11-1-2003

An Automated procedure for simulating complex arrival processes: A Web-based approach

Sachin Sumant

Rochester Institute of Technology

AN AUTOMATED PROCEDURE FOR SIMULATING COMPLEX ARRIVAL PROCESSES: A WEB-BASED APPROACH

A Thesis

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Industrial Engineering

in the

Department of Industrial & Systems Engineering

Kate Gleason College of Engineering

by

Sachin G. Sumant

B.E., Production Engineering, V. J. T. I., University of Bombay, 2000

November, 2003

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 Sachin G. Sumant

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. Moises Sudit

Thesis Reproduction Permission Statement

Permission granted

Title of thesis

AN AUTOMATED PROCEDURE FOR SIMULATING COMPLEX ARRIVAL PROCESSES: A WEB-BASED APPROACH

I, Sachin G. Sumant, hereby grant permission to the RIT Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: 12/04/03 **Signature of Author:** _____

ACKNOWLEDGEMENTS

I am short of words to express my gratitude towards Dr. Kuhl. Not only has he been a phenomenal academic advisor, but also a mentor, assisting me on several occasions not directly related to my academic career.

I would like to thank Dr. Sudit for serving as a member on my thesis committee. I am grateful to Dr. Wilson from North Carolina State University for providing guidance and valuable inputs related to this research. I wish to thank Mr. Madhu Nair for his assistance during my thesis work.

Special thanks to Dr. Mozrall for her support and encouragement during the project.

I appreciate the support of my friends who have been excellent companions and I thank them for making my stay at Rochester Institute of Technology enjoyable. This work is dedicated to my parents and my elder brother for motivating me to come so far and for teaching me to be successful in life.

ABSTRACT

In industry, simulation is one of the most widely used probabilistic modeling tools for modeling highly complex systems. Major sources of complexity include the inputs that drive the logic of the model. Effective simulation input modeling requires the use of accurate and efficient input modeling procedures. This research focuses on nonstationary arrival processes. The fundamental stochastic model on which this study is conducted is the nonhomogeneous Poisson process (NHPP) which has successfully been used to characterize arrival processes where the arrival rate changes over time. Although a number of methods exist for modeling the rate and mean value functions that define the behavior of NHPPs, one of the most flexible is a multiresolution procedure that is used to model the mean value function for processes possessing long-term trends over time or asymmetric, multiple cyclic behavior. In this research, a statistical-estimation procedure for automating the multiresolution procedure is developed that involves the following steps at each resolution level corresponding to a basic cycle: (*a*) transforming the cumulative relative frequency of arrivals within the cycle to obtain a linear statistical model having normal residuals with homogeneous variance; (*b*) fitting specially formulated polynomials to the transformed arrival data; (*c*) performing a likelihood ratio test to determine the degree of the fitted polynomial; and (*d*) fitting a polynomial of the degree determined in (*c*) to the original (untransformed) arrival data. Next, an experimental performance evaluation is conducted to test the effectiveness of the estimation method. A web-based application for modeling NHPPs using the automated multiresolution procedure and generating realizations of the NHPP is developed. Finally, a web-based simulation infrastructure that integrates modeling, input analysis, verification, validation and output analysis is discussed.

TABLE OF CONTENTS

1. INTRODUCTION

Simulation is the process of designing and creating a model of a real or proposed system via a mathematical-logical model, conducting experiments on the model, and drawing inferences from the model about the operation of the real system (Kelton *et al.* 2002, Kuhl 1997). Simulation can be applied to describe and analyze the behavior of a system, analyze "What if...?" questions about systems under study, compare alternative system configurations, optimize system performance, and aid in the design of new systems. Although other operations research tools such as queuing theory, linear programming, etc. can also be used for these types of studies and provide closed form analytical solutions, the simplifying assumptions necessary to use these tools may result in models that do not represent the true behavior of the system. The power of simulation when compared to other analysis tools is in the ability to analyze complex systems. Thus, in the case of complex systems involving variability and interdependencies between system components, simulation provides the capabilities necessary for obtaining valid solutions.

In general, simulation studies can be divided into six basic processes, namely, problem formulation, creation of a conceptual model, creation of a computer model, input modeling, verification and validation, and experimentation. Problem formulation involves developing problem statement, determining the objectives of study and planning the study. After formulating the problem, the next task is to study the real world process and create a theoretical model, which is called the conceptual model. The conceptual model defines the aspects of the system that should be included in the study as well as level of detail needed to accurately model the system. Once the conceptual model is

established, a computer simulation model is constructed. The simulation model, like all other modeling techniques, requires inputs. For simulation, the input modeling procedures encompass all of the activities needed to specify the parameters that drive the simulation model including data collection, analysis, and random variate generation. The verification and validation process consists of checking the operation of the model and determining if the model accurately represents the real system. Finally, experimentation includes designing and performing a set of experiments, analyzing the simulation outputs and drawing the inferences about the system.

Of the six processes mentioned above, one of the most important and influential processes in any simulation study is input modeling. Input modeling performs a role of an interpreter of the real world system for the simulation model. When available, data are collected from the real system, and probabilistic input models are fit to the data by applying various statistical algorithms. Then, the models are used in the simulation model to create random processes representing the system's arrival processes, service times, etc. Since input modeling provides inputs to the simulation model, the overall performance of the study depends heavily on the accuracy of the input models. Although there is a large body of research in the area of input modeling for stationary processes, the derivation of methods for estimating and assessing the goodness of fit of models applicable to nonstationary processes is ongoing (Banks, 1998).

Nonstationary arrival processes are routinely encountered in practice, and thus being able to generate realizations of nonstaionary processes in simulation models is essential. The arrival of customers to a bank, demand for seasonal products such as lawn mowers, arrival of patrons to amusement parks, arrival of patients to emergency rooms,

and the arrival of telephone calls at call centers, are some typical examples of situations where the arrival rate changes over time. One input model that has been used successfully in the past to model a large class nonstationary arrival processes is the nonhomogeneous Poisson process (NHPP). The key to modeling NHPP's is to characterize the arrival rate as the rate changes over time. Both parametric and non-parametric methods have been developed to model the arrival rate (These methods are discussed in detail in Chapter 3). In many simulation studies, nonstationary arrival processes exhibit long-term trends or multiple periodic behaviors. For example, in analyzing the arrival streams of liver-transplant donors and patients for the UNOS Liver Allocation Model, Pritsker *et al.* (1995) observed some arrival rates to reveal significant growth over time as well as daily, weekly and annual effects. To model such type of NHPPs with long term trend and periodic effects, Kuhl and Wilson (2000) derive a flexible theoretical procedure called the multiresolution procedure.

Kuhl and Wilson (2001) present the theory behind the multiresolution procedure and show examples of how their method works. However, these examples require some simplifying assumptions and require the user to determine the adequacy of the fitted model. In this thesis, statistical procedures are developed that can be used to automate the model fitting process for the multiresolution procedure without the need for simplifying assumptions or intervention by the user. The statistical procedures are implemented in a computer program, and an experimental performance evaluation of the procedure is conducted.

The dominance of the Internet in the development of information and technology has made web based distributed solutions increasingly attractive, consequently, in order

to disseminate the automated NHPP fitting procedure, a web-based input modeling interface is developed. Based on a thorough literature review, this is the first web-based input modeling software of its kind. Thus, some unique challenges were encountered in the development of web-based system. Included in the web-based input modeling system is the ability for the user to submit a file containing collected data (arrival times) and to specify parameters of the fitting procedure. Graphs of the fitted NHPP and each of the components of the mutliresolution procedure are plotted versus the observed data. Output files of the parameters are also generated and can be downloaded by the user. In addition, the web-based system can be used to generate realization of the fitted NHPP that could be used in simulation experiments.

Research into the area of web-based input modeling, has led to the investigation of a web-based simulation infrastructure that would allow users to conduct complete simulation studies on the web. The available functions would include the ability to (a) upload a simulation model, or at some point in the future, to build a simulation model on the web; (b) perform verification and validation over the web; (c) conduct input modeling and analysis; and (d) conduct output analysis. The web-based input modeling module is demonstrated through the implementation of the NHPP fitting procedure. Although full development of each of the four main components is beyond the scope of this thesis, the infrastructure for each of the modules will be discussed and demonstrated with simple examples.

2. PROBLEM STATEMENT

The focus of this research is on modeling and simulation of complex arrival process. In particular, the research examines the multiresolution procedure for estimating the mean value function of nonhomogeneous Poisson processes having long term trends or nested cyclic behavior over time with the goal of being able to generate realizations of the NHPPs in simulation experiments. This work extends the capability and functionality of the theoretical basis for the multiresolution procedure established by Kuhl and Wilson (2001) by providing an automated statistical analysis methodology for modeling NHPP that is implemented in a web-based input modeling and generation software. The research work has four main objectives, and each objective has its own unique challenges.

The first objective of this thesis is to automate the multiresolution procedure. The multiresolution procedure estimates mean value function of a NHPP. The mean value function is calculated from the functions fitted to long term trends and all periodic effects (i.e. to all resolutions). Therefore, to automate the multiresolution procedure, the first task is to determine a function, which should not only fulfill multiresolution procedure requirements but also demonstrate ability to model complex trends and cyclic effects in data. Next, a procedure has to be determined to estimate the functions at all the resolutions. After this research, a computer program needs to be developed and tested. Finally, a complex example has to be selected to validate the results of the automated multiresolution procedure.

The second objective of this research involves conducting rigorous experiments on the automated multiresolution procedure to evaluate its performance under various

circumstances. This experimental performance evaluation includes design of a set of experiments, determination of performance measures and finally analysis of the results.

The third objective concentrates on design and development of web-based multiresolution procedure. This applied research encompasses selection of appropriate software and hardware, design and development of the web based infrastructure for input modeling, and finally implementation and testing of the infrastructure with the multiresolution procedure algorithm.

The final objective consists of developing a conceptual web-based simulation environment. According to Kuljis and Paul (2000), web-based simulation researchers are exploring all new directions but still not able to achieve the full advantages of the internet. This thesis work studies the latest web-technologies and their capabilities, analyzes approaches taken in web-based simulation literature, and finally designs a conceptual web-based simulation environment, which can utilize the benefits of the Internet benefits to conduct simulation studies. In addition, the simulation environment infrastructure is demonstrated with some examples.

3. LITERATURE REVIEW

The focus of this research is to automate the multiresolution procedure for modeling nonhomogeneous Poisson processes and to illustrate web-based input modeling by developing a web-based implementation of the multiresolution procedure. To establish a basis for the automation of the multiresolution procedure, simulation input modeling methods incluing nonhomogeneous Poisson processes and their characteristics, parameter estimation procedures, and likelihood ratio tests are discussed. A detailed summary of the multiresolution procedure and its characteristics is also provided. Finally, the current research in the field of web-based simulation is reviewed.

3.1 Simulation Input Modeling

When a simulation model is created, there are many details that must be specified in order to define a valid simulation model. These details include simulation model inputs. The procedures to determine these inputs are referred as input modeling methods.

Figure 3.1 illustrates the input modeling process (Banks, 1998). In general, this input modeling process is the method by which the process and events that occur in the real system are translated into mathematical models that will be used in the simulation program. Thus, each simulation model input corresponds to a process or event in the real-world system. The input modeling process requires combination of data, relevant applicable theory (theory of stochastic processes or related to the system under consideration) and prior experience. The data required for input modeling are collected from the real world. Sampling methods are used if observation data are available. The application of theory on data is completely influenced by the characteristics of the process and the data. Prior experience is useful at every stage of simulation. Prior

experience of the practitioner or expert opinion can be valuable in ascertaining proper understanding of current process, appropriate application of theory and valid results. Various input modeling strategies are developed and used to fit the input model (Nelson, 1995). To be useful, the input model that is developed must be both reasonable and valid. That is, the model must be statistically sound and at the same time be able to accurately represent the system under study. Thus, an assessment of the reasonableness of the resulting model and a validity assessment of the resulting model must be made. The fitted input model is then passed to the simulation program where the random variate generator employs the input model to create realizations of the process.

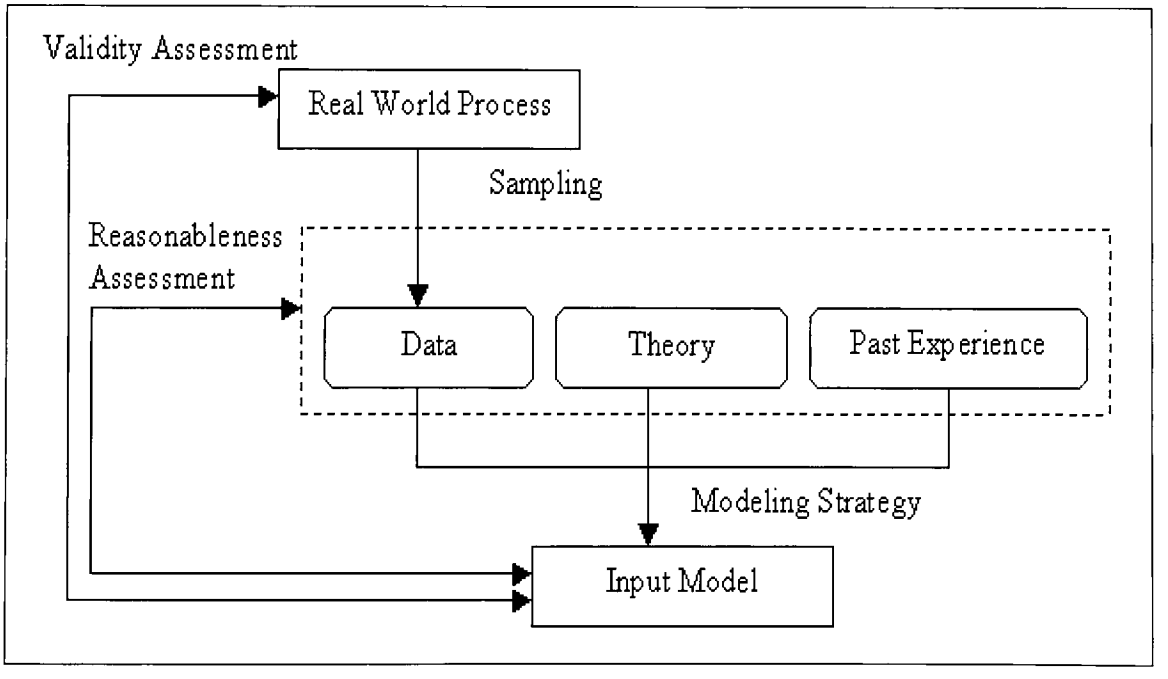

Figure 3.1 Input Modeling Procedures (Banks, 1998)

In practice, there are many different types of processes and data resulting in various modeling strategies. Here, the thesis focuses on processes from which observational data can be obtained to which stochastic process models and theory can be applied.

3.2 Estimating NHPPs

Any process that has a random element is considered a stochastic process. Stochastic process models are ways of statistically analyzing the dynamic relationships of sequences of random events (Cox and Isham, 1980, Taylor and Karlin, 1994). Random processes occur very often in almost every field of natural and engineering sciences. Stochastic modeling can be applied to analyze the variability inherent in manufacturing systems, biological and medical processes, to deal with uncertainties affecting decision making, and with the complexities of psychological and social interactions. In addition, stochastic modeling can provide viewpoints, methods, models and insight to aid in other mathematical and statistical studies.

Stochastic processes can be classified as being stationary or nonstationary. A process is said to be stationary if the process mean and the process variability remain constant over time. A nonstationary process, however, changes over time.

