in this project. Java code is used in JSPs to set the values of applets which display graphs of fitted mean value function to user. Java code is also utilized to upload text file with arrival times on server. Along with Java code, JSPs also contain Java script to perform various validations discussed in next section.

Java Script:

The *Java script* is web scripting language used to add interactivity to web pages. It is a programming language, which is used to perform the functions like form field validations, dynamic display of HTML elements, creation of cookies etc. The Java script can read and change content of HTML elements, reload html page, execute events and thus provide the designers a better control over HTML pages. The developed web application uses Java script to perform validation checks on inputs entered by user before the inputs are submitted to web server. The Java script provides feedback to users about legitimacy of their inputs by sending alerts if the entered format of input is other than the expected format. For example Java script checks whether user enters a positive integer value for the input field 'number of realizations' in

the table 7.1 and 7.2. Appendix D contains an example of a Java script function that validates an input form field.

Other purpose of Java script in this project is to control display of HTML elements. For example Java script is used to dynamically change the number of text boxes displayed on HTML page to enter coefficients of the polynomial function in generation program. The number of text boxes is adjusted to the degree r of polynomial entered by user.

Classes:

Java Classes are deployed on server in the form JAR files or Java archive files and stored with extension *.jar. It is essentially a zipped folder containing all the Java classes, which are used by Java Servlets to perform various functions. The Java classes are same as discussed in section 7.3 that implement the fitting and the generation methods.

Deployment Descriptor:

A *Deployment descriptor* describes components of a web application and is an XML file with the name, web.xml. XML stands for *Extensible Markup Language* and is a general-purpose specification for creating custom markup languages. In the developed application, the web.xml describes various servlets responsible for catering different types of HTML requests by assigning different type of URL patterns to servlet names. Also, web.xml is used to define a default URL pattern to start execution of an operation. The deployment descriptor for the application is included in appendix D for reference.

Java Applets:

Java applets are the java programs executed on the client machine by using the *Java Runtime Environment* (JRE). The Java applets are object embedded in HTML pages within tags <applet> and </applet>. Research uses Java applets to plot graphs for fitted functions and data. The data is dynamically populated for an applet on client machine from HTML files.

References

A., H.-T., J., B., & S., B. (2002). *Professional Java Servlets 2.3*: Peer Information.

Besset, D. H. (2000). *Object-Oriented Implementation of Numerical Methods: An Introduction with Java & Smalltalk*: Morgan Kaufmann.

Booch G., R. J., Jacobson I. (2005). *Unified Modeling Language User Guide, 2nd Edition*.

Çinlar, E. (1974). *Introduction to Stochastic processes*. Englewood Cliffs, New Jersey: Prentice Hall.

Crowder. (1991). *Statistical Analysis of Reliability Data*. London: Chapman & Hall.

Draper, N. R. (1998). *Applied Regression Analysis* (3rd ed.). New York: John Wiley.

Ebeling, C. E. (1996). *An Introduction To Reliability and Maintainability Engineering*. New York: McGraw-Hill.

Harrod, S., & Kelton, D. W. (2006). Numerical methods for realizing nonstationary poisson processes with piecewise-constant instantaneous-rate functions. *Simulation, 82*(3), 11.

Henderson, s. G. (2003). *Input modeling: input model uncertainty: why do we care and what should we do about it?* Paper presented at the Winter Simulation Conference, New Orleans, Louisiana, US.

Johnson, M. A., Lee, S., & Wilson, J. R. (December 1991). *Experimental evaluation of a procedure for Estimating non-homogeneous Poisson processes having cyclic behavior.* Paper presented at the Winter Simulation Conference, Phoenix, Arizona, United States.

Kuhl, M. E., & Bhairgond, P. S. (December 2000). *Nonparametric estimation of*

nonhomogeneous Poisson processes usingwavelets. Paper presented at the Winter Simulation Conference, Orlando, Florida, USA.

Kuhl, M. E., Sumant, S. G., & Wilson, J. R. (2006). An automated multiresolution procedure for modeling complex arrival processes. *INFORMS Journal on Computing, 18*(1), 16.

Kuhl, M. E., Wilson, J. R., & Damerdji, H. (1998). *Least squares estimation of nonhomogeneous Poisson processes.* Paper presented at the Winter Simulation Conference, Washington, D.C., US.

Kuhl, M. E., Wilson, J. R., & Johnson, M. A. (1997). Estimating and Simulating Poisson Processes Having Trends or Multiple Periodicities. *IIE Transactions, 29*(9), 11.

Leemis, L. M. (1991). Nonparametric estimation of the Cumulative intensity function for a nonhomogeneous Poisson process. *Management Science, 37*(7), 5.

Lewis, P. A. W., & Shedler, G. S. (1976a). Simulation of nonhomogeneous Poisson processes with log linear rate function. *Biometrika, 63*(3), 5.

Lewis, P. A. W., & Shedler, G. S. (1976b). Statistical Analysis of Non-stationary Series of Events in a Data Base System. *IBM Journal of Research and Development, 20*(5), 18.

Maclean, C. J. (1974). Estimation and testing of an exponential polynomial rate function within the nonstationary Poisson process. *Biometrika, 61*(1), 5.

Majeske, C. D. (2007). A non-homogeneous Poisson process predictive model for automobile warranty claims. *Reliability Engineering & System Safety, 92*(2), 9.

Massey, W. A., Parker, G. A., & Whitt, W. (1996). Estimating the parameters of a non-homogeneous Poisson process with linear rate. *Telecommunication Systems, 5*(2), 28.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2006). *Introduction to Linear Regression Analysis* (4th ed.): Wiley.

N.R., D., & Smith, H. (1998). *Applied Regression Analysis* (3rd ed.). New York: Wiley.

Nourie, D., & Pawlan, M. (2007). Introducing the Java Platform available via http://java.sun.com/new2java/programming/intro/

Pritsker, A. A. B. (1998). Life and death decisions: organ transplantation allocation policy analysis. *OR/MS today, 25*, 7.

Sumant, S. G. (2003). *An Automated Procedure for Simulating Complex Arrival Processes: A web-based approach.* Rochester Institute of Technology, Rochester.

Vere-Jones, D. (1970). Models for Earthquake Occurrence. *Journal of the Royal Statistical Society, 32*(1), 62.

Walpole, R. E., Myers, R. H., & Myers, S. L. (1998). *Probability and Statistics for Engineers*: Prentice Hall, Inc.

Wang, Z., Wang, J., & Liang, X. (2007). Non-parametric Estimation for NHPP Software Reliability Models. *Journal of Applied Statistics, 34*(1), 107.

White, K. P., Jr. (1999). *Simulating a nonstationary Poisson process using bivariatethinning: the case of "typical weekday" arrivals at aconsumer electronics store.* Paper presented at the Winter Simulation Conference, Phoenix, Arizona, US.

APPENDIX A - USER GUIDE FOR WEB INTERFACE

Guide for Using Fitting Program and Generation Program

This research develops flexible models for fitting the nonhomogeneous Poisson process. A web interface is provided to users to invoke the software program over the internet. This is a step by step guide for fitting the arrivals observed over single or multiple process realizations of NHPPs using a polynomial function and generating the arrivals times from the estimated mean-value function. The web-based software application can be accessed by typing http://simulation.rit.edu/Fittingmethod in the web browser. The link invokes the welcome page, which essentially is "start.jsp".