In the case of stationary stochastic processes, a random sample of independent and identically distributed observations can be collected and a probability distribution can be fit to the data. A three–step procedure used to fit theoretical distribution to the data is as follows:

1. Hypothesize a candidate distribution: first, ascertain whether the current process is discrete or continuous.

2. Estimate the parameters of the hypothesized distribution: For example, if the hypothesis is that the underlying data are exponential, the parameter to be estimated from the data is the exponential rate.
3. Perform goodness-of fit test such as the chi-squared test: If the test rejects the hypothesis that is a strong indication that the hypothesis is not true. In that case, go back to step 1, or apply the empirical distribution of the data.

In the case of nonstationary stochastic process, complete observations of the processes over the time interval of interest are needed to fit a nonstationary stochastic model. One model, which has successfully been used to represent a large class of processes, is the nonhomogeneous Poisson process (NHPP). The NHPP is a generalization of the stationary process known as the (homogeneous) Poisson process. Therefore, to aid in discussion, the properties of the (homogeneous) Poisson process will be presented in the next section prior to discussing NHPPs.

3.2.1 Poisson Processes

A Poisson process is a particular type of stochastic process referred to as a counting process, point process, or birth process. Namely, a Poisson process N_t measures the number of randomly occurring events that have occurred in the time interval $(0,t)$. The Poisson process is characterized by the average rate of occurrence of events, λ. In addition, a Poisson process has the following properties:

1. The number of events at the start of the process (time 0) is zero.
2. The process has independent increments. That is, the number of customers arriving during one time interval does not affect the number arriving during a different time interval (non- overlapping).

3. The average number of events that occur in an interval of time is proportional to the length of the interval.
4. Customers arrive one at a time.

Formally, these properties have the following mathematical formulation

1. $N_t = 0$.
2. For any time t, $s \geq 0$, the distribution of $N_{t+s} - N_t$ is independent of t.
3. The average rate means $E[N_t] = \lambda t$ (λ is the average rate).
4. For almost all realizations, for small s, and λ as the rate

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

$$P\{N(t,t+s) = 1\} = \lambda s + o(s); \text{ and}$$

$$P\{N(t,t+s) > 1\} = o(s)$$

for any t, $N_{t+s} - N_t$ is independent of $\{ N_u ; u \leq t \}$.

A stochastic process N_t satisfying the above assumptions is called a Poisson process with rate parameter λ (Cinlar, 1975, Bhat, 1972). Furthermore, the number of events that occur in an interval of time has a Poisson distribution with a probability mass function of the form,

$$p(x) = \frac{(\lambda t)^x e^{-\lambda t}}{x!}, \quad x = 0,1,2,...$$

In addition, the distribution of the time between events has an exponential distribution with a mean of $1/\lambda$ and has the following density function,

$$\lambda e^{-\lambda t}, t \geq 0.$$

The Poisson process is commonly used in simulation to model arrival process, such as arrival of parts to a machine in a manufacturing system, the arrival of customers to an ATM, and the number of defects in a bolt of material.

3.2.2 Nonhomogeneous Poisson Processes

In many practical situations, stochastic processes such as arrival processes are nonstationary. That is, the arrival rate changes over time. Examples of this may include arrivals to a bank over the coarse of a day, demand for lawn mower over the year, and the demand for a product over its life cycle. A model frequently used to model these types of nonstationary processes is the nonhomogeneous Poisson process.

An NHPP $\{N(t), t \geq 0\}$ is a counting process where $N(t)$ is the number of arrivals in the time interval $(0, t]$ where the instantaneous arrival rate at time t, $\lambda(t)$, is a nonnegative integrable function of time and the corresponding (cumulative) mean value function is

$$\mu(t) \equiv \mathrm{E}[N(t)] = \int_0^t \lambda(z)dz \quad \text{for all} \quad t \geq 0.$$

The rate or mean value function of the NHPP completely characterizes the probabilistic behavior of the process (Cinlar, 1972, Cox and Isham, 1980). Furthermore, the following properties hold for an NHPP:

$$N_t(0) = 0 ;$$

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

$$P\{N(t,t+s) = 1\} = \lambda(t)s + o(s) ; \text{ and}$$

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

Therefore, by accurately modeling the rate or mean value function, the NHPP can be completely defined and used to generate realizations of the process in the simulation model. Both parametric and nonparametric methods have been developed to estimate the rate or mean value function from observed data. Leemis (1991) developed a simple nonparametric method to specify a observed piecewise-linear estimate where the breakpoints are determined by the observed arrival times in a superposition of several replicated observations of the process. Lewis and Shedler (1976b) derive techniques for estimating the rate function to model the transactions in a database system. One nonparametric estimator that they derive is

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

where t_0 is the of the time interval, n is the number of observations in $(0, t_0]$, $T_0, T_1, \ldots, T_n$ are the observations, $b(n)$ is a bandwidth function that tends to zero as $n \to \infty$ and W is a bounded, nonnegative, integrable weight function satisfying

$$\int_{-\alpha}^{\alpha} W(u)du = 1.$$

In parametric models, several authors have proposed procedures where the estimated rate function is translated as piecewise-linear or piecewise-polynomial forms. Law and Kelton (2000) discuss the piecewise constant method of modeling an NHPP. This procedure divides the time axis into nonoverlapping time interval where the rate is considered to be constant and estimates a single rate for each interval. Kao and Chang (1988) simulate the times of calls for analysis of electrocardiograms over several days at a hospital using piecewise-polynomial intensity function. Lewis and Shedler (1976a, 1979) model NHPP with log linear rate function and degree two exponential polynomial rate function.

A second approach is to assume that the rate function has some specific functional form that has a sufficient number of parameters to allow it to fit observed data well. Lee, Wilson and Crawford (1991) illustrate a generalized model, which uses an exponential-polynomial-trigonometric function (EPTF) in the rate function as

$$\lambda(t) = \exp\left[\sum_{i=0}^{r} \alpha_i t^i + \gamma \sin(\omega t + \phi)\right]$$

, which they apply to simulate offshore weather events in the Arctic Sea involving both a cyclic component and a trend. Johnson, Lee and Wilson (1994) evaluate the EPTF-type rate function and conclude that the EPTF can adequately model an NHPP with a rate function having a slowly varying long-term trend or a cyclic rate component. Further, they demonstrate that the parameters of EPTF-type rate function can be effectively estimated by using likelihood ratio test to determine the degree of polynomial rate component and by solving the likelihood equations to obtain the maximum-likelihood estimates of the continuous parameters of the rate function. Kuhl, Wilson and Johnson (1997) further generalize this model to include multiple cyclic effects. They propose a parametric model, an EPTMP –type rate function of the form

$$\lambda(t) = \exp\left[\sum_{i=0}^{r} \alpha_i t^i + \sum_{k=1}^{p} \gamma_k \sin(\omega_k t + \phi_k)\right]$$

to model Poisson process having trends or multiple periodicities. This model was implemented in a large-scale simulation model (ULAM) of the organ procurement and transplantation network (Pritsker, 1998). Kuhl and Wilson (2000) formulated and evaluated an ordinary least squares (OLS) procedure for estimating the parametric mean-value function of a NHPP and demonstrated that the estimation accuracy and computational efficiency of the OLS procedure in this type of application is reasonable. During the evaluation process, they use an approximate likelihood ratio test to determine degree of polynomial. Kuhl and Wilson (2001) derive a nonparametric technique titled multiresolution procedure for estimating the (cumulative) mean-value function of a nonhomogeneous Poisson process having a long-term trend or some cyclic effects that

may not have familiar trigonometric characteristics such as symmetry over the corresponding cycles.

The next section focuses on a nonparametric method developed by Kuhl and Wilson (2001) referred to as a multiresolution method for estimating the mean value function of an NHPP from observed data where the process may exhibit behavior including a long term trend over time as well as multiple (nested) cyclic effects. Since this proposal entails automating this procedure (Chapter 4) and implementing it in a web-based environment (Chapter 6), this model is discussed in detail.

3.3 Multiresolution Procedure for Modeling NHPP

The multiresolution procedure (Kuhl and Wilson, 2001) is used to estimate the mean value function of an NHPP having long term trend or multiple periodic components or both. The mean value function is fit data observed over the time interval (0, S], where the data may contain a long term trend or p cyclic effects having periods of length, $b_i (i=1,2,\ldots,p)$, respectively. The observed arrival process must statisfy the following two assumptions:

Assumption 1. There are p distinct cycle lengths (periods) $b_1 > b_2 > \cdots > b_p$ *such that* b_i *is an integral multiple of* b_{i+1} *for i=1,2,...,p-1. Moreover, the time horizon (0,S] is taken such that S is and integral multiple of* b_1*; and let* $b_0 \equiv S$.

Assumption 2. Within the j^{th} *cycle* $[(j-1)b_i, jb_i)$ *of length* b_i *(i = 1,2,..,p) , the arrival rate at time t is proportional to a single baseline function of* $\lambda_i(s)$*, where* $s = t - (j-1)\, b_i$ *is the offset from the beginning of the cycle so that* $\lambda(t) = \alpha_{i,j}\lambda_i(t-(j-1)b_i) = \alpha_{i,j}\lambda_i(s)$*, and* $\{\, \alpha_{i,\ell} : \ell = 1,2,..., S/b_i \,\}$ *are the constants of proportionality for all cycles of length* b_i.

To estimate the mean value function $\mu(t)$, the function $R_i(s)$ at each resolution for $s \in [0, b_i)$ and $i = 0,1,2,..., p$ needs to be estimated first. Note that at each resolution, $\hat{R}_i(0) = R_i(0) = 0$ and $\hat{R}_i(b_i) = R_i(b_i) = 1$ for $i = 0,1,2,..., p$. At resolution 0, a monotonically increasing function $\hat{R}_0(t), t \in [0, S]$, is fitted to the points, $\{ [jb_1, N(jb_1 / N(S)]^T : j = 0,1,..., S/b_1 \}$ such that $\hat{R}_0(0) = 0$ and $\hat{R}_0(S) = 1$.

For resolution i for $i = 1,2,..., p-1$, a monotonically increasing function $\hat{R}_i(s)$ is constructed to estimate the cumulative proportion of arrivals that are expected to occur during the first s time units of the cycle, where $s \in [0, b_i)$. That is, for resolution I, $\hat{R}_i(s)$ is fitted to the points $\left\{ [jb_{i+1}, G_i(jb_{i+1})]^T : j = 0,1,..., b_i / b_{i+1} \right\}$, where

$$G_i(s) = \frac{1}{N(S)} \sum_{\ell=0}^{(Sb_i)-1} [N(\ell b_i + s) - N(\ell b_i)]$$

for all $s \in [0, b_i)$.

At resolution p, let $\tau_k : k = 1,2,...N(S)$ denote the observed arrival times and define $\eta_k = \tau_k \bmod b_p$ for $k = 1,2,..., N(S)$. Let $\eta_{(k)} : k = 1,2,..., N(S)$ denote the ordered arrival times on the interval $[0, b_p)$. A monotonically increasing function is fitted to the points $\left\{ [\eta_{(k)}, k / N(S)]^T : k = 1,2,..., N(S) \right\}$.

The final estimate of the mean value function $\hat{\mu}(t)$ is computed as follows:

$$\hat{\mu}(t) = N(S)\hat{Q}_0(t) \text{ for } t \in [0, S]$$

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

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

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

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

where $j_{i,t}$ is a unique integer j such that $(j-1)b_i \le t < jb_i$.

To illustrate the multiresolution procedure, an example of arrival process whose instantaneous arrival rate over a time horizon of 28 days contains two cyclic effects including weekly and daily cyclic effects that is, periodic rate components with periods of 1 week and one day is considered. The NHPP is constructed so as to satisfy all the assumptions discussed in the previous part, where $\mu(S) = 216$ and

Resolution 0:

$R_0(s) = 0.0357143s \qquad s \in (0,28]$

Resolution 1:

$R_1(s) = 0.258203s - 0.0164780\,s^2 \quad s \in (0,7]$

Resolution 2:

$$R_2(s)= 1.83200s-0.890900\ s^2 -0.967333s^3+ 1.02624\ s^4 \qquad s \in (0,1]$$

Data for realization of arrival process is generated using the simulation algorithm described by Kuhl and Wilson (2000). Application of the multiresolution procedure started by estimating the function $R_i(s)$ corresponding to each resolution i for $i = 0,1,\ldots,p$. Since the NHPP defined above consists of two periodic components, the procedure estimated the function $R_i(s)$ at: (a) resolution 0, corresponding to the long-term trend or with a period of 28 days; (b) resolution 1, corresponding to the cyclic rate component with a period of b_1 =7 days; and (c) resolution 2, corresponding to the cyclic rate component with a period of b_2 = 1 day. While estimating R_i(s), the degrees of polynomials are specified by the user and assumed to be known.

For estimating mean-value function, the points of importance at resolution 0 are those that correspond to the cumulative fraction of arrivals observed at the end of each week. At resolution 0, 1- degree polynomial was fitted to the fraction of all arrivals that were accumulated at the end of each week. The estimated function at resolution 0, is shown in Figure 3.1. The points of importance at resolution 1 are those that correspond to the cumulative fraction of arrivals observed at the end of each day (that is, at the end of each resolution-2 cycle). In this example, a quadratic function was fitted to the cumulative fraction of arrivals observed at the end of each day of the week. The estimated function at resolution1 is shown in Figure 3.2. The next step in the procedure was to estimate the resolution-2 function $R_2(s)$ associated with a period of 1 day. In this

example, all the arrivals are important points. A fourth degree polynomial is fitted to the observed arrival times. The estimated function at resolution 2 is shown in Figure 3.3.

The fitted polynomials at each resolution are:

Resolution 0:

$R_0(s) = 0.0357143s \qquad s \in (0,28]$

Resolution 1:

$R_1(s) = 0.260803665s - 0.0168495\, s^2 \qquad s \in (0,7]$

Resolution 2:

$R_2(s) = 1.87299418s - 1.32037175\, s^2 - 0.183941s^3 + 0.631318599\, s^4 \qquad s \in (0,1]$

When all the functions are estimated, next task was to estimate the mean value function. The process of estimating the overall mean value function started by estimating the mean value function at resolution 0, $\mu_0(t)$, for $t \in [0,28]$. To complete this,

$$\mu_0(t) = N(S)\, Q_{0,0}(t) = N(S)\, R_0(t) \quad \text{for all } t \in [0,28]$$

was calculated.

After the calculation of resolution 0 component, next task was to include the details of the weekly periodic component to the estimated mean value function. This was done by calculating $\mu_1(t)$, for $t \in [0, 28]$.

Applying the equations, the following estimate of mean value function was acquired for the first week:

$$\mu_1(t) = N(S)\, Q_{0,1}(t) = N(S)\, R_0(7)\, Q_{1,1}(t)$$

$$= N(S)\, R_0(7)\, R_1(t) \text{ for all } t \in [0,28].$$

The final step was to include the details of the daily periodic component to the estimated mean value function calculating $\mu_2(t)$, for $t \in [0,28]$. Applying the mean value function equations, the following estimate of mean value function was acquired for the first day:

$$\mu_2(t) = N(S)\, Q_{0,2}(t) = N(S)\, R_0(7)\, Q_{1,2}(t)$$

$$= N(S)\, R_0(7)\, R_1(1)\, Q_{2,2}(t)$$

$$= N(S)\, R_0(7)\, R_1(1)\, R_2(t)$$

for all $t \in [0,28]$. The mean value function was calculated at resolution 1 for subsequent weeks and at resolution 2 for subsequent days. Since the process consists of only two periodicities, the final estimate of the mean value function was $\mu_2(t)$. The fitted mean value function plotted against the observed cumulative number of arrivals over the entire time horizon is shown in Figure 3.4.