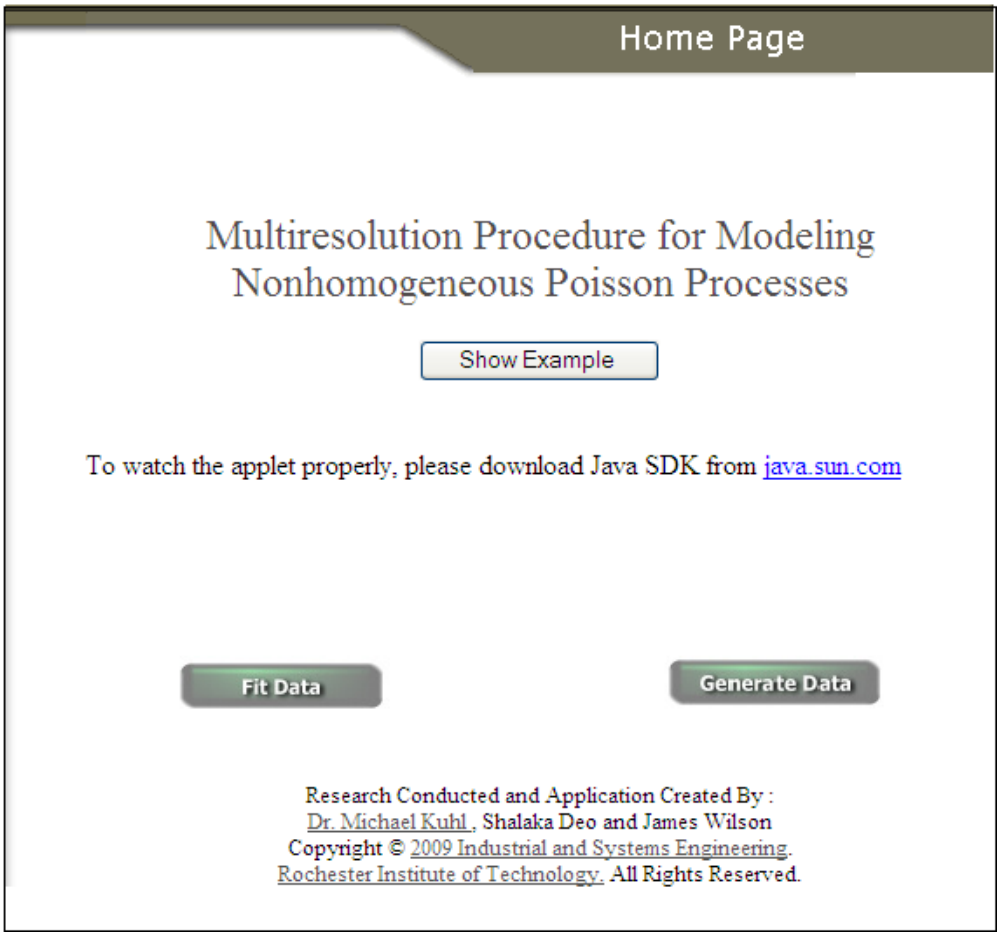

Figure A.1 Welcome Screen of Web Based Input Modeling

Figure A.1 represents the snapshot of home screen, which is starting point of the application and allows execution of both the generation and fitting programs. Follow the instruction to execute the various programs.

Generation Program:

The generation program is initiated by selecting "Generate Data" option from the welcome screen. The user is shown the page in Figure A.2 where the user is provided with two alternatives for entering the inputs necessary for the generation method. The user can either enter the input parameters manually from the input screen using the keyboard, or user can utilize the text file generated by the fitting program as output. Select the appropriate radio button and press the "Start Generation Program" button.

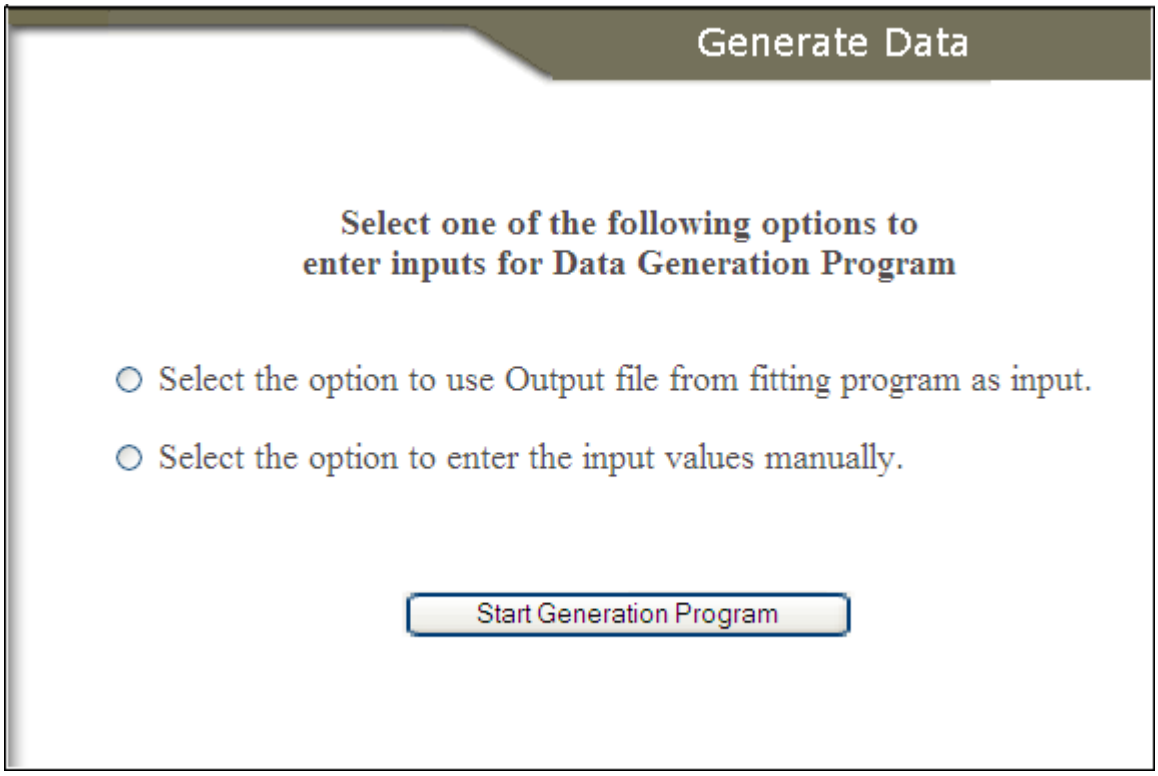

Figure A.2 Options for Inputting Parameters

Option 1: Output file from Fitting Program for Input

The option of using text file as input displays a screen, as shown in Figure A.3, to facilitate the upload of the text file. The input file contains the details about NHPP required by the generation program. The "Browse" button may be used to browse to the location where the input file is stored and "Send File" button can be used to send the file to the server.

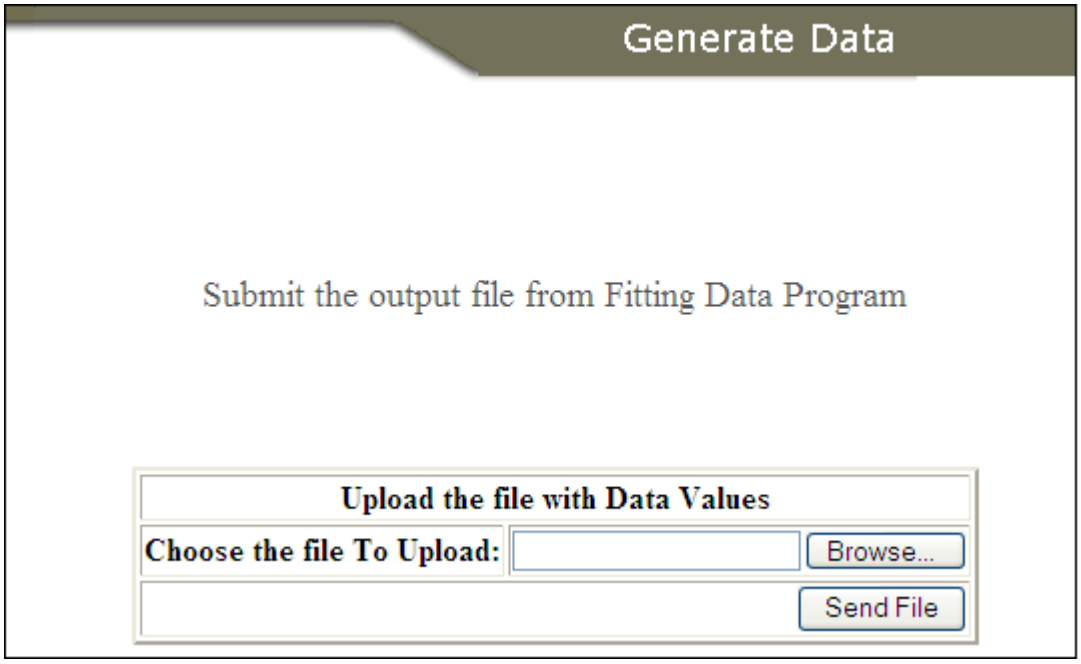

Figure A.3 File Upload Screen for Generation Program Inputs

```
/*Random Seed */
86836444
/* Length of obsevation interval (S) */
10
/* Expected number of arrivals in [0,S] */
150
/*Number of Replications */
1
/* Number of periodic components */
1
/* Periods (Longest to shortest) */
1
/* Degree of Polynomial */
1 , 3
/* Polynomial Coefficients */
0.01
1.62999919 , -2.69999442 , 2.06999523
```

Figure A.4 Parameters of Input File for Generation Program

Figure A.4 shows the format of the input text file, which is accepted by the generation program as input. The random seed is a default number printed out in the text file 'mrsim.in'. Always use a comma as a delimiter between the two values. Only text file formats are accepted by the Generation program. Any invalid file format generates an exception condition.

Option 2: Enter Inputs Manually

If the user wishes to manually enter the input values, then user can choose the corresponding option in Figure A.3. The form for entering input variables displays in figure A.5 Enter inputs for the first four fields in the respective text boxes. At any point, the "Help Menu" can be invoked by clicking on the question mark icon on right side of text boxes. The help menu explains the purpose of individual form fields and the format of input accepted from the user.

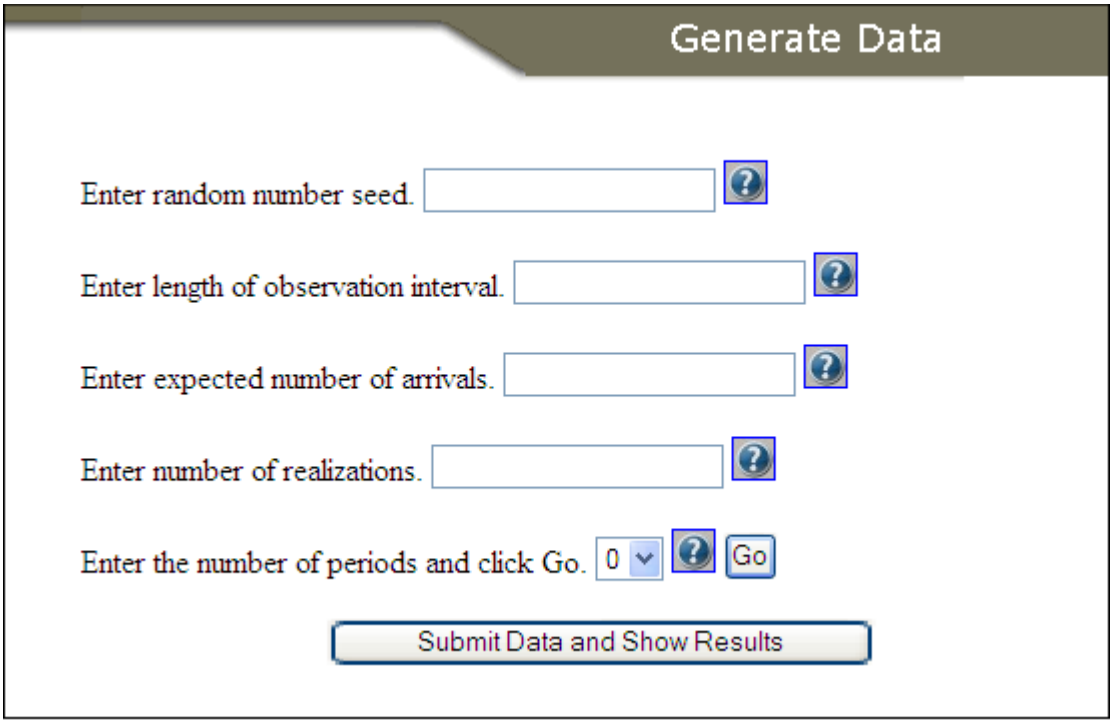

Figure A.5 Input Screen for Generation Program

The input field for the "number of cycles" is a drop-down list and the available options can be seen by clicking on the arrow pointing downwards. Select the correct number of cycles and then

press the “Go” button to enter the details of each resolution. After pressing “Go”, additional form fields are displayed as shown in the figure A.6 to enter the resolution lengths and the degree of the polynomial mean-value function at each resolution.

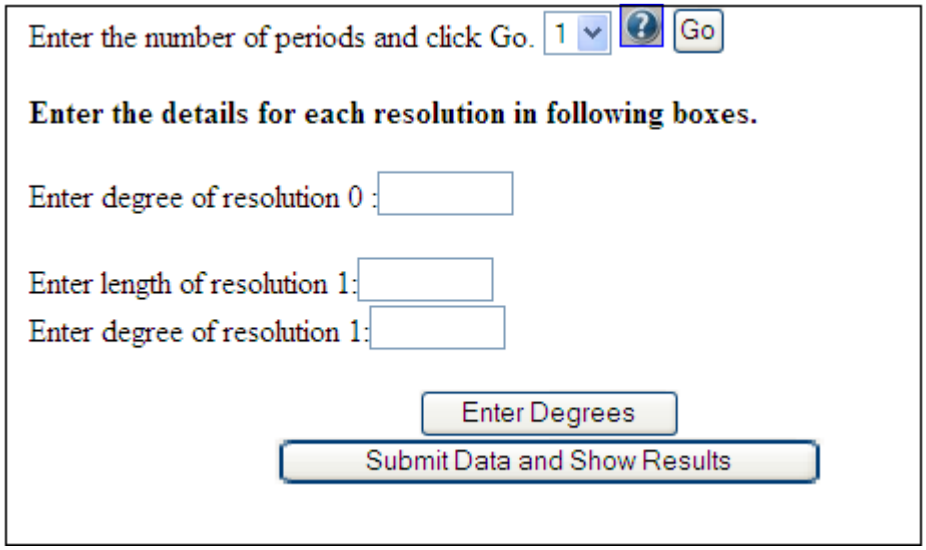

Figure A.6 Form Fields for Cycle Lengths and Degrees.

Input the values of the cycle lengths and the degree r of the polynomial at each resolution, and then press the “Enter Degrees” button for displaying the text boxes for entering the polynomial coefficients as illustrated in Figure A-1.6. Observe that the number of text boxes for the polynomial coefficients correspond to the values entered for the resolution degree. After entering all the input variables, click the “Submit Data and Show Results” If there is any error in input values, the user is alerted by a prompt, which highlights the specific error. An example of validation prompt is seen in Figure A.7. The prompt also displays if any field is left blank and when the execution of the generation program is attempted. With the correct input values, the generation program is executed and the user views the graph displaying the cumulative NHPP arrivals. If the user requests multiple realizations of NHPP, the graph is plotted for the first NHPP realization. The plotted graph demonstrates the pattern of realized NHPP. Along with graph, the user can download the file of NHPP arrival times using “Show Generated Data” link.