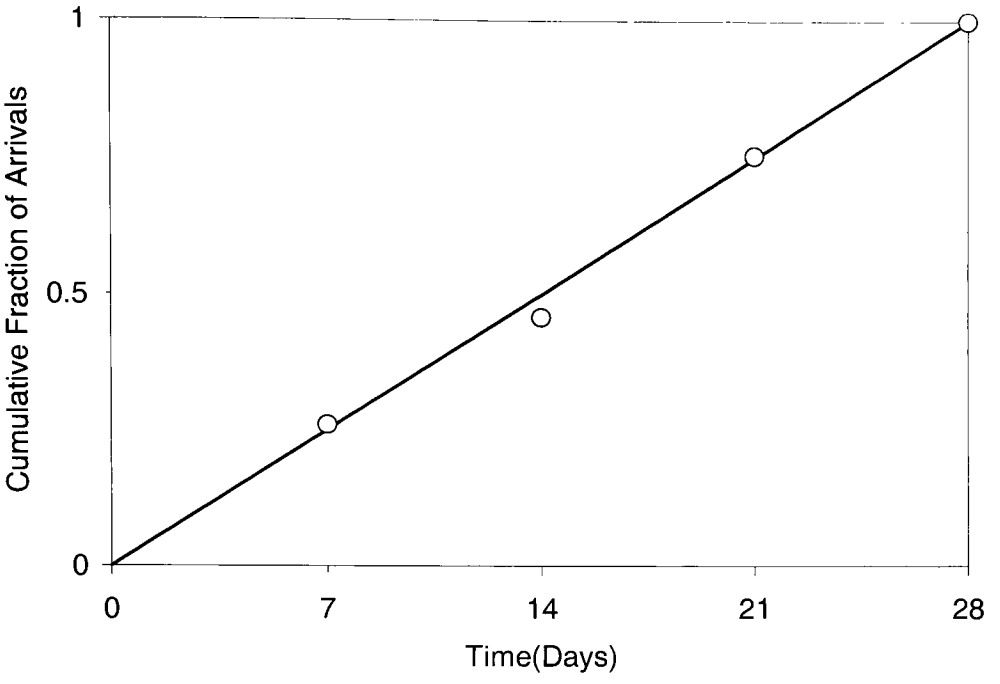

Figure 3.1 Estimated Linear Function at Resolution 0

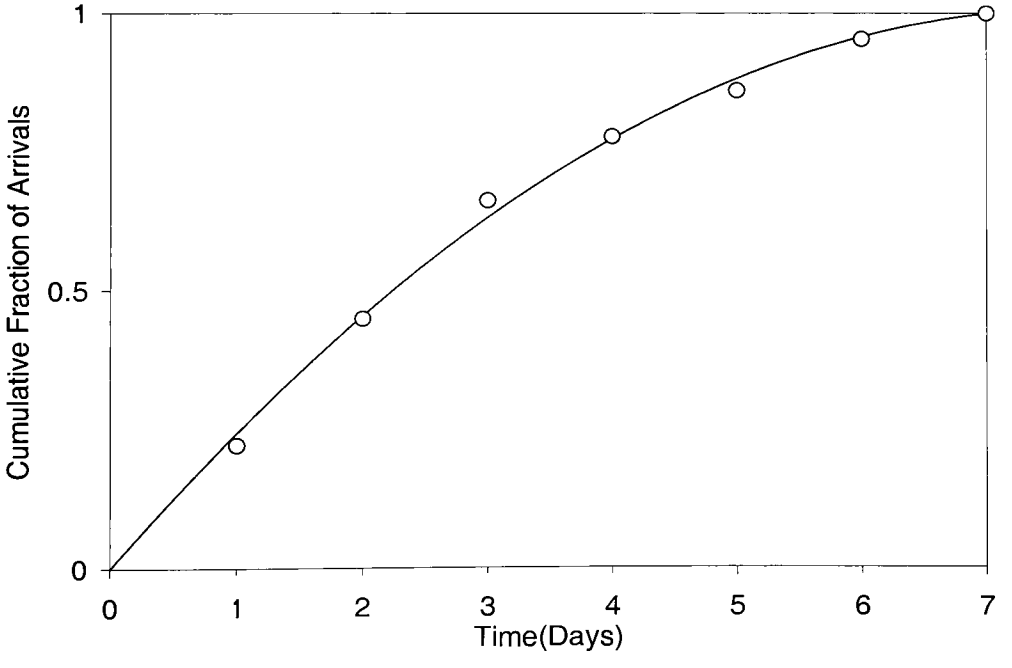

Figure 3.2 Estimated Quadratic Function at Resolution 1

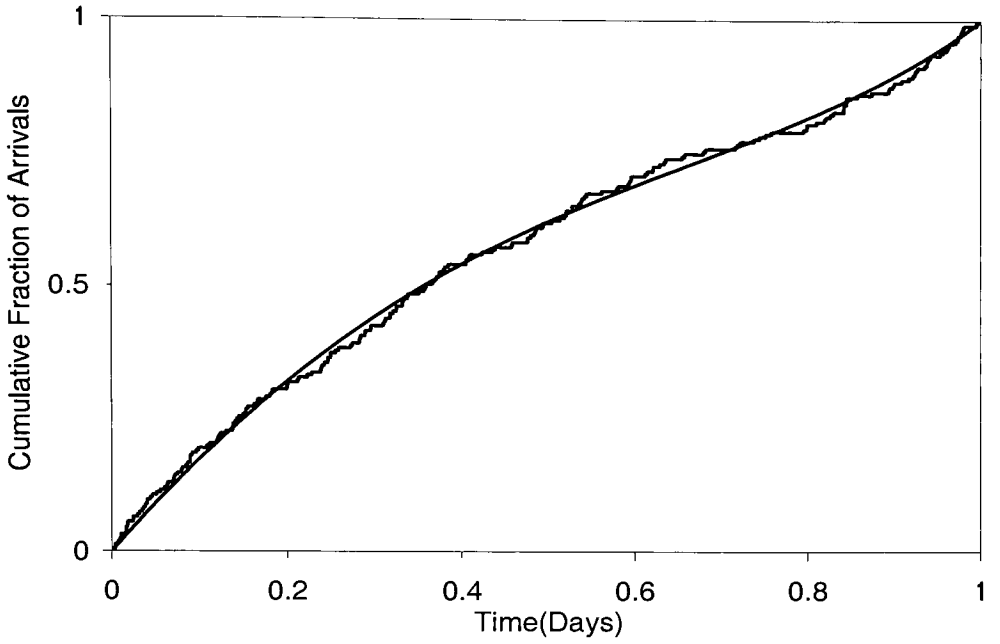

Figure 3.3 Estimated Quartic Function at Resolution 2

Kuhl and Wilson (2000) applied the method of inversion to generate realizations of the fitted function while simulating NHPPs having a multiresolution type mean value function. Clearly the efficiency of this simulation scheme depends on the numerical methods applied to invert estimates of the functions. A nonparametric estimate of the mean value function of an NHPP having long-term trend or cyclic effects that may exhibit nontrignometric characteristics can be obtained by the multiresolution procedure. The multiresolution procedure allows us to model asymmetric periodic behavior with no need to store all of the observed data.

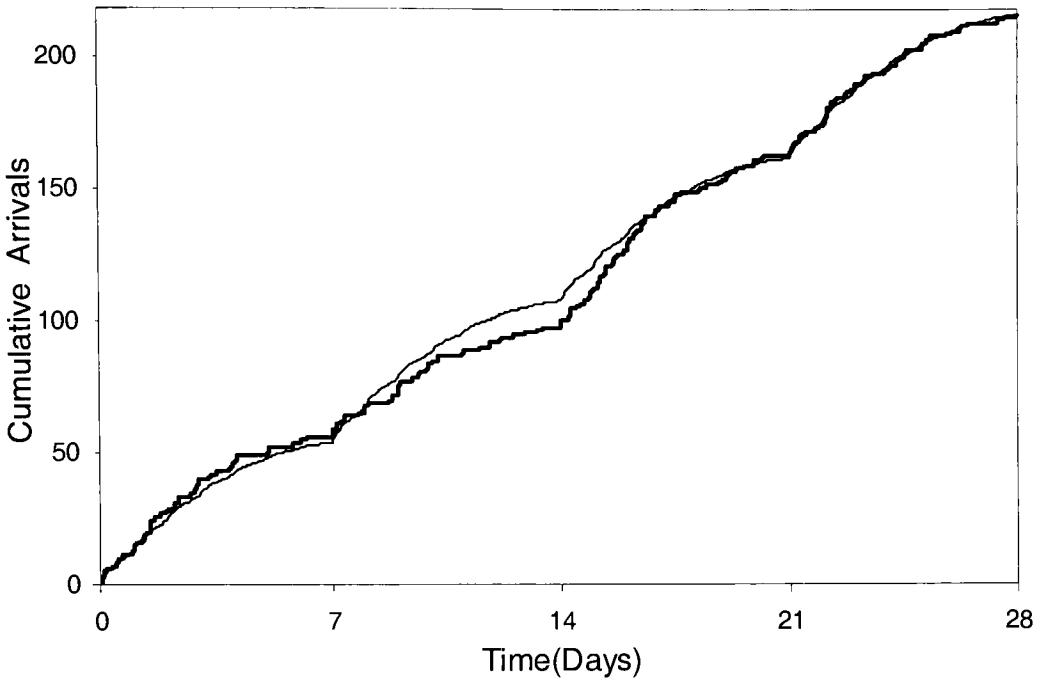

Figure 3.4 Estimated Mean Value Function (smooth curve) Superimposed on the Observed Cumulative Arrival Function (Step function)

3.4 Estimation Procedures

In order to automate the multiresolution procedure described in the last section, this research designs a method, which includes a likelihood ratio test (Chapter 4). Therefore, we will review, in this section, maximum likelihood estimation, hypothesis testing, and likelihood ratio tests.

3.4.1 Maximum Likelihood Estimation

One of the most commonly used estimation method is maximum likelihood estimation. Maximum likelihood estimation has been described as the original jackknife because of its applicability to a wide variety of problems (Breiman, 1973). The maximum likelihood estimate is defined as the estimate of parameter vector θ in the parameter

space ψ that will maximize the value of the likelihood function. Suppose that $X = (X_1, \ldots, Xn)$ are random variables with joint density or frequency function $f(x:\theta)$ where $\theta \in \psi$. Given the outcomes $X = x$, the likelihood function is

$$L(\theta;x) = f(x;\theta)$$

(Casella and Berger, 1990, Ross, 1989). The distinction between these two functions is the variable that is considered fixed and the one that is varied. When the pdf $f(x;\theta)$ is considered, θ is considered as fixed and x as the variable; when the likelihood function $L(\theta;x)$ is considered, x is considered as fixed and θ as the variable.

If the data range does not depend on the data, the parameter space ψ is an open set, and the likelihood function is differentiable with respect to $\theta = (\theta_1 \ldots\ldots \theta_p)$ over ψ, then the maximum likelihood estimate (θ) satisfies the equations

$$\frac{\partial \ln(L(\hat{\theta}))}{\partial \theta_k} = 0 \quad \text{for} \quad k = 1,2,\ldots., p\,.$$

These equations are called likelihood equations and $\ln(L(\hat{\theta}))$ is called log-likelihood function (Knight, 2000).

There are two inherent drawbacks associated with the general problem of finding the maximum of a function, and hence of maximum likelihood estimation. The first problem is that of actually finding the global maximum and verifying that, indeed, a differential calculus exercise but sometimes even for common densities, difficulties do

arise. The second problem is that of numerical sensitivity. It is sometimes the case that a slightly different sample will produce a totally different MLE, making its use suspect.

In some cases, function $R(\theta)$ of parameters θ needs to be estimated. In such situation, one of the most important properties of maximum likelihood estimators what has come to be known as the invariance property of MLE is applied. The invariance property states that if $\hat{\theta}$ is the MLE of θ, then for any function $R(\theta)$, the MLE of $R(\theta)$ is $R(\hat{\theta})$(Casella and Berger, 1990).

Maximum Likelihood estimation provides the theoretical basis on which statistical tests such as hypothesis tests (and in particular likelihood ratio tests) can be developed. Therefore, the idea of hypothesis testing is discussed first in the next section and then the likelihood ratio test is elaborated.

3.4.2 Hypothesis Testing

A statistical hypothesis is a statement about the parameters of one or more populations and the decision-making procedure about the hypothesis is called hypothesis testing. Statistical hypothesis testing can be divided into six steps. The first step in any type of hypothesis testing is to identify the parameter of interest. In the current problem domain the diameter of rod is the parameter of interest. The second step is to state the null hypothesis and to specify the alternative hypothesis. In the third step, a significance level is selected. As the significance level increases, the precision in the decision decreases. Here, the significance level is the probability of obtaining accurate estimation .In the fourth step, an appropriate test statistic is determined. In the next step, the test statistic is computed by calculating all the necessary sample quantities. The last step of

hypothesis testing comprises whether to accept or reject the null hypothesis and to represent that in the problem context.

A widely used method of test construction is the likelihood ratio principle. Statistical methods obtained by using likelihood ratio principle often turn out to have smallest type II error among all tests that have the same type I error probability (Nelson, 1995). These tests are called as likelihood ratio tests (LRT).

3.4.3 Likelihood Ratio Tests

The likelihood ratio method of hypothesis testing is related to the maximum likelihood estimators, and the likelihood ratio tests are as widely applicable as maximum likelihood estimation. Recall that if $X_1, \ldots, X_n$ is a random sample from a population with probability distribution function $f(x;\theta)$, the likelihood function is defined as

$$L(\theta; x_1, \ldots, x_n) = L(\theta; x) = f(x;\theta) = \prod_{i=1}^{n} f(x_i;\theta).$$

The likelihood ratio statistic χ is defined as

$$\chi(x) = \frac{\underset{\psi_1}{\text{Sup}}\, L(\theta; x)}{\underset{\psi}{\text{Sup}}\, L(\theta; x)},$$

where L(θ;x) is a likelihood function. A likelihood ratio test is $H_0 : \theta \in \psi_1$ versus $H_1 : \theta \in \psi_1'$ where $\psi = \psi_1 \cup \psi_1'$. The likelihood ratio test for ψ_1 consists of fixing a

critical ratio value c and accepting if $\chi(x) \geq c$ (Casella and Berger, 1990, Brieman, 1973). Even if the two suprema of $L(\theta,x)$, over the sets ψ_1 and ψ, can not be analytically obtained, they can usually be computed numerically.

The rationale behind LRT may best be understood in the situation in which $f(x;\theta)$ is the pdf of a discrete random variable. In this case the numerator of $\chi(x)$ is the maximum probability of the observed sample, the maximum being computed over parameters in the null hypothesis. The denominator of $\chi(x)$ is the maximum probability of the observed sample over all possible parameters. The ratio of these two maxima is small if there are parameter points in the alternative hypothesis for which the observed sample is much more likely than for any parameter point in the null hypothesis. In this situation, the LRT criterion says H_0 should be rejected and H_1 accepted as true (Knight, 2000).

In this thesis work, a likelihood ratio test is designed to determine the degree of a polynomial. This test automates the multiresolution procedure.

3.5 Web-Based Simulation

After reviewing web based simulation research papers from 1996-2001, it is observed that the speed in which the web based simulation is progressing, and the desire to create complex simulation models and perform simulation operations on web will not be possible in immediate future. However, According to Kuljis and Paul (2000), there is certainly promise for certain type of applications to utilize the advantages of Internet computing and the web. The most likely candidates are applications that deal with huge quantities of data such as meteorological modeling; applications that enable users on multiple sites to collaborate in model design; applications that require direct input from a

customer (for example, manufacturing which offers customized products) or applications in education and training which increasingly have to cater to distance learning students.

As far as web based simulation is concerned, there are currently two main approaches that are both based on the Java language. One approach develops a new simulation language using Java, and the other approach ports an existing simulation language, such as GPSS, and creates it in Java (Whitman 1998). Advances in Java development tools and JavaBeans technology that can facilitate developments of reusable components in combination with the ability to distribute and execute models via the Internet may foster increased activity in the development of high-level, domain-specific simulation tools aimed at the end users.

The integration of the web and Java represents a technological advancement that enables a fundamentally new approach to simulation modeling. Healy (1998) attributes to the role, the Java language plays in both the specification and implementation of the model. Paul (2000) explains how Java is beneficial for web-based simulation. Kilgore (1998) lists Java features that have the potential to dramatically change the process used to build computer simulation models.

Kuljis and Paul (2000) discuss several Java based discrete simulation environments and conclude that developments are surprisingly conservative and lack the very entrepreneurial nature of media they are trying to conquer. Most of the work focuses on the tools that are being used (Java, Javabeans etc.) rather than how the new medium can enhance our use of simulation as a modeling vehicle. As it is, most applications are "invented" and answer the question " What can I put on the web" rather than "This

simulation has to be available on the web, how can I design to facilitate the needs of its users."

The main architectures of web-based simulations have been shown and tested successfully during recent years (Fishwick 1996, Healy and Kilgore 1997). Wiedemann (2001) compares common web technologies with simulation characteristics as shown in Table 3.1. The comparison reveals large differences concerning all criteria. A simple transformation of actual simulation techniques into the web will result in the same disadvantages plus some new restrictions and problems caused by the web itself. The main cause for the differences consists the variability in the different types of operations. A typical Internet surfer essentially works in a passive, assimilating mode. His creative work is limited to decisions about selecting the next interesting link or new combinations of search criteria. A wrong decision can be promptly corrected by pressing the "Back" button. Modeling, simulation and result analysis require a high level of active and creative work. A problem of web based simulation systems is the high complexity of their user interfaces.

Fishwick (1996) has divided the web based simulation in three topics: 1) Education and Training 2) Publications 3) Simulation Programs and discusses procedures for implementing web technologies and concepts in simulation modeling. Yu-hui Tao and Shin-Ming Guo (2001) designed a web based training system for simulation analysis. According to Pidd, Oses, Brooks (1999), various forms of distributed simulation are possible over the world wide web, including simple multiple replications of the same model, client server architectures for one or more simultaneously running models and the distributed operation of one or more linked models. In addition, they suggest a

component-based approach for web-based simulation. Alfonseca (2000, 2001) has built and executed distributed web based models.

Table 3. 1. Comparison of Web and Simulation Technology Characteristics

	Common web technologies	**Traditional simulation technologies**
Common Standards	Yes (HTML, XML, TCP/IP)	No common standard for model and results
Data handling	By unique URL	Proprietary
Information Structures	Tree structures and lists	Very complex
Specialist Knowledge	No (only clicking and reading)	Yes (from a lot of scientific areas)
Navigation	Easy	Difficult
Ease of Use	Very good	Very difficult
Type of Operations	Read and analyze information	Synthesize models and analyze results.

Seila and Miller (1999) present a design for scenario management in web based simulation. The objective of the design is to perform output analysis on the web. The JSIM environment provides the requirements for scenario management and output analysis. The web based interface created by Guru, Savory, Williams (2000) connected SIMAN server, Oracle server and Application server to allow users to store and execute SIMAN simulation models over the Internet. Bodner, Govindraj (2000) connected Arena, CORBA and JAVA applets to design a web-based simulation environment to support modeling and design of manufacturing and material handling systems.

Current Java based simulation languages are Simjava (Howell and McNab 2000), DEVSJAVA (Sarjoughian and Zeigler 1998), JSIM (Miller 1998), Javasim, JavaGPSS (Klein 1998), Silk (Healy and Kilgore 1997). The WSE (web-enabled simulation

environment) combines web technology with the use of Java and CORBA (Iazeolla and D'Ambrogio 1998). Most of the languages are Java versions of existing simulation languages.

The concept of open source modeling (Wiedemann, 2001) represents an opportunity to improve the quality of common core simulation functions, improve the potential for creating reusable modeling components using XML, HLA and other simulation community standards. It aims at leveraging the unique communication and distribution opportunities created by the Internet to open the development of simulation software to a worldwide community of talented software developers and researchers. This concept is at extremely early stage of development.

In contrast, current object oriented commercial simulation languages are capable of modeling any type of complex systems. Benson (1997) illustrated ProModel as a powerful yet easy to use simulation tool for modeling all types of manufacturing systems ranging from small job shops and machining cells to large mass production, flexible manufacturing systems, and supply chain systems. Most of the languages are windows based systems with intuitive graphical interfaces and object-oriented modeling constructs that eliminate the need for programming. Swets and Drake (2001) have introduced the new Arena product family, which allows the modeler to create discrete, continuous as well as combined models. Arena combines the ease of use found in high-level simulators with the flexibility of simulation languages, and even all the way down to general purpose procedural languages such as the Microsoft VB programming system or C if the user needs. The hierarchical structure of Arena allows user to shift from lower level programming to higher-level templates depending on the requirement. From the current

discussion, it can be concluded that current commercial modeling languages are more user friendly as compared to web-based simulation environments. They allow the modeler to construct complexities in the model and the level of flexibility is high for animation, output analysis and other simulation study requirements.

In Chapter 5 and Chapter 6, a web-based simulation infrastructure is demonstrated utilizing current web technologies and current web-based simulation languages. This thesis primarily concentrates on development of web-based input modeling environment.

4. AN AUTOMATED MULTIRESOLUTION PROCEDURE FOR MODELING NHPPS

The objective of this chapter is to develop a method for automating the multiresolution procedure for modeling NHPPs. The multiresolution procedure, as discussed in Chapter 3, requires several components to fully automate the procedure including the selection and estimation of a function to be fit at each resolution. In summary to automate the procedure, a special form of a polynomial is selected to fit at each resolution. The polynomial at each resolution is estimated using ordinary least square procedure (Kuhl *et al.* 2000). A likelihood ratio test is designed to determine the degree of each polynomial. The likelihood ratio test requires the data, which have normalized residuals. However, Kuhl and Wilson (1997) show that the data observed from a nonhomogeneous Poisson process do not possess normalized residuals. Therefore, to apply likelihood ratio test, the original observed data from the nonhomogeneous process is transformed using an arcsin square root transformation to obtain data with normalized residuals. This transformed data is utilized only to determine the degrees of polynomials. Once the degrees are determined, these degrees are used to estimate polynomials from original data. These polynomials are utilized to calculate the mean value function to represent the NHPP. The derivation along with a demonstration of the automated multiresolution procedure with the likelihood ratio test is discussed below.

4.1 Input Modeling and the Need for Automation

In the words of Friedrich Engels (1820-1895), "An ounce of action is worth a ton of theory." This statement can be applied effectively to many fields of research. In the context of input modeling, the best theoretical models can be developed for simulation

input modeling, however, without a method developed for implementation; the model may not be used in actual practice. Due to the high cost and large amount of time required for implementation, the use of a model may become economically prohibitive. Kuhl and Wilson (2001) establish the theoretical basis and general methodology for modeling NHPP's using the multiresolution procedure. However, for a simulation analyst to apply the multiresolution procedure in practice, the analyst must complete the following tasks:

1. Obtain an appropriate function that follows the theoretical requirements;
2. Obtain an appropriate method for fitting the function to data at each resolution;
3. Verify that the model meets the theoretical assumption of the procedure as well as the assumption of an NHPP;
4. Validate the model;
5. Write computer code for the fitting procedure; and
6. Apply the model to the system under study.

Therefore, this research work develops, tests and implements an automated procedure that will satisfy the first 5 criteria, leaving the user only with applying the model to the system under study.

4.2 Selection of an Appropriate Function for $R_i(\cdot)$

Recall that in the multiresolution procedure, at each resolution level i, an estimator $\hat{R}_i(s)$ of the function $R_i(s)$ must be obtained with the following properties: the initial value $\hat{R}_i(0) = R_i(0) = 0$; the final value $\hat{R}_i(b_i) = R_i(b_i) = 1$; and the derivative $\hat{R}_i'(s) > 0$ for $s \in [0, b_i]$ and $i = 0, 1, 2, \ldots, p$.

At resolution level 0, a monotonically increasing function $\hat{R}_0(t)$, $t \in [0, S]$ is fitted, to the points $\left\{ [\, jb_1,\ N(jb_1)/N(S)]^{\mathrm{T}} : j = 0, 1, \ldots, S/b_1 \right\}$. At resolution level i for $i = 1, 2, \ldots, p-1$, a monotonically increasing function $\hat{R}_i(s)$ is constructed to estimate the cumulative percentage of arrivals that are expected to occur during the first s time units of the associated cycle of length b_i, where $s \in [0,\ b_i]$. That is, for resolution level i, the function $\hat{R}_i(s)$ is fitted to the points $\left\{ [\, jb_{i+1},\ G_i(jb_{i+1})]^{\mathrm{T}} : j = 0, 1, \ldots, b_i / b_{i+1} \right\}$, where

$$G_i(s) = \frac{1}{N(S)} \sum_{\ell=0}^{(S/b_i)-1} [N(\ell b_i + s) - N(\ell b_i)] \tag{1}$$

for all $s \in [0,\ b_i]$.

At resolution p, let $\left\{ \tau_k : k = 1, 2, \ldots, N(S) \right\}$ denote the observed arrival times; and define $\eta_k = \tau_k \bmod b_p$ for $k = 1, \ldots, N(S)$. Let $\eta_{(1)} \le \cdots \le \eta_{(k)}$ denote the ordered arrival times within the level-p cycle $[0,\ b_p]$. Then a monotonically increasing function $\hat{R}_p(s)$ is fitted to the points $\left\{ [\eta_{(k)},\ k/N(S)]^{\mathrm{T}} : k = 1, \ldots, N(S) \right\}$ subject to the usual boundary conditions that $\hat{R}_p(0) = 0$ and $\hat{R}_p(b_p) = 1$. The final estimate $\hat{\mu}(t)$ of the mean value function is computed as follows:

$$\hat{\mu}(t) = N(S)\hat{Q}_0(t) \text{ for } t \in [0, S],$$

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

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

and for $i = p-1, \ldots, 1, 0$, take

$$\begin{aligned} \hat{Q}_i(t) = \hat{R}_i((j_{i+1,t} - 1)b_{i+1} - (j_{i,t} - 1)b_i) \\ + \left[\hat{R}_i(j_{i+1,t}b_{i+1} - (j_{i,t} - 1)b_i)\right. \\ \left. - \hat{R}_i((j_{i+1,t} - 1)b_{i+1} - (j_{i,t} - 1)b_i)\right] \hat{Q}_{i+1}(t), \end{aligned} \tag{2}$$

where the interval containing t is within each resolution is given by the index,

$j_{i,t}$ is the unique integer j such that $(j-1)b_i \le t < jb_i$.

In order to construct the mean value function utilizing the multiresolution procedure, the functions $\{\hat{R}_i(s): i = 0,1,2,\ldots,p\}$ at each resolution must be selected. Therefore, a special form of a polynomial that meets these requirements—namely, a degree r polynomial of the form

$$\hat{R}_i(s) \equiv \begin{cases} s / b_i, & \text{if } r = 1, \\ \sum_{k=1}^{r-1} \beta_k (s / b_i)^k + \left(1 - \sum_{k=1}^{r-1} \beta_k\right)(s / b_i)' & \text{if } r > 1, \end{cases} \tag{3}$$

for $s \in [0, b_i]$ is utilized for $i = 0, 1, 2, \ldots, p$. The coefficients $\{\beta_k : k = 1, \ldots, k-1\}$ in (3) are constrained to yield $\hat{R}_i'(s) > 0$ for all $s \in [0, b_i]$. This polynomial automatically satisfies the required boundary conditions. Further, the simplest process that can be modeled is the one with a constant arrival rate over the interval $[0, b_i]$, which corresponds to the simplest form of the model (3) wherein $r = 1$ is taken (that is, a linear mean value function).

4.3 Estimating $R_i(\cdot)$

To estimate the function (1) by a fitted function $\hat{R}_i(s)$ of the form (3) at each resolution level i, the appropriate degree r of the polynomial (3) must be determined and then the associated polynomial coefficients $\beta_1, \beta_2, \ldots, \beta_{r-1}$ are estimated. The standard approach for solving such a problem is to perform regression analysis using a forward selection or backward elimination procedure (Draper and Smith 1998).

In a forward selection procedure, polynomials of degree $r - 1$ and r are fitted to the sample data; then the mean square error is computed for each fit and an appropriate likelihood ratio test is performed to determine if significant improvement in the fit is achieved by increasing the degree of the fitted polynomial from $r - 1$ to r. If a significantly better fit is obtained with the degree-r polynomial, then the fitting procedure is continued by incrementing the value of r and repeating the likelihood ratio test; otherwise, the fitting procedure is terminated, delivering $r - 1$ as the degree of the fitted polynomial.

However, the assumptions of classical regression analysis require observations of a dependent variable that are independent and normal with a constant variance so that the corresponding residuals are independent and identically distributed normal random variables. In the case of fitting arrival data obtained from an NHPP that satisfies Assumptions 1 and 2, the observed cumulative relative frequencies within each basic cycle are neither normal nor independent; moreover, such responses do not possess a constant variance. Therefore, a variance stabilizing transformation must be applied to the data before standard statistical procedures to determine an appropriate degree r for the function $\hat{R}_i(s)$ are used.

4.4 Likelihood Ratio Test Procedure

At each resolution level i, let $m_i \equiv b_i / b_{i+1} - 1$ for $0 \le i \le p-1$ and $m_i \equiv N(S)$ for $i = p$; and a polynomial function $\hat{R}_i(s)$ of the form (3) is fitted with appropriate degree r to a set of points having the general form

$$\left\{ (Z_j, W_j)^{\mathrm{T}} : j = 1, \ldots, m_i \right\} \tag{4}$$

as defined in §1. For example if the resolution level i is in the range $\{1, \ldots, p-1\}$, then at the jth point in (4) the abscissa $Z_j = jb_{i+1}$ and the ordinate $W_j = G_i(jb_{i+1})$ for $j = 1, \ldots, m_i$ are taken. Since W_j is always a proportion, the variance stabilizing transformation

$$Y_j \equiv \sin^{-1}\left\lfloor \sqrt{W_j} \right\rfloor \quad \text{for} \quad j = 1, \ldots, m_i ; \tag{5}$$

is exploited. See pp. 231–238 of Box, Hunter, and Hunter (1978) and pp. 291–294 of Draper and Smith (1998).

Corresponding to the dependent variable Y_j defined by (5), the independent variable

$$X_j \equiv Z_j / b_i = jb_{i+1} / b_i \quad \text{for } j = 1, \ldots, m_i; \tag{6}$$

is defined and in terms of the vector of regression coefficients

$$C_r \equiv \begin{cases} \pi/2, & \text{if} \quad r = 1, \\ [C_1, C_2, \ldots, C_{r-1}], & \text{if} \quad r \geq 2, \end{cases} \tag{7}$$

for the degree-r polynomial

$$\Gamma_r(u; C_r) \equiv \begin{cases} \left(\frac{\pi}{2}\right)u, & \text{if } r=1, \\ \sum_{k=1}^{r-1} C_k u^k + \left(\frac{\pi}{2} - \sum_{k=1}^{r-1} C_k\right)u^r, & \text{if } r>1, \end{cases} \tag{8}$$

The following statistical model for Y_j as a function of X_j,

$$Y_j = \Gamma_r(X_j; C_r) + \varepsilon_j \quad \text{for} \quad j = 1, 2, \ldots, m_i \tag{9}$$

is postulated. If (9) is valid, then the transformation (5) yields approximately normal residuals with constant variance σ^2 so that

$$\{\varepsilon_j : j=1, \ldots, m_i\} \sim N(0, \sigma^2). \tag{10}$$

Notice that in (8)

$$\Gamma_r(0; \boldsymbol{C}_r) = 0 \quad \text{and} \quad \Gamma_r(1; \boldsymbol{C}_r) = \pi/2. \tag{11}$$

For fixed r (r = 2, 3, ...), the estimator $\tilde{\boldsymbol{C}}_r$ of the coefficient vector $\boldsymbol{C}_r$ is the solution of the following constrained least-squares regression problem:

$$\min_{\hat{\boldsymbol{C}}_r} \sum_{j=1}^{m_i} \left[Y_j - \Gamma_r\left(X_j; \hat{\boldsymbol{C}}_r\right) \right]^2 \tag{12}$$

subject to

$$\Gamma_r'\left(u; \hat{\boldsymbol{C}}_r\right) \neq 0 \quad \text{for all } u \in (0,1). \tag{13}$$

Observe that

$$\Gamma_r'\left(u;\hat{C}_r\right)=\sum_{k=1}^{r-1}k\hat{C}_k u^{k-1}+r\left(\frac{\pi}{2}-\sum_{k=1}^{r-1}\hat{C}_k\right)u^{r-1}, \tag{14}$$

so that (13) can be interpreted as requiring the zeros of the degree-$(r-1)$ polynomial (14) to lie outside the unit interval.