The output screen is illustrated in Figure A.8. For multiple realizations of NHPP, arrivals from the first realization are followed by the arrivals from second and so on. Figure A.9 displays the graph for generated cumulative arrivals over the observation duration.

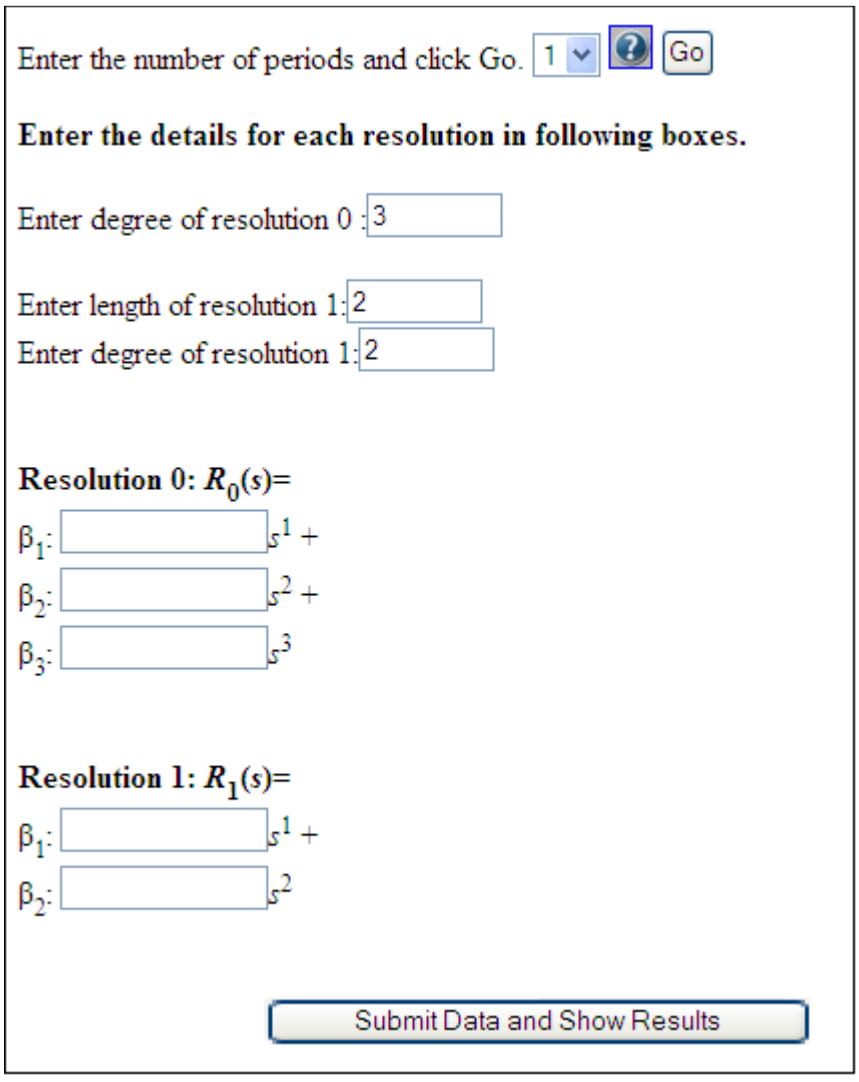

Figure A.7 Form Fields for Polynomial Coefficients.

Figure A.8 Validation Prompt for Erroneous Inputs.

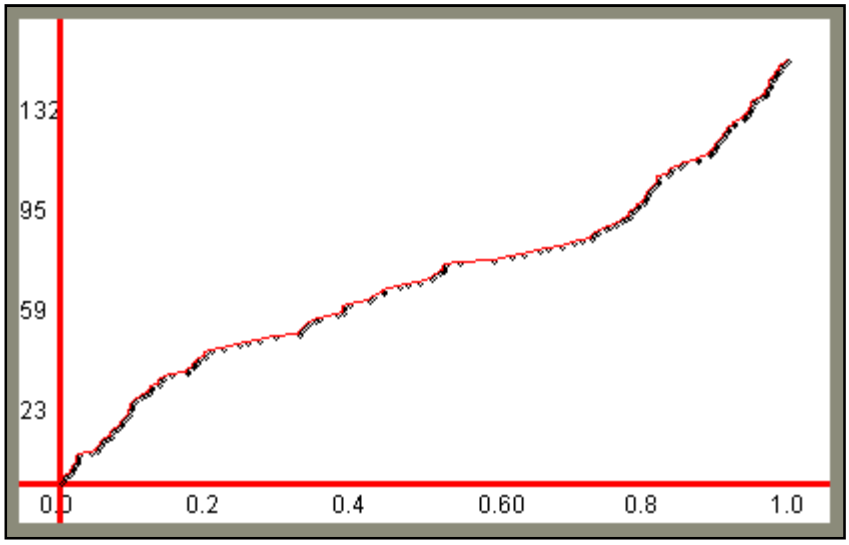

Figure A.9 Graph for Generated Cumulative Arrivals

Fitting Program:

The "Fit Data" button on screen A.1 is used to start the fitting program that models the mean-value function for the NHPP arrivals. At the beginning, the user is prompted to upload the text file with the arrival times. Time of each new arrival is saved on new line. The screen shown in Figure A.10 is used to browse location of text file. Press the "Send File" button to upload arrival times on the server.

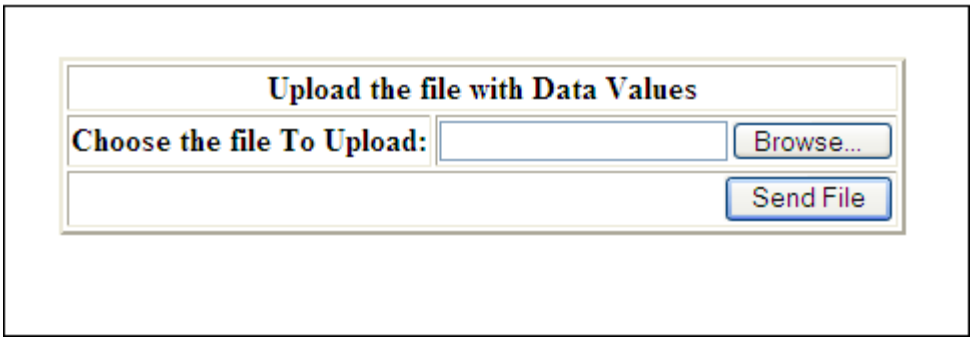

Figure A.10 File Upload Screen for NHPP arrivals

Once the file is uploaded on the server, the user receives the screen in Figure A-1.10 for setting parameters for the fitting method. The help menu may be opened using the question mark icon (the same as in the generation method).

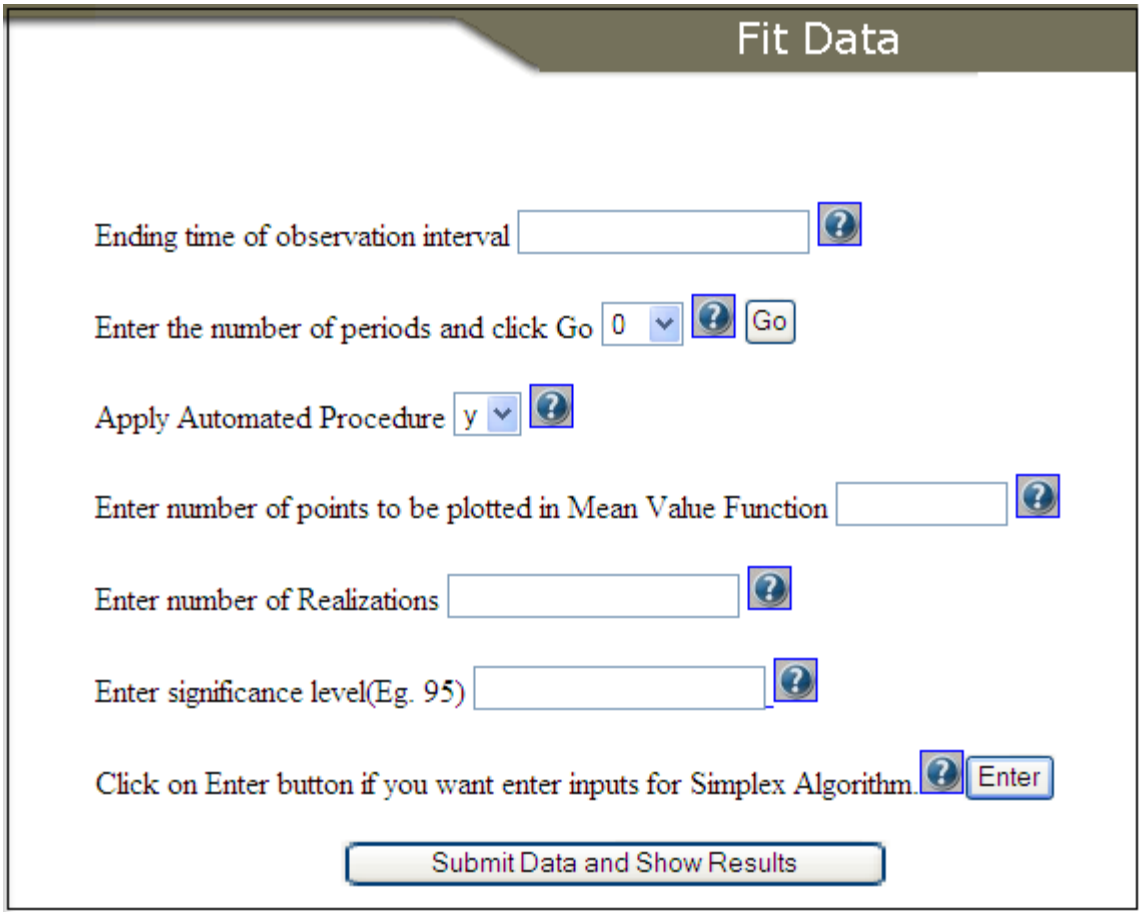

Figure A.11 Input Screen for Fitting Program

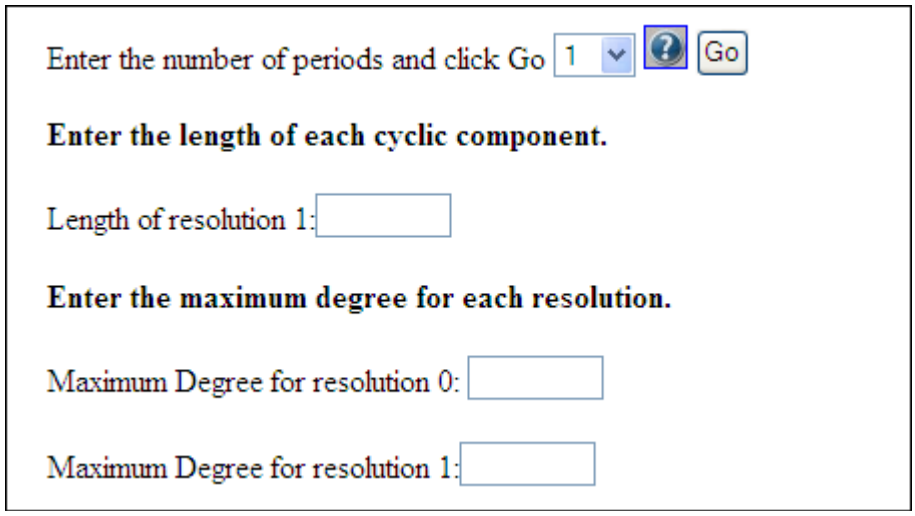

Figure A.12 Input Fields for each Resolution

Select the number of periods from the drop-down list and press "Go" to enter details for each resolution. Additional input fields are displayed as shown in figure A.11. The length at zero resolution is the duration for the entire NHPP realization; hence, there is no input field present for that particular property. After entering values for all fields, click on "Submit Data and Show Results" button to execute the Fitting program.

Validation for the input parameter values is performed and the user is prompted in case of any error. The prompt message is similar to the screen shot shown in Figure A.7 with details of all errors made by user. After rectifying the errors, the user resubmits the form. The Fitting program is executed and a polynomial function is estimated. The Fitted mean-value function is plotted against actual cumulative arrivals as shown in Figure A.9. Various options are provided to the user to the fitted function for original data and transformed data at each resolution. These graphs can be seen by selection respective options in fields.

All the steps in executing fitting and generation program are summarized in preceding section.

APPENDIX B - UNDERSTANDING UML DIAGRAMS

The Unified Modeling Language (UML) is a popular notation for creating models of object-oriented software (Booch et al. (2005)). UML is a set of graphical notations used to view the software model at varying levels of abstractness. There are various types of UML diagrams, for example, “Use Case Diagrams”, “Class Diagrams”, “Interaction Diagrams”, “State Diagrams”, and “Physical Diagrams”. All the mentioned types of UML diagrams are specifically designed to serve different purposes and provide a different perspective to software model designers and implementers. Current research uses a class diagram to describe the architecture of the software program. A brief discussion of the class diagram is included in this section to help the user understand the class diagrams of Section 7.1 with most commonly used notations of class diagrams explained.

The class diagrams are used to describe class structure in a package and the class methods and the attributes. Apart from detailing the class content, the class diagrams also explain the relationships among various classes. A typical class diagram consists of the class name, the member attributes and the class methods. Using the UML class diagram constituents of a class are shown graphically as seen in the figure B.1

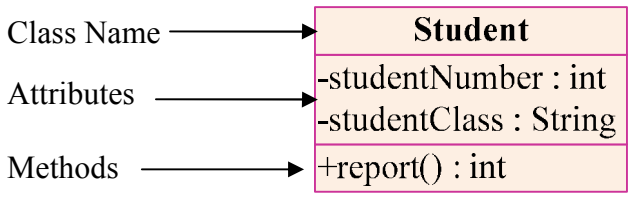

Figure B.1 Class Diagram

The name of the class shown in the figure B.1 is ‘Student’. The attributes or member variables are ‘studentNumber’ of data type ‘int’ and ‘studentClass’ of datatype ‘String’. The (-) sign at beginning of the name of the member attributes indicates that the member variables are private and can be accessed only from the class. The ‘report’ is the method of the class ‘Student’ with return type of ‘int’. The (+) sign at beginning of the method name mean that the method is public and can be accessed from outside the class. Thus the signs (-) and (+) are used to indicate the type of access modifier for class members.

Apart from outlining the class structure, class diagrams illustrate the relationships among the various classes such as containment, inheritance, associations etc. Association relationship demonstrates relationship between instances of the classes. It is shown by drawing a solid line connecting two classes. Multiplicity at end indicates total number of classes that is/are associated with a single instance of connecting class and vice versa.