Associated with (12) is the error sum of squares

$$\mathrm{SSE}_r=\sum_{j=1}^{m_i}\left[Y_j-\Gamma_r\left(X_j;\tilde{C}_r\right)\right]^2 \tag{15}$$

for $r = 1, 2, \ldots$, where $\tilde{C}_1 \equiv (\pi/2)$ is taken in conformance with (7). Now for fixed r ($r \geq 2$), $\tilde{C}_r$ is the maximum likelihood estimator of C_r when (9)–(10) hold and the residuals are independent; moreover, the observation is

$$\tilde{\sigma}_r^2 \equiv \mathrm{SSE}_r/m_i \tag{16}$$

is the maximum likelihood estimator of σ^2 (Arnold 1981).

Given $Y \equiv [Y_1, \ldots, Y_{m_i}]^{\mathrm{T}} = y \equiv [y_1, \ldots, y_{m_i}]^{\mathrm{T}}$, it is seen that the associated likelihood

function is given by

$$L_r(\tilde{C}_r; y) = \prod_{j=1}^{m_i} \frac{1}{\tilde{\sigma}_r \sqrt{2\pi}} \exp\left\{ -\frac{1}{2}\left[\frac{Y_j - \Gamma_r(X_j; \tilde{C}_r)}{\tilde{\sigma}_r} \right]^2 \right\} \tag{17}$$

$$= (2\pi)^{-m_i/2} \left(\tilde{\sigma}_r^2\right)^{-m_i/2} \exp\left[-\frac{1}{2}\left(\frac{\mathrm{SSE}_r}{\tilde{\sigma}_r^2} \right) \right]$$

$$= \left(2\pi e\, \tilde{\sigma}_r^2\right)^{-m_i/2} \tag{18}$$

so that the resulting log-likelihood function for the fixed degree r is

$$\mathcal{L}_r(\tilde{C}_r; y) = -\frac{m_i}{2}\left[\ln(2\pi) + 1 + \ln\left(\tilde{\sigma}_r^2\right)\right]. \tag{19}$$

The degree r of the polynomial (8) is determined by a likelihood ratio test that has been adapted to constrained nonlinear regression. This approach to determining r is based on a similar technique used by Avramidis and Wilson (1994) in the context of another constrained nonlinear regression problem. At the outset, it is assumed that (8)–(10) hold for some value of r to be determined. Starting with the degree $r = 2$ and computing the optimal solution $\tilde{C}_r$ to (12), the null hypothesis is tested as

$$\sum_{k=1}^{r-1} C_k = \frac{\pi}{2} \tag{20}$$

(so that the degree of the polynomial (8) is at most $r-1$) versus the alternative hypothesis that

$$\sum_{k=1}^{r-1} C_k \neq \frac{\pi}{2} \tag{21}$$

(so that the degree of the polynomial (8) is at least r). If (20) holds, then Theorem 4.4.4 of Serfling (1980) ensures that

$$2\left[\mathcal{L}_r\left(\tilde{C}_r;\mathbf{y}\right)-\mathcal{L}_{r-1}\left(\tilde{C}_{r-1};\mathbf{y}\right)\right] \xrightarrow[m_i\to\infty]{D} \chi^2(1), \tag{22}$$

where $\chi^2(1)$ is a chi-square random variable with 1 degree of freedom. In view of (19), it can be seen that

$$2\left[\mathcal{L}_r\left(\tilde{C}_r;\mathbf{y}\right)-\mathcal{L}_{r-1}\left(\tilde{C}_{r-1};\mathbf{y}\right)\right] = -m_i \ln\left(\tilde{\sigma}_r^2/\tilde{\sigma}_{r-1}^2\right); \tag{23}$$

and in the formal algorithmic statement of the likelihood ratio procedure for estimating r given below,

$$\mathrm{SSE}_1 \equiv \sum_{j=1}^{m_i}\left[Y_j - \frac{\pi j}{2(m_i+1)}\right]^2 \tag{24}$$

and

$$\tilde{\sigma}_1^2 \equiv \frac{\mathrm{SSE}_1}{m_i}. \tag{25}$$

The likelihood ratio test is performed at the level of significance α, where $0 < \alpha < 1$, and the final estimate of r is

$$\tilde{r} = \min\left\{ r : r \geq 2 \text{ and } -m_i \ln\left(\frac{\tilde{\sigma}_r^2}{\tilde{\sigma}_{r-1}^2}\right) \leq \chi_{1-\alpha}^2(1) \right\} - 1, \tag{26}$$

where $\chi_{1-\alpha}^2(1)$ is the $1-\alpha$ quantile of the chi-square distribution with 1 degree of freedom. After determining the degree $\tilde{r}$ of the polynomial (8) used to fit the arrival data that was transformed via (5), the same degree $\tilde{r}$ for the polynomial (3) used to fit the original (untransformed) arrival data defined in (4) is selected.

Kuhl and Wilson (2000) formulated and evaluated an ordinary least squares (OLS) procedure for estimating the parametric mean value function of a NHPP and demonstrated that the estimation accuracy and computational efficiency of the OLS procedure in this type of application is reasonable. The same OLS procedure is applied here.

From (26), it can be observed that if the shape passing through Y_j (j: 1,..., m_i) points is symmetric, degree two might not improve the fitting significantly as compared to degree one letting the likelihood ratio to be successful and stop at degree one. This

results in underfitting of the data points. To avoid such type of underfitting, LRT is not considered for degree one. For all resolutions, degree one is fitted using R^2 test (Walpole, et. al 1998, pg . 377), where R^2 signifies the proportion of variation in W explained by the straight line regression. If an experiment has m_i experimental data points in the usual form (X_j , W_j) and that the regression line is estimated, then the total corrected sum of squares of W is given by

$$SST = SSR + SSE$$

An alternative and more informative formulation is

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

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

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

Value of R^2 is selected as greater than 99 to accept the degree one for fitting. The likelihood ratio test algorithm is summarized in Figure 4.1.

Likelihood Ratio Test Algorithm

[1] Perform R-Square test to check degree one polynomial.

$[R^2 = (SSR/SST)*100] > 99$ then $\tilde{r} \leftarrow 1$ and go to the next resolution.

[2] Take the composite (5) of the arc-sine and square-root transformations of the original arrival data (4) within each cycle.

[3] Set $r \leftarrow 1$ and check the maximum value of $\tilde{r}$

If $m_i \leq 2$, then deliver $\tilde{r} \leftarrow 1$ and stop; else set $r \leftarrow 2$ and go to **[4]**.

[4] Compute OLS estimator $\tilde{C}_r$.

a) Use starting value $\hat{C}_r = (\tilde{C}_{r-1}, 0)$ to obtain $\tilde{C}_r$ (e.g., to compute $\tilde{C}_2$, start with $\hat{C}_2 = (\pi/2, 0)$).

b) Determine the estimator $\tilde{C}_r$ that minimizes SSE_r as given by (15) subject to (13).

[5] Calculate $\tilde{\sigma}^2_{r-1} = \mathrm{SSE}_{r-1}/m_i$ and $\tilde{\sigma}^2_r = \mathrm{SSE}_r/m_i$.

[6] If $-m_i \ln\left(\frac{\tilde{\sigma}^2_r}{\tilde{\sigma}^2_{r-1}}\right) \leq \chi^2_{1-\alpha}(1)$, then deliver $\tilde{r} \leftarrow r-1$ and stop; otherwise, set $r \leftarrow r+1$ and go to **[4]**

Figure 4.1 Likelihood Ratio Test Algorithm

4.5 An Example Illustrating the Automated Multiresolution Procedure

To illustrate the automated multiresolution procedure for estimating NHPPs including the likelihood ratio test, the same example presented in Section 2.3 is considered. Recall that in the original example, the user is required to know the degree of the fitted function at each resolution. But here, the likelihood ratio test is used to determine the degree of the polynomial for the transformed data and then estimated the parameters of the functions $\{ R_i(\cdot) ; i=1,2,\ldots,p\}$ from the original data using a polynomial of the form (2) having the same degree. Since the NHPP in the example consists of two periodic components, the function $R_i(s)$ was estimated at: (a) resolution 0, corresponding to the long term trend over a period 28 days; (b) resolution 1, corresponding to the cyclic rate component with a period of 1 week; and (c) resolution 2, corresponding to the cyclic rate component with a period of 1 day. For this example, the input to the fitting program include the arrival data file with 216 arrival times and the α-level of 0.10 for the likelihood ratio test.

At resolution 0, the number of points involved in the fitting procedure is 4, excluding the origin. The 4 points are those corresponding to the cumulative fraction of arrivals observed at each end of week. The likelihood ratio test summarized in Figure 4.1 is applied. The fitted function at resolution 0 for the transformed data, $\Gamma_0(s)$ is a linear function (that is, a polynomial of degree $r = 1$). Consequently, a degree 1 polynomial is the fit to the original data. Figures 4.2 and 4.3 show the fitted functions at resolution 0.

Next, the resolution-1 function $R_1(s)$ associated with a period of 1 week is estimated. First the arrivals over each week were superimposed. The number of points is 7 corresponding to the cumulative fraction of arrivals observed at the end of each day.

These data are then transformed and the fitting procedure is applied resulting in a cubic function being fit to the data. The fitted function at resolution 1 for the transformed data, $\Gamma_1(s)$ is shown in Figure 4.4. A corresponding degree 3 polynomial is then fit to the original data. The estimated function at resolution 1 $\hat{R}_1(s)$ is shown in Figure 4.5.

The next step in the procedure is to estimate the resolution-2 function $R_2(s)$ associated with a period of 1 day. The 216 data points corresponding to the cumulative fraction of arrivals superimposed over each day. The fitted function at resolution 2 resulted in a polynomial function with degree 5. The function, $\Gamma_2(s)$ is shown in Figure 4.6. The corresponding estimated function for the original data at resolution 2, $\hat{R}_2(s)$ is shown in Figure.4.7.

The final estimations of all the functions for the transformed data are

Resolution 0:

$\Gamma_0(s) = 0.0560998693s \quad s \in (0,28]$

Resolution 1:

$\Gamma_1(s) = 0.552856266s - 0.102613039s^2 + 0.00795580633s^3 \quad s \in (0,7]$

Resolution 2:

$\Gamma_2(s) = 7.20057869s - 32.8600502s^2 + 78.1004944s^3 - 84.660675s^4 + 33.7904472s^5$

$s \in (0,1]$

The final estimations of all the functions for the original data are

Resolution 0:

$R_0(s) = 0.0357142857s \quad s \in (0,28]$

Resolution 1:

$R_1(s) = 0.264645542s - 0.0184986202s^2 + 0.0001571824s^3 \qquad s \in (0,7]$

Resolution 2:

$R_2(s) = 2.2876811s - 5.25465107s^2 + 11.5634518s^3 - 13.1874561s^4 + 5.59097481s^5$

$s \in (0,1]$

After fitting the required function at each resolution, the estimated mean-value function is constructed. The fitted mean value function is plotted against the observed cumulative number of arrivals over the entire time horizon (0,28] and is shown in Figure 4.8.

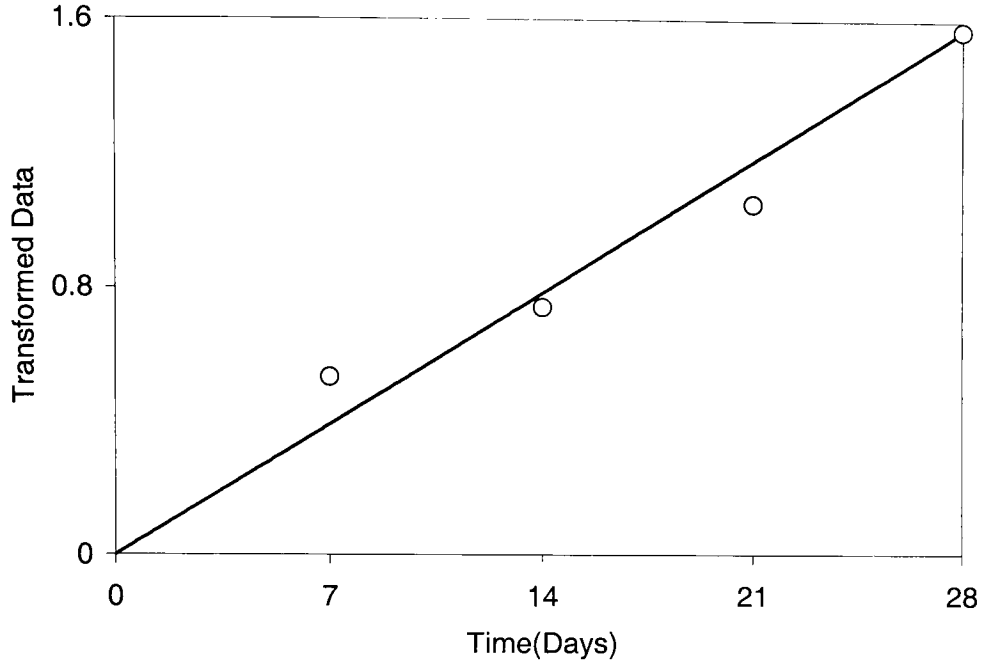

Figure 4.2 Resolution 0, Estimated Function for the Transformed Data

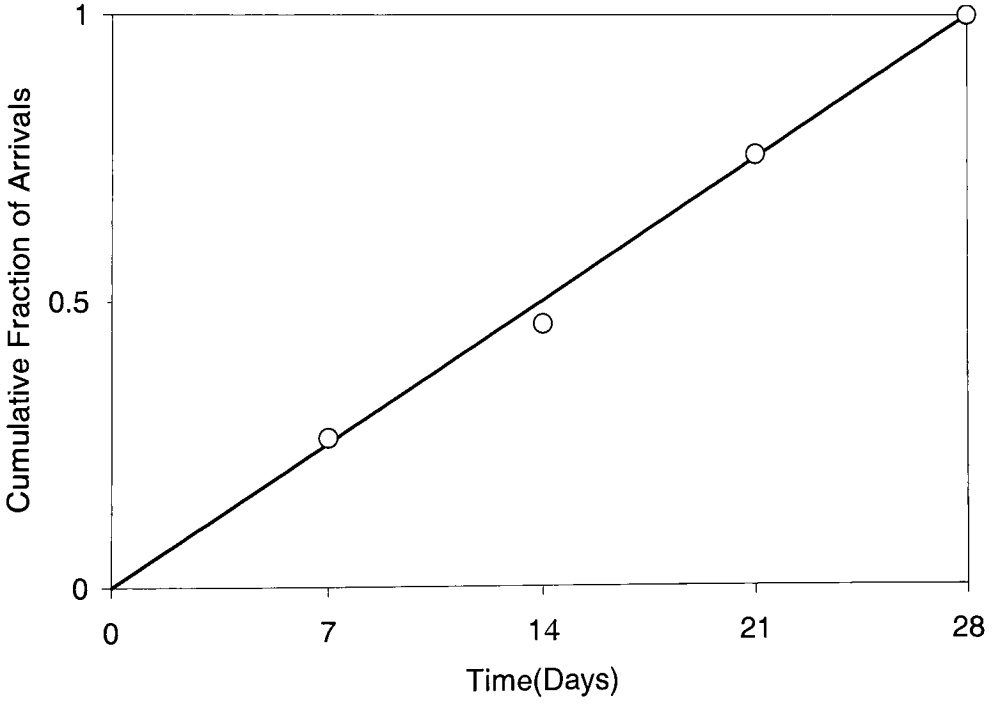

Figure 4.3 Resolution 0, Estimated Function for the Original Data

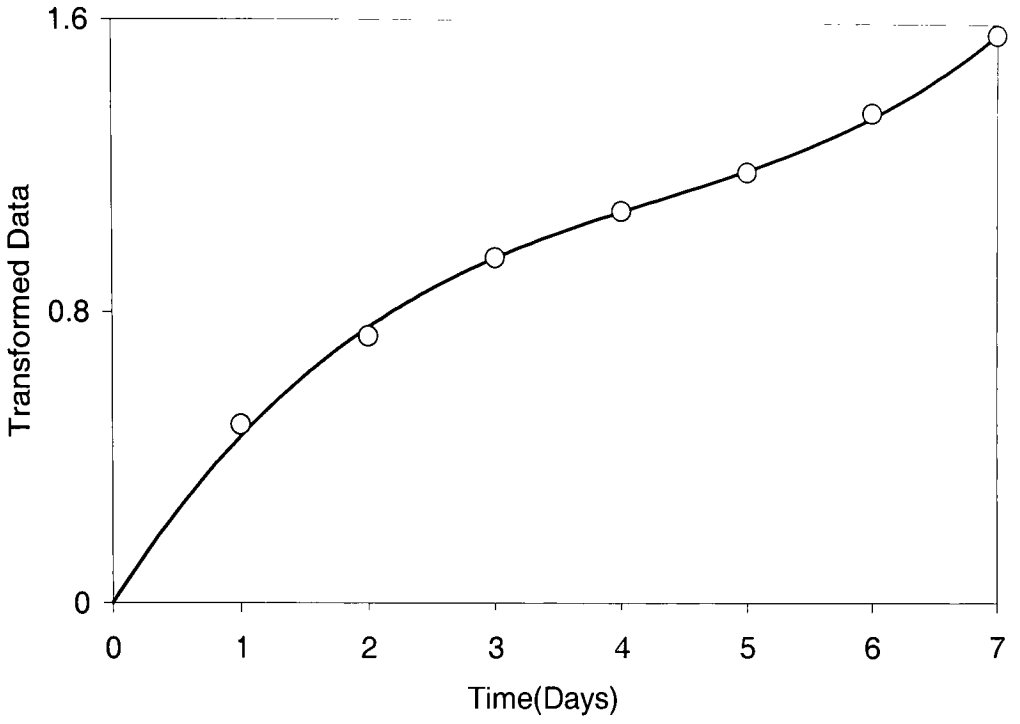

Fig. 4.4 Resolution 1, Estimated Function for the Transformed Data

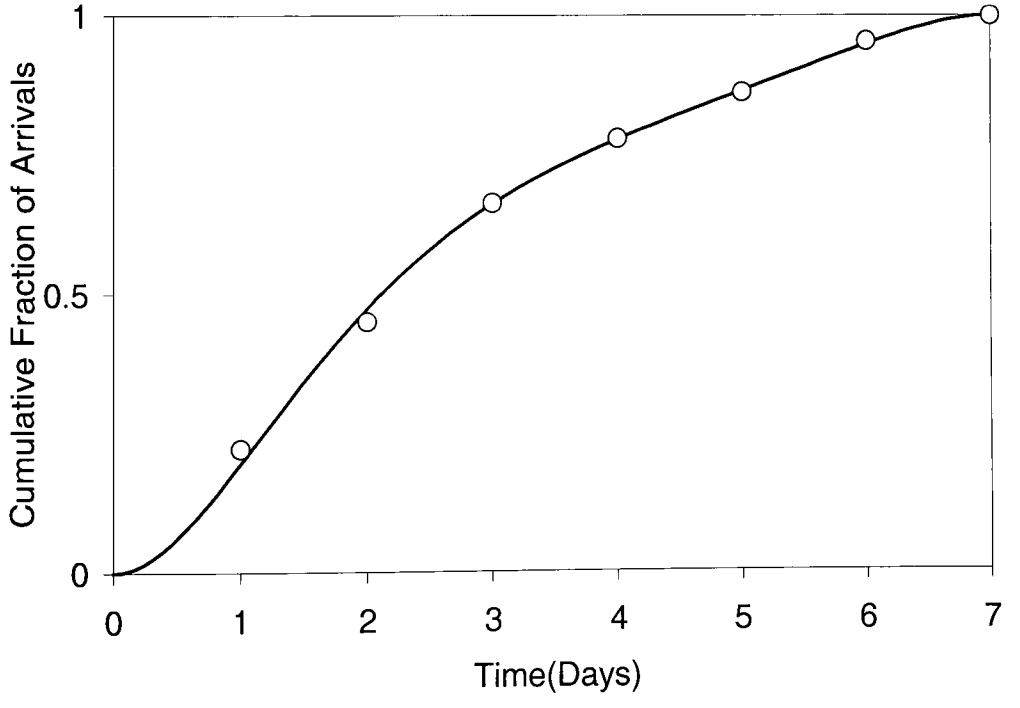

Figure 4.5 Resolution 1, Estimated Function for the Original Data

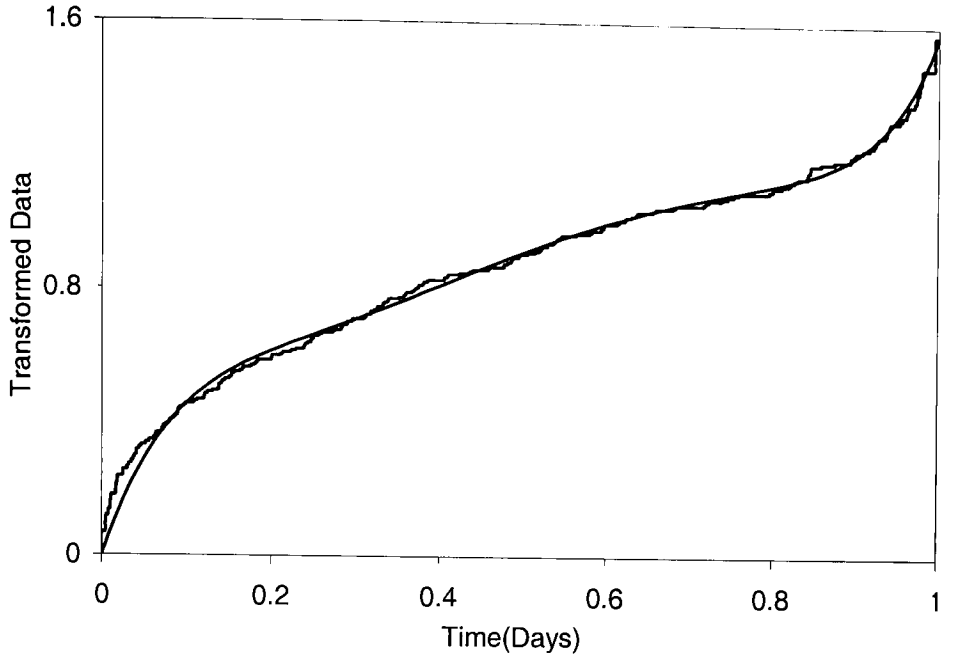

Figure 4.6 Resolution 2, Estimated Function for the Transformed Data

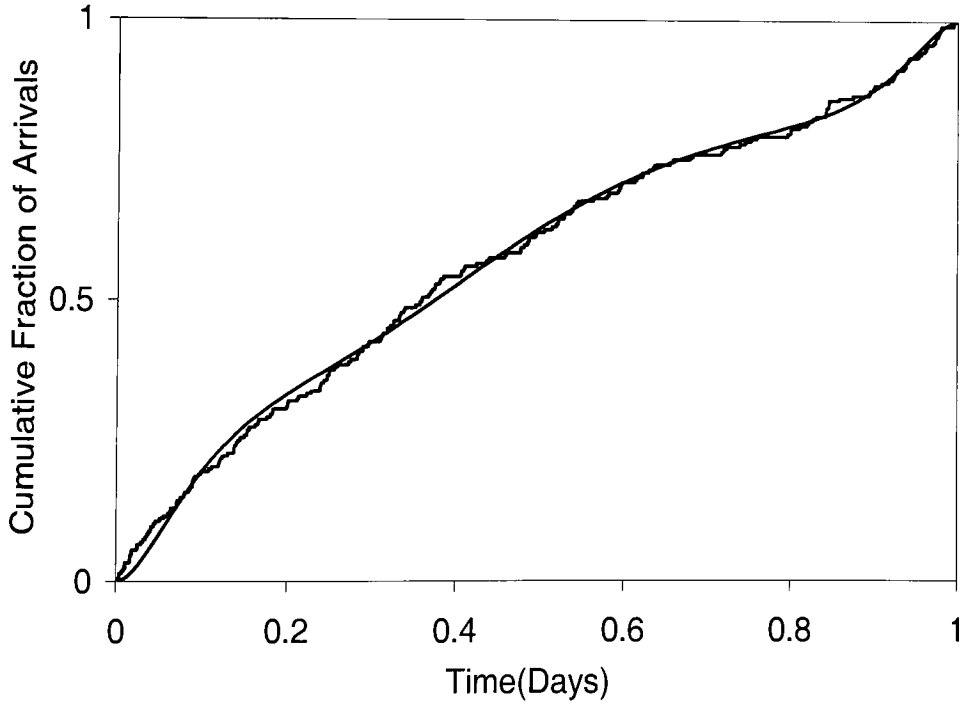

Figure 4.7 Resolution 2, Estimated Function for the Original Data

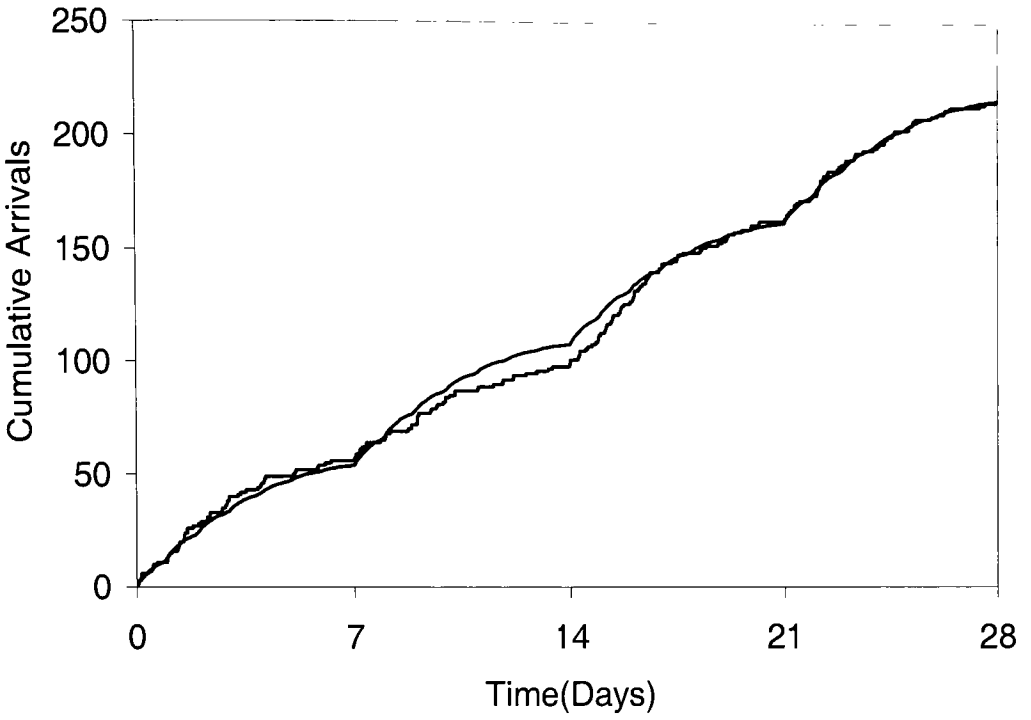

Figure 4.8 Estimated Mean Value Function (smooth curve) Superimposed on the Observed Cumulative Arrival Function (Step function)

This example demonstrates that the likelihood ratio test is capable of determining appropriate degrees of the polynomials fitted to the transformed data at each resolution, and using these fitted degrees to estimate the coefficients of the polynomials fit the original data at each resolution resulting in an accurate estimate of the mean-value function for the NHPP. In the next section, an experimental performance evaluation is conducted to evaluate the ability of the automated multiresolution procedure to accurately and consistently model NHPPs for use in simulation experiments.

5. EXPERIMENTAL PERFORMANCE EVALUATION

To evaluate the estimation procedure for applying the multiresolution procedure to an NHPP having long term trend or multiple cyclic effects, eight NHPPs that represent processes having up to two cyclic components or a general trend over time or both are selected. The first evaluation criterion is to compare the fitted degrees with the original degrees at all resolutions. Next, two types of performance measures are chosen to analyze the accuracy of the fitting procedure. The first type includes performance measures that estimate accuracy of the fitted mean value function with respect to the underlying (known) mean value function. The second type performance measures evaluate accuracy of the fitted mean value function to the observed data from the NHPP.

5.1 Generation of Experimental Data

The factors, which are considered while designing the experiments, are number of cyclic effects, lengths of the cyclic effects, degrees of polynomials for cyclic components, and degrees of polynomial for long-term trends. The lengths of cyclic effects are selected in the multiples of ten so that likelihood ratio test will have at least 10 points at each resolution to fit. The summary of the experiments is shown in Table 5.1.

Table 5.1 Experimental Set-up

Case	Long Term Trend Degree	First Cyclic Effect Degree	Second Cyclic Effect Degree
1	1	3	
2	1	5	
3	3	3	
4	5	5	
5	1	3	3
6	1	5	5
7	3	3	3
8	5	5	5

The degrees of polynomial are selected as 1, 3 and 5 for long-term trends, and 3 and 5 for the periodic components. The polynomials are created using LAB Fit Curve Fitting Software (Wilton Pereira da Silva, 2003). The polynomials for all the experiments are shown in Table 5.2. After selection of polynomials, the data is generated using the multiresolution procedure data generation algorithm (Kuhl and Wilson, 2001).

The experiment is conduced as follows. For each case, 100 realizations of the process are generated from the NHPP with the associated underlying (known) mean-value function. For each realization, the automated multiresolution procedure is used to fit a mean value function to the observed data and statistics are collected.

5.2 Formulation of Performance Measures

Johnson *et al.* (1994) formulate and use a set of numerical performance measures to evaluate EPT- type rate function. Kuhl *et al.* (1997) utilize these and derive an additional set of numerical performance measures to evaluate their maximum likelihood estimation procedure for fitting an EPTMP-type rate function. In this paper, to design the performance measures for the designed estimation procedure, both visual-subjective and numerical goodness-of-fit criteria are considered and are based on Kuhl *et al.* (1997). For completeness, all of these performance measures will be defined explicitly.

The goodness-of–fit statistics encompass absolute measures of error for each experiment as well as relative performance measures, which can be compared across different experiments. One additional performance measure is used to confirm efficiency of the likelihood ratio test. The degrees, determined by the likelihood ratio test, will be compared with the degrees of the polynomials that are used to generate the realizations.