APPENDIX C – ADDITIONAL GRAPHS OF EXPERIMENTS

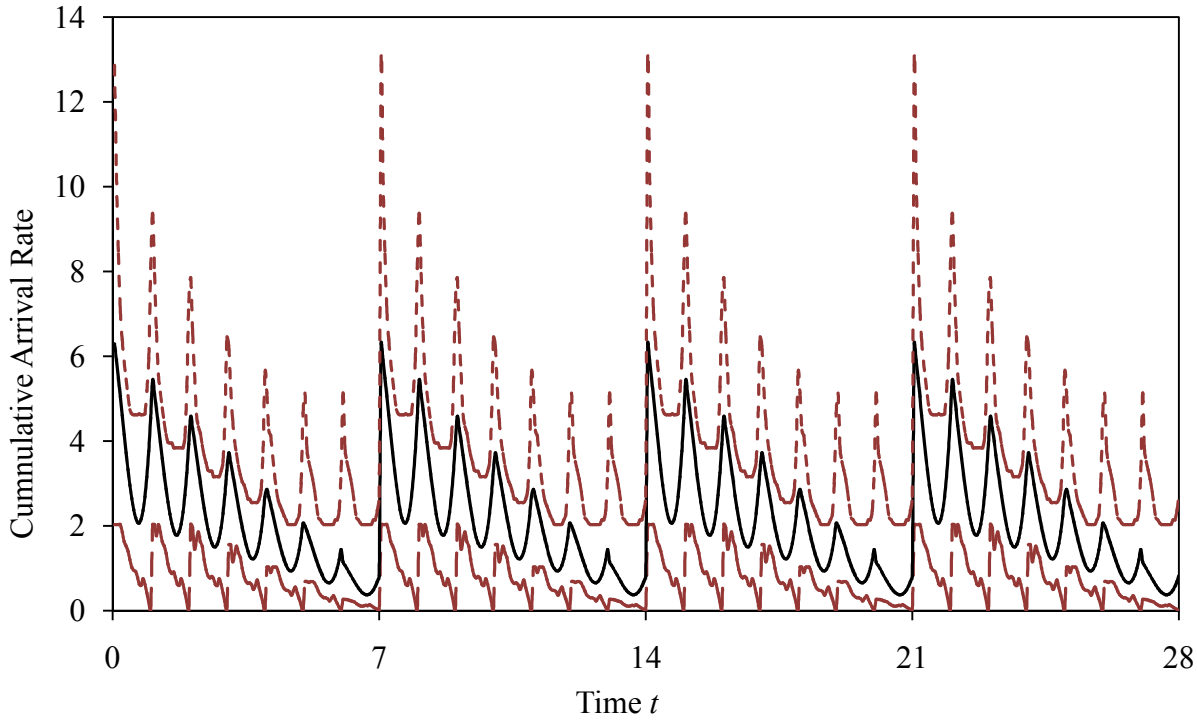

Figure C.1 90% Tolerance Intervals for λ(t),t ∈ [0,28] in Case 1, Single Realization

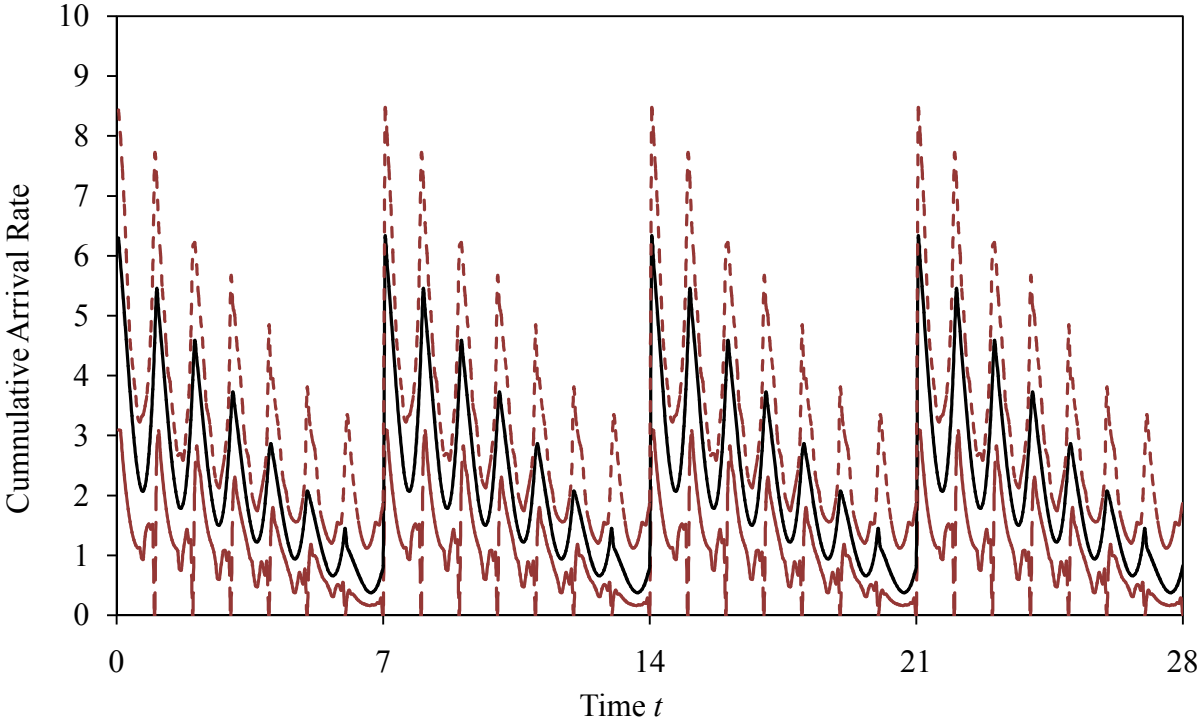

Figure C.2 90% Tolerance Intervals for λ(t),t ∈ [0,28] in Case 1, 3 Realizations

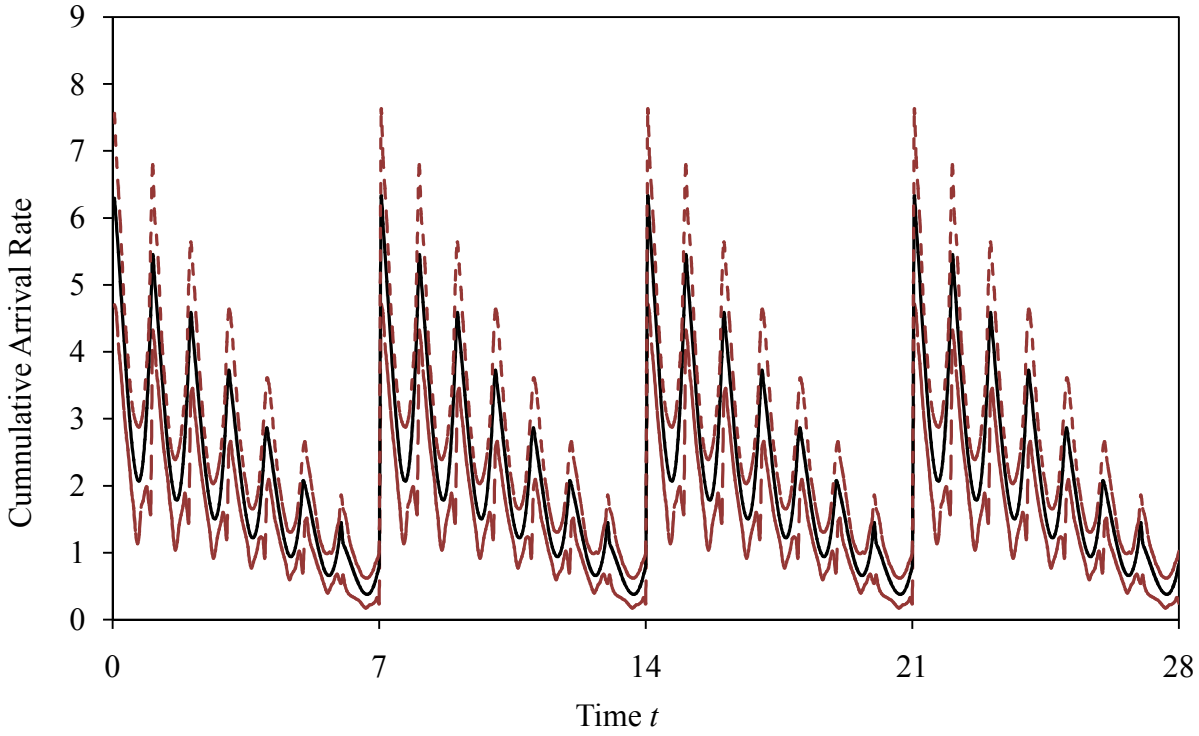

Figure C.3 90% Tolerance Intervals for λ(t),t ∈ [0,28] in Case 1, 8 Realizations

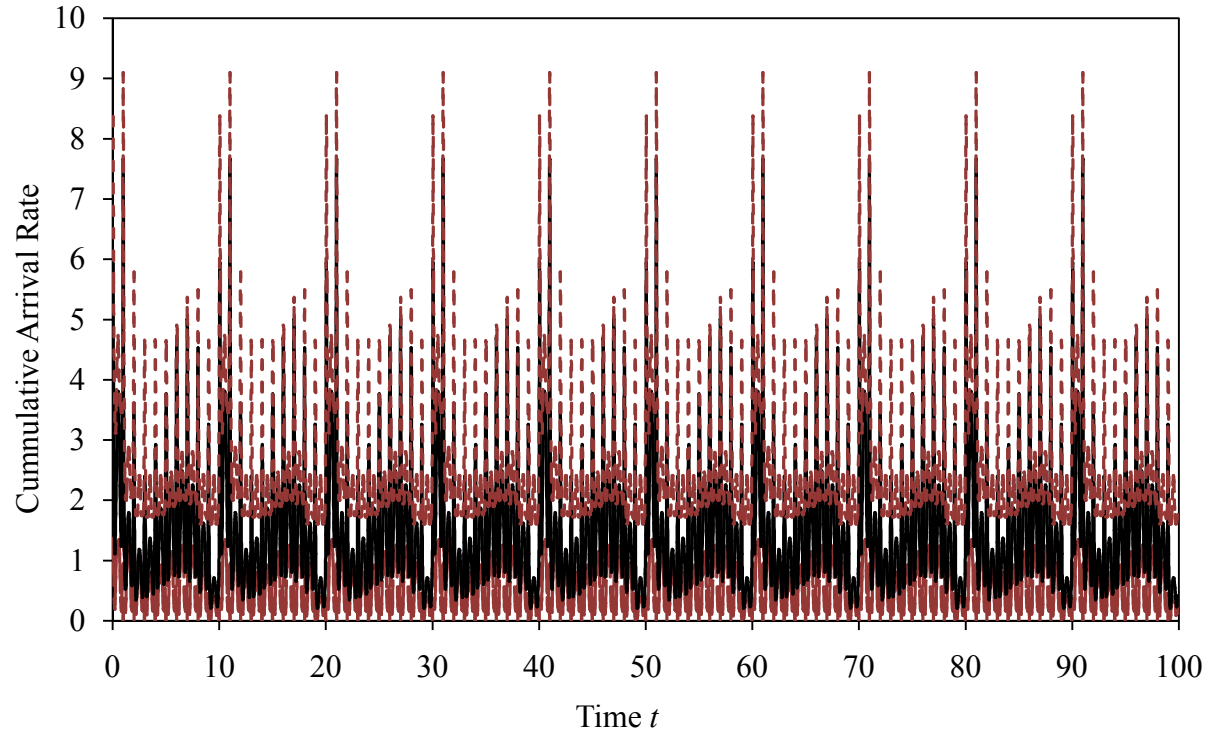

Figure C.4 90% Tolerance Intervals for λ(t),t ∈ [0,100] in Case 2, Single Realization

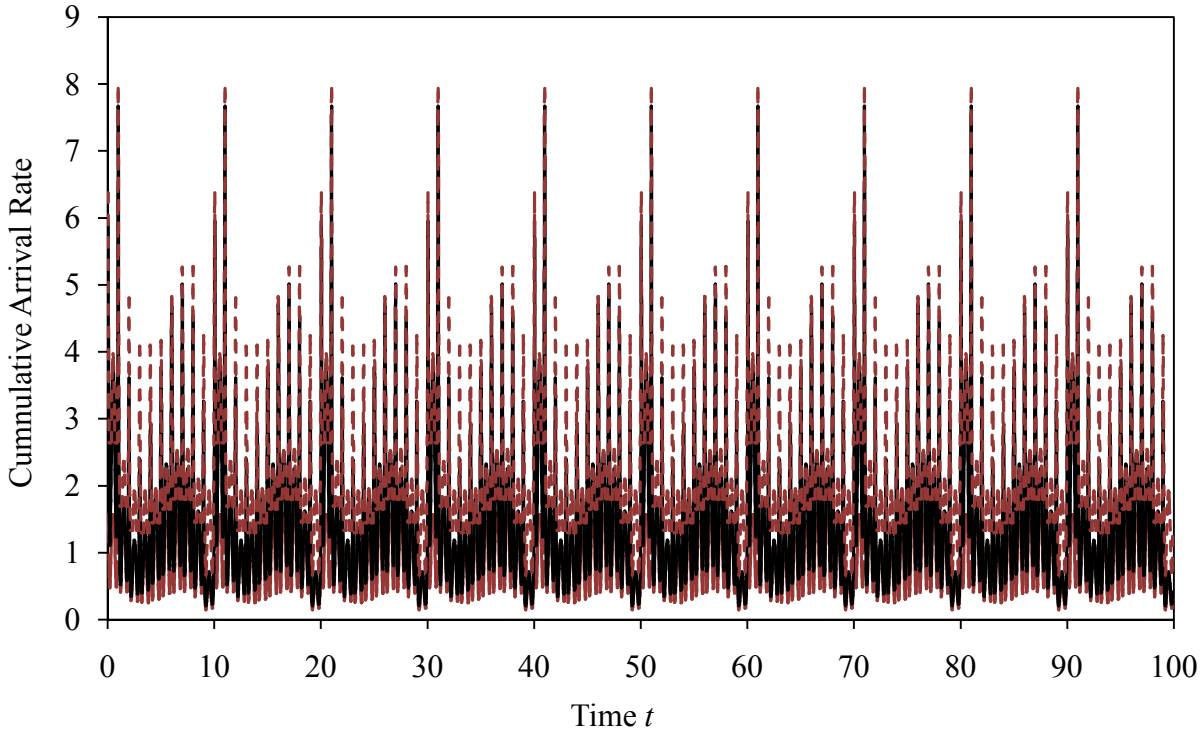

Figure C.5 90% Tolerance Intervals for λ(t),t ∈ [0,100] in Case 2, 8 Realizations