Table 5.2 Polynomial Coefficients Used to Generate Data

Case	Res.	Deg.	Polynomial Degree Coefficients				
			1	2	3	4	5
1	0	1	0.1	-	-	-	-
	1	3	1.62999919	-2.69999442	2.06999523	-	-
2	0	1	0.1	-	-	-	-
	1	5	3.570480499892	-19.04968756733	50.16160310278	-55.05709976298	21.37470372765
3	0	3	0.2845238095238	-0.05615079365079	0.003769841269841	-	-
	1	3	1.62999919	-2.69999442	2.06999523	-	-
4	0	5	0.2810480499892	-0.1047879013144	0.02163275694893	-0.001818530489119	0.00005220857573799
	1	5	3.570480499892	-19.04968756733	50.16160310278	-55.05709976298	21.37470372765
5	0	1	0.01	-	-	-	-
	1	3	0.2845238095238	-0.05615079365079	0.003769841269841	-	-
	2	3	1.62999919	-2.69999442	2.06999523	-	-
6	0	1	0.01	-	-	-	-
	1	5	0.2810480499892	-0.1047879013144	0.02163275694893	-0.001818530489119	0.00005220857573799
	2	5	3.570480499892	-19.04968756733	50.16160310278	-55.05709976298	21.37470372765
7	0	3	0.0022619047619	0.0002559523809524	-0.00000178571428571	-	-
	1	3	0.2845238095238	-0.05615079365079	0.003769841269841	-	-
	2	3	1.62999919	-2.69999442	2.06999523	-	-
8	0	5	0.005667226292226	0.00007088293650793	0.000002290776353277	-7.189598595849E-08	4.62327024827E-10
	1	5	0.2810480499892	-0.1047879013144	0.02163275694893	-0.001818530489119	0.00005220857573799
	2	5	3.570480499892	-19.04968756733	50.16160310278	-55.05709976298	21.37470372765

The following goodness-of-fit statistics will be utilized to measure the performance of the estimation procedure in estimating the theoretical rate and mean-value function in each experimental case. Let $\tilde{\lambda}_k(t)$ and $\tilde{\mu}_k(t)$ denote the estimated rate and mean-value functions respectively for replication k of a given case ($k = 1,.....K$).

As defined in Kuhl *et al.* (1997), the average absolute error δ_k and the maximum absolute error δ_k^* in the estimation of the rate function $\lambda(t)$ on the k^{th} replication are

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

and

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

for $k = 1,\ldots,K$. The average absolute deviation Δ_k and the maximum absolute deviation Δ_k^* of the estimated mean-value function from $\mu(t)$ are

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

and

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

for $k = 1,\ldots,K$.

Johnson *et al.* (1994) developed aggregate performance measures based on the average and maximum absolute errors in the estimation of the rate function and the mean-value function. These measures are computed over all replications of a given experiment. Let $\bar{\delta}$ and V_δ represent the sample mean and the sample coefficient of variation of the observations $\{\delta_k: \quad k = 1,...,K\}$. Then the variation V_δ is given by

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

The maximum values $\bar{\delta}^*$ and V_{δ^*} are calculated similarly from the observations $\{\delta_k^*: \quad k = 1,...,K\}$. $\bar{\Delta}(\bar{\Delta}_k^*)$ and $V_\Delta(V_{\Delta^*})$, the analogous aggregate performance measures for the errors $\Delta_k(\Delta_k^*)$ incurred in estimating the mean-value function are also reported, by replacing $\bar{\delta}$ by Δ_k and V_δ by V_Δ. As stated in Kuhl *et al.* (1997), the normalized statistics computed to allow comparison of results for different cases

$$Q_\delta \equiv \frac{\bar{\delta}}{\mu(S)/S} \quad \text{and} \quad Q_{\delta^*} \equiv \frac{\bar{\delta}^*}{\mu(S)/S}$$

and

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

are also reported to permit comparison of results for different areas.

In addition to goodness-of-fit statistics, which measure the ability of the LRT algorithm to estimate the theoretical rate and mean-value functions of the underlying NHPP, Kuhl *et al.*(2000), formulated statistics that measure the ability of the LRT algorithm to approximate each observed arrival process. They let $\{t_{i,k} : i = 1,2,...,N_k(S)\}$ denote the arrival epochs observed in the time interval $[0,S]$ and on the kth replication of a given NHPP (k = 1,2,...K). Then for k = 1,2,...K, the kth replication of the sum of squared LRT algorithm estimation errors and the mean squared estimation error are

$$\mathrm{SS_E}(\tilde{\Theta})_k \equiv \sum_{i=1}^{N_k(S)} \left(\sqrt{\tilde{\mu}_k(t_{i,k})} - \sqrt{i - \frac{1}{4}} \right)^2$$

and

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

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

V_{MS_E} respectively represent the sample mean and the sample coefficient of variation of the observed values $\{MS_E(\tilde{\Theta})_k : k = 1,2,...,K\}$.

The average absolute error and maximum absolute error that occur in estimating the empirical mean-value function on the kth replication are estimated as follows:

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

and

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

for k = 1,2,...,K. Again, $\overline{D}$ and $\overline{D}^*$ denote the sample means of the observed values $\{D_k : k = 1,2,...,K\}$ and $\{D_k^* : k = 1,2,...,K\}$ respectively. Kuhl *et al.* (2000) also formulate two types of aggregate performance measures to compare $\overline{D}$ and $\overline{D}^*$ across experiments. 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 $\overline{D}$ and $\overline{D}^*$ so that

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

and

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

The second type of aggregate performance measure is estimated by expressing each performance measure $\overline{D}_k$ and $\overline{D}_k^*$ observed on the kth replication as a percentage of the average level of the empirical mean-value function on that replication. Then 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

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

respectively.

In addition to numerical performance measures, graphs will be plotted to provide a visual means of determining the quality of the estimates. The underlying theoretical rate and mean value function were graphed along with a tolerance band for the estimated rate and estimated mean-value function respectively.

Let

$$\tilde{\lambda}_{(1)}(t) < \tilde{\lambda}_{(2)}(t) < \cdots < \tilde{\lambda}_{(K)}(t)$$

for a fixed time $t \in (0, S]$, denote the ordered estimates of $\lambda(t)$ received on all K replications of the estimation procedure. Then an approximate $100(1-\beta)\%$ tolerance interval for $\lambda(t)$ is obtained as follows:

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

where $\lceil z \rceil$ denotes the smallest integer greater than or equal to z. Similarly, tolerance intervals are determined for the mean-value function $\mu(t)$ at a fixed time $t \in (0, S]$.

4.3 Presentation and Analysis of Results

Kuhl *et al.* (2000) provide some evidence that taking K = 100 replications of the OLS estimation procedure yield reasonably stable estimates of the goodness-of-fit statistics for all the cases they considered. Since LRT algorithm utilizes the same OLS procedure, take K = 100 replications.

The comparison of fitted degrees and original degrees is shown in Table 5.3. In all the cases, at resolution 0, LRT algorithm fits the same degree as the original ones for all the replications. Slight variations in the fitting are expected at the last resolutions because of the higher number of points; the LRT has to fit at those resolutions. As predicted, at the last resolutions, variations are observed in the fitting in all the cases except the first

one, where it fits 3 degree for all the replications. The maximum variation is observed in the second case, where the fitted degrees are 5 for 88 times, 6 for 10 times and 2 for 7 times. Overall, in all the cases, LRT algorithm never underestimated the fitting, which support the claim of high accuracy, consistency and reliability of LRT.

Tables 5.4 and 5.5 contain a summary of the goodness-of-fit statistics formulated in Section 5.2. Figures 5.1 through 5.20 display the graphs of 90% tolerance bands for the rate functions and mean value functions associated with all the cases.

Comparing the results in Table 5.4, for the cases, with no long term trend, i. e. cases 1, 2, 5 and 6 to the results for the respective cases with long term trends, i. e cases 3, 4, 7, 8, it is seen that in no long term trend cases the theoretical rate and mean-value functions are both estimated with greater accuracy. For example, in the Case 4 the average maximum absolute errors $\overline{\delta}^*$ and $\overline{\Delta}^*$ in estimating the rate and mean-value functions are 86.437 and 39.415, respectively; by contrast in Case 2 the corresponding statistics are 21.262 and 20.565, respectively. However, values of all variances including V_{δ^*}, V_{Δ} and V_{Δ^*}, are less for cases 3, 4, 7, 8 with long term trends as compared to cases 1, 2, 5, 6 with no long term trends.

For the cases 1, 3, 5, 7 with maximum polynomial degree three, the theoretical rate and mean-value functions are estimated more accurately as compared to the results of the cases 2, 4, 6, 8 respectively with maximum polynomial degree five. This is evident from the values of $\overline{\delta}^*$ and $\overline{\Delta}^*$ as 86.437 and 39.415 for Case 4, which are much greater than values 40.02 and 27.884 for Case 3.

The theoretical rate function is calculated more precisely with more number of resolutions as in cases 5, 6, 7, 8 with two cyclic components as compared to the cases 1,

2, 3, 4 with one cyclic component. For example, $\overline{\delta}^*$ is minimum of 12.758 and maximum of 86.43 for the cases with one cyclic component, while it is minimum of 4.167 and maximum of 20.41 for the cases with two cyclic components. The reason for this might be either number of cyclic components or length of the cyclic component or both. There is no effect of number of cyclic components on the calculation of theoretical mean value function.

Table 5.3 Comparison between Original and Fitted Degrees.

Case	Resolution	Original Degrees	Fitted Degree						
			1	2	3	4	5	6	7
1	0	1	100						
	1	3			100				
2	0	1	100						
	1	5					88	10	2
3	0	3			99	1			
	1	3			98	2			
4	0	5					100		
	1	5					96	3	1
5	0	1	100						
	1	3			99	1			
	2	3			95	2	2		1
6	0	1	100						
	1	5					100		
	2	5					98	2	
7	0	3			98	2			
	1	3			98	2			
	2	3			100				
8	0	5					100		
	1	5					99	1	
	2	5					99	1	

The results in Table 5.5 give some evidence that the LRT procedure generally yields a precise estimate of the target empirical mean-value function in each data set to

which the procedure is applied. For example, in Case 5, it can be observed that H_D = 0.000124 and H_{D^*} = 0.000369 so that on a "specific" replication of Case 5, the average and maximum absolute errors in estimating the empirical mean-value function are respectively 0.0124% and 0.0369% of the time-weighted average level of the empirical mean-value function computed over that replication of Case 5. Further, the residual mean square MS_E in Case 5 is 0.132069; and this provides another perspective on the accuracy with which the empirical mean-value function is estimated on each realization of Case 5. In general, the performance measures MS_E, Q_D, Q_{D^*}, H_{D^*} and H_D in Table 5.5 indicate that the LRT procedure consistently results in accurate fitting to the empirical mean-value function across all eight cases.

Table 5.4 Goodness of Fit Statistics for Estimating $\lambda(t)$ and $\mu(t)$,

C	$\bar{\delta}$	V_{δ}	Q_{δ}	$\bar{\delta}^{*}$	$V_{\delta^{*}}$	$Q_{\delta^{*}}$	$\bar{\Delta}$	V_{Δ}	Q_{Δ}	$\bar{\Delta}^{*}$	$V_{\Delta^{*}}$	$Q_{\Delta^{*}}$
1	4.63	0.45	0.05	12.76	0.54	0.13	13.47	0.71	0.03	27.36	0.70	0.06
2	5.77	0.34	0.06	21.26	0.49	0.21	10.09	0.88	0.02	20.57	0.84	0.04
3	5.89	0.31	0.06	40.02	0.47	0.40	14.88	0.59	0.03	27.88	0.54	0.06
4	13.00	0.09	0.13	86.44	0.25	0.86	18.40	0.40	0.03	39.42	0.30	0.07
5	0.58	0.34	0.06	4.17	0.51	0.42	13.50	0.71	0.03	27.39	0.70	0.06
6	1.29	0.09	0.13	8.69	0.33	0.87	13.76	0.70	0.03	28.29	0.66	0.06
7	0.68	0.29	0.07	7.11	0.40	0.71	16.15	0.58	0.03	30.48	0.56	0.06
8	1.40	0.09	0.14	20.42	0.29	2.04	15.22	0.55	0.04	35.00	0.46	0.08

Table 5.5 Goodness of Fit Statistics for Estimating $N(t)$

C	SS_E	V_{SS_E}	MS_E	V_{MS_E}	$\bar{D}$	$\bar{D}^{*}$	Q_D	$Q_{D^{*}}$	H_D	$H_{D^{*}}$
1	178.30	0.017	0.18	0.018	10.39	30.12	0.021	0.061	0.0001	0.0005
2	257.02	0.037	0.26	0.038	11.51	32.32	0.023	0.064	0.0003	0.0008
3	280.16	0.080	0.28	0.081	8.85	33.74	0.018	0.069	0.0002	0.0008
4	664.49	0.042	0.66	0.043	14.58	42.91	0.026	0.078	0.0003	0.0009
5	132.80	0.022	0.13	0.022	9.99	27.47	0.020	0.055	0.0001	0.0004
6	137.72	0.013	0.14	0.013	10.01	27.75	0.020	0.055	0.0001	0.0004
7	61.50	0.027	0.06	0.027	6.42	20.28	0.012	0.039	0.0001	0.0003
8	43.38	0.046	0.04	0.046	4.79	17.45	0.011	0.040	0.0001	0.0003

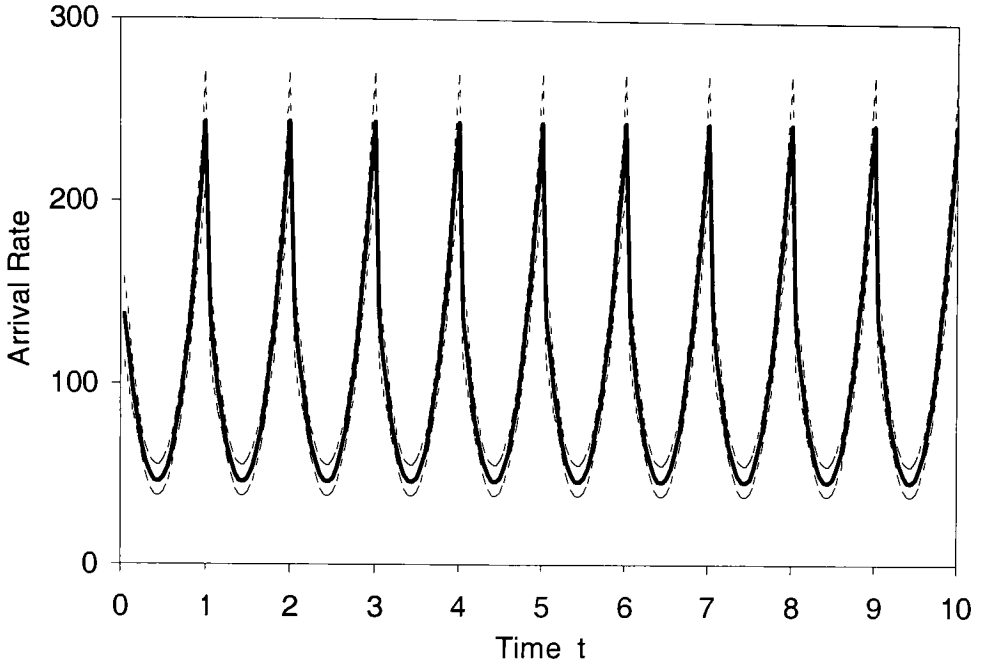

Figure 5.1 90% Tolerance Intervals for $\lambda(t), t \in [0,10]$, in Case 1.