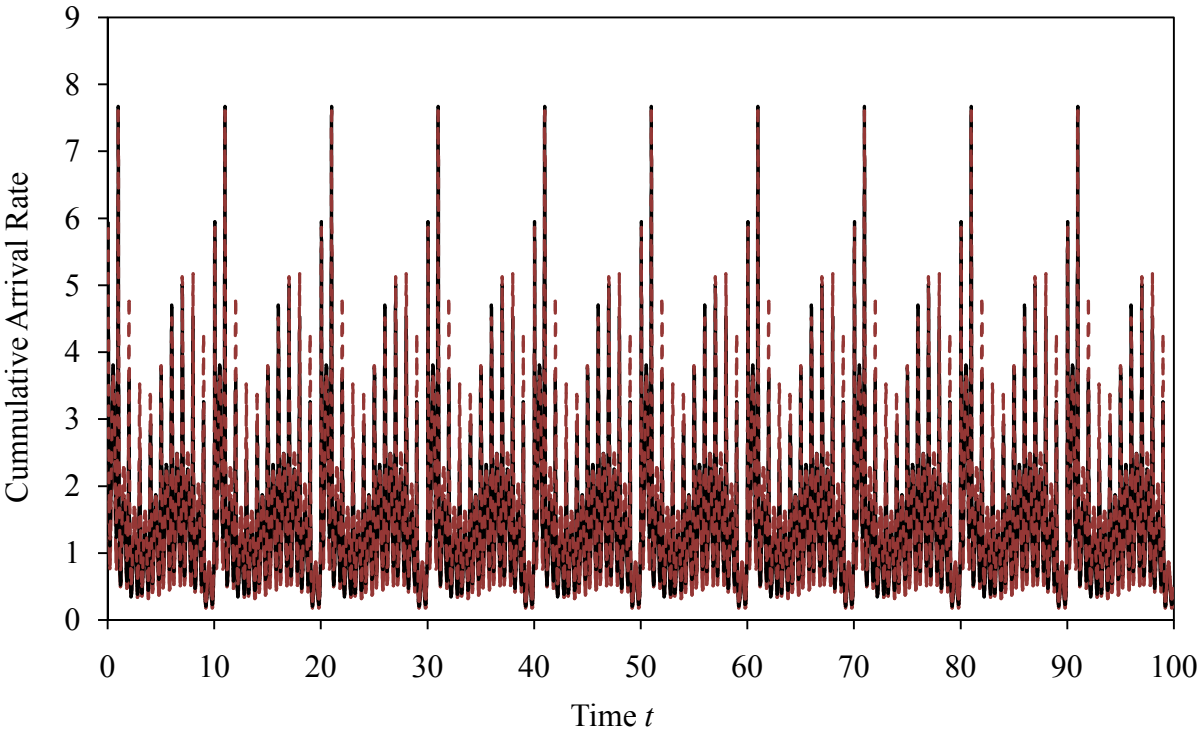

Figure C.6 90% Tolerance Intervals for λ(t),t ∈ [0,100] in Case 2, 15 Realizations

APPENDIX D - SAMPLE CODE EXAMPLES

Deployment Descriptor:

Web.xml file for application defines url patterns for invoking data generation and fitting programs and is shown in Figure D.1

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
<servlet>
<servlet-name>CreateArrivalDataServlet</servlet-name>
<servlet-class>ServletFiles.FitArrivalDataServlet</servlet-class>
</servlet>
<servlet>
<servlet-name>GenerateInputDataServlet</servlet-name>
<servlet-class>ServletFiles.GenerateArrivalDataServlet</servlet-class>
</servlet>
<servlet>
<servlet-name>DisplayExample</servlet-name>
<servlet-class>ServletFiles.DisplayExample</servlet-class>
</servlet>
<servlet>
<display-name>GenerateFromTextServlet</display-name>
<servlet-name>GenerateFromTextServlet</servlet-name>
<servlet-class>ServletFiles.GenerateFromTextServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>CreateArrivalDataServlet</servlet-name>
<url-pattern>/Create</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>GenerateInputDataServlet</servlet-name>
<url-pattern>/Generate</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>DisplayExample</servlet-name>
<url-pattern>/example</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>GenerateFromTextServlet</servlet-name>
<url-pattern>/textGenerate</url-pattern>
</servlet-mapping>

<welcome-file-list>
<welcome-file>start.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

Figure D.1 Web.xml

Java Script functions:

Java script function that performs form field validation is included as example.

```
function doFormValidation(thisform){
        var errorMessage = '';
        var correct = true ;

        //Check for random seed
        if (thisform.randomSeedTextId.value == ''){
                correct = false ;
                errorMessage = errorMessage + 'Please enter random seed. ' ;

        }else {
              if (isNaN(thisform.randomSeedTextId.value)){
                    correct = false ;
                    errorMessage = errorMessage + 'Please enter random seed as a
                                                                    numeric value. ' ;
              }else {
                    if(thisform.randomSeedTextId.value < 0){
                          correct = false ;
                          errorMessage = errorMessage + 'Please enter random seed as a
                                                                    positive numeric value. ' ;
                          }
                    }
        }
if(errorMessage != '')
        alert(errorMessage);
        if(correct)
        thisform.submit();
```

Figure D.2 Java Script Validations

Functionunction doFormValidation() shown in the figure D.2 is used to validate value entered by user in input field random seed. If checks for 3 conditions

a) If a value is entered for random seed.
b) If the value is numerical
c) If the value is positive.

If any of these conditions is not satisfied, the user is alerted with a corresponding error message; the form is not submitted. The validation function ensures submission of correct values to server.

APPENDIX E – MODIFIED LIKELIHOOD ALGORITHM

In section 4.2, a discussion is included about the under fitting phenomenon observed during estimation of the mean-value function for NHPPs where observed number of arrivals is insufficient to estimate mean-value function accurately. To provide theoretical evidence behind the underfitting phenomenon, an example is included in this section. The experimental set up discussed in section 5.3.1 is used to conduct experiments using the test cases in table E.1 for multiple process realizations. The fitting method is used to estimate the mean-value function for hundred replications of the test cases. A linear fit is estimated for majority of the replications for example in test case 1; degree $r = 1$ is fitted to 58 out of 100 replications. It is also observed that, if the fitting procedure does not accept degree $r = 1$, then generally a correct fit of degree $r = 5$ is estimated. Underfitting occurs due to fact that the quadratic fit doesn't improves significantly over the linear fit. Hence, the program is modified as shown in the figure 4.1 which verifies the optimality of linear mean-value function fit to the cumulative NHPP arrivals. The linear fit is confirmed by verifying whether significant improvement is seen at degree $r = 3$ over degree $r = 2$ fit. The comparison between actual degrees and estimated degrees in the table E.2 reinforces the assumption that the misfit generally occurs only due to fit at degree $r = 2$, which lacks substantial improvement over linear fit observed across multiple process realizations. This effect is evident for NHPPs with sparse arrivals. For example, if a single process realization of NHPP available, then the underfitting is prominent as compared to three process realizations in test case 1. Hence the modified likelihood ratio test algorithm helps in better estimation of mean-value function in applications where inadequate data is available about the process.

Table E.1 Input Polynomial Coefficients for Multiple Process Realizations

Case	$\bar{N}(S)$	S	r	Input Coefficients				
				β_1	β_2	β_3	β_4	β_5
1	150	4	5	0.8926201	-1.190605473	0.783775048	-0.2150667	0.02087373
2	50	1	3	1.62999919	-2.69999442	2.06999523	-	-

Table E.2 Comparison between Original and Fitted Degrees for Multiple Realizations

Case	Original Degrees	Realizations	Degree Fitted							
			1	2	3	4	5	6	7	8
1	5	1	58		3	5	34			
		3	35			7	53	5		
		8	17				74	5		4
		15					90	2	1	7
2	3	1	42		54	2	2			
		3	10		72	5	9	3		
		5	2		78	2	13			7
		8			80	3	12			5