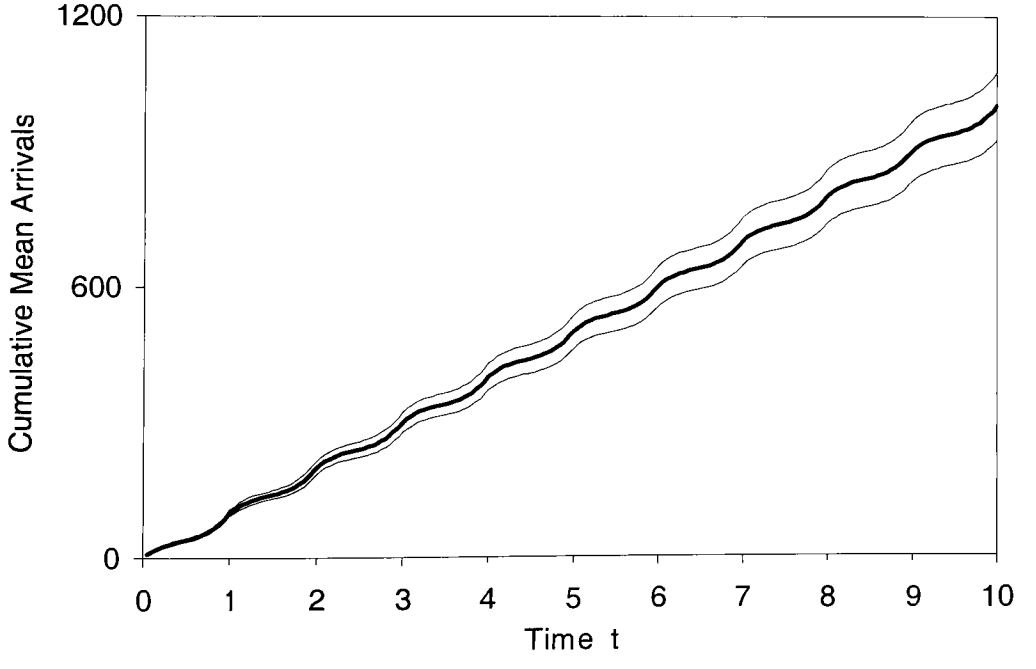

Figure 5.2 90% Tolerance Intervals for $\mu(t), t \in [0,10]$, in Case 1

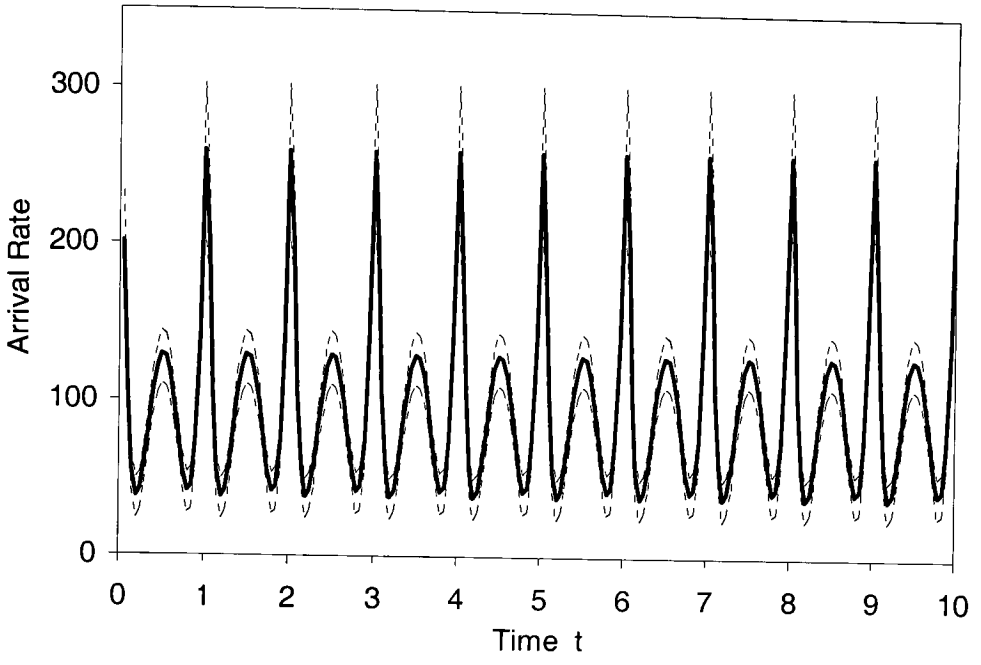

Figure 5.3 90% Tolerance Intervals for $\lambda(t), t \in [0,10]$, in Case 2.

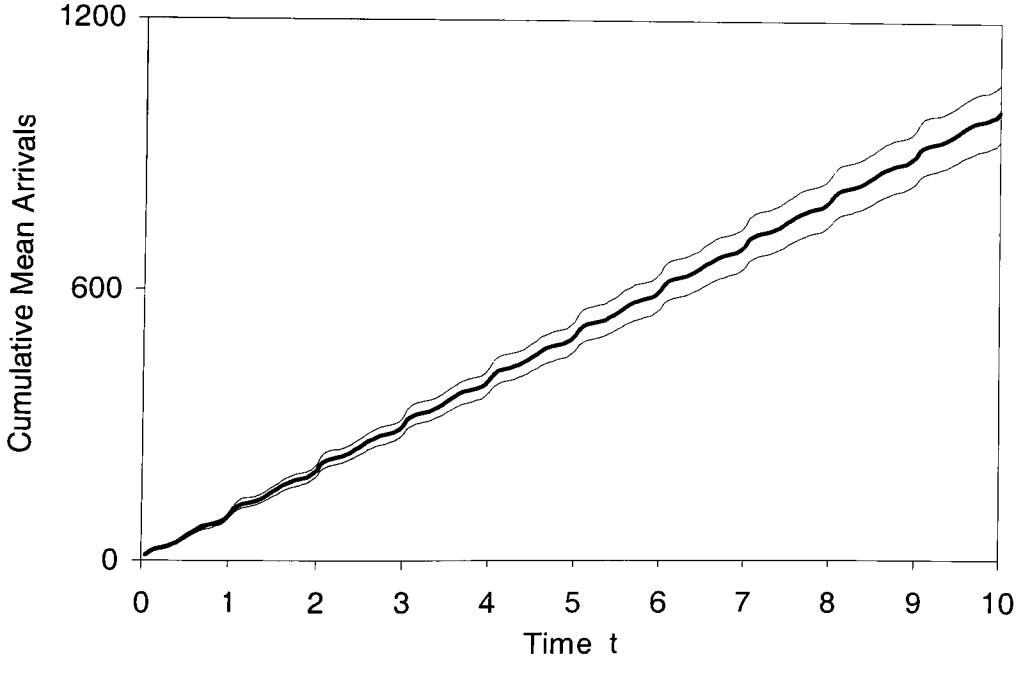

Figure 5.4 90% Tolerance Intervals for $\mu(t), t \in [0,10]$, in Case 2.

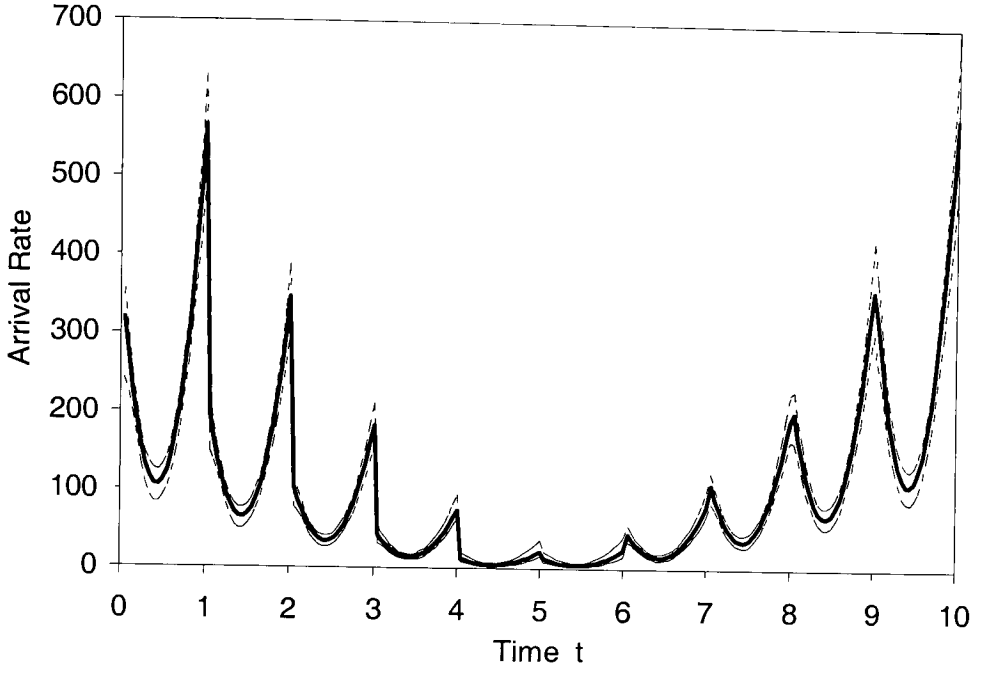

Figure 5.5 90% Tolerance Intervals for $\lambda(t), t \in [0,10]$, in Case 3.

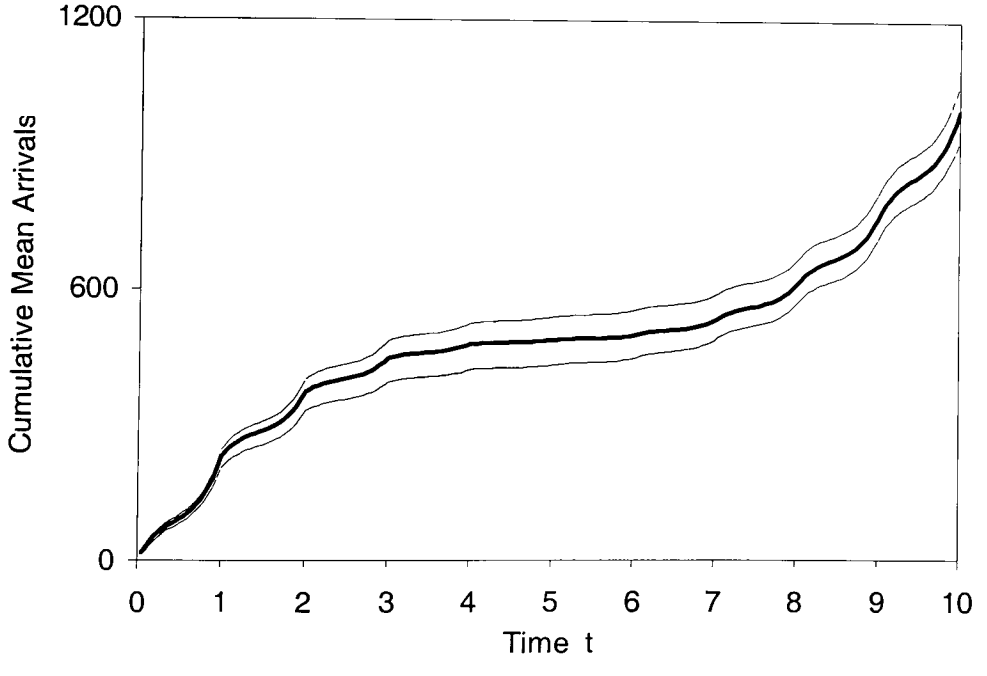

Figure 5.6 90% Tolerance Intervals for $\mu(t), t \in [0,10]$, in Case 3.

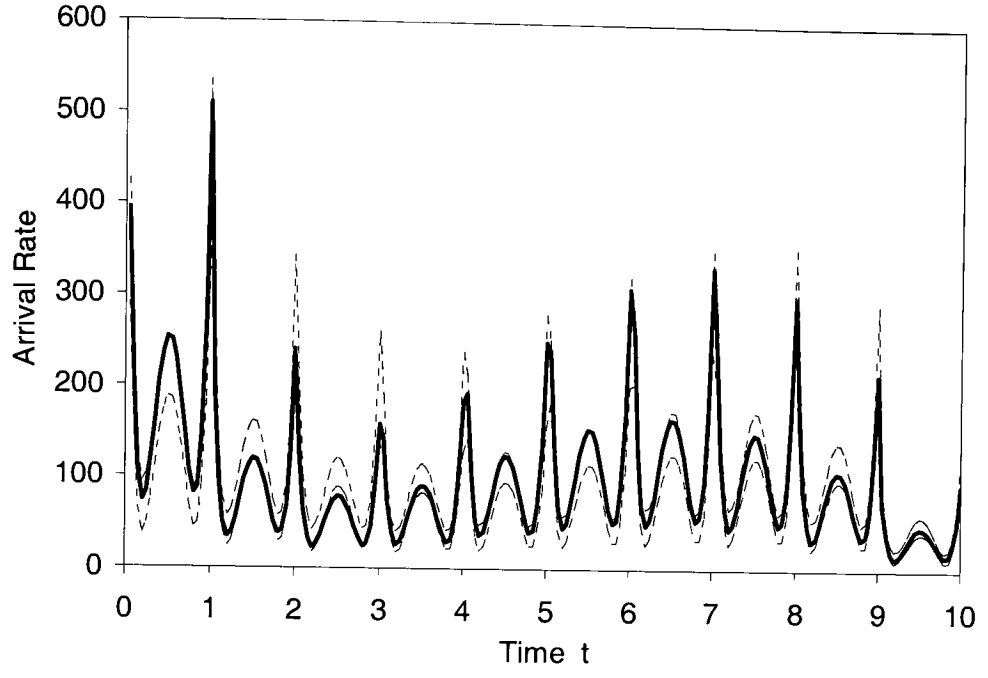

Figure 5.7 90% Tolerance Intervals for $\lambda(t), t \in [0,10]$, in Case 4.

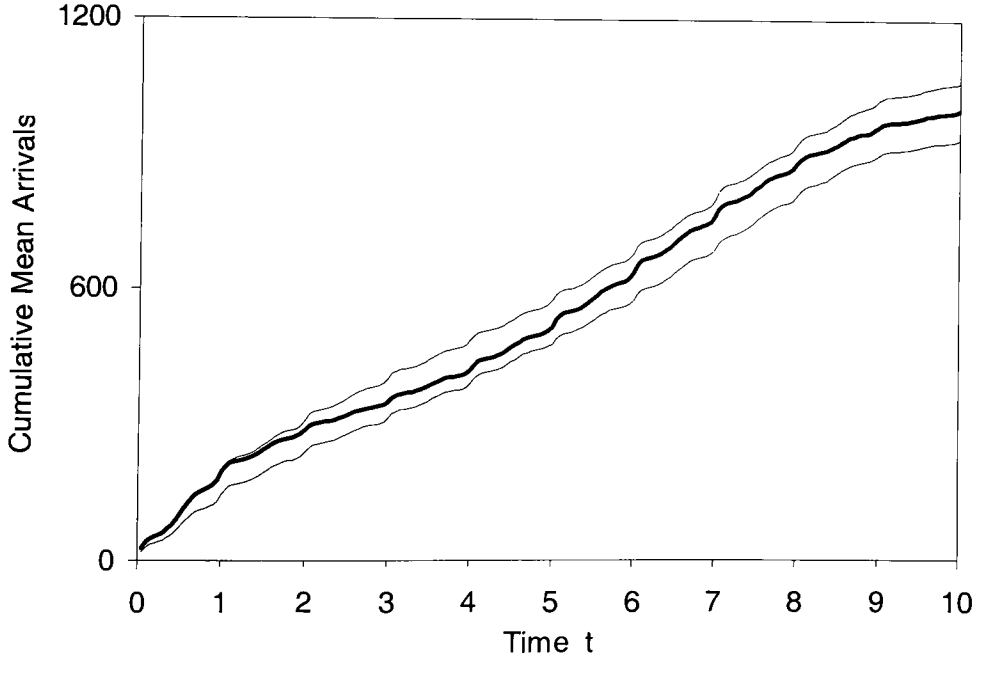

Figure 5.8 90% Tolerance Intervals for $\mu(t), t \in [0,10]$, in Case 4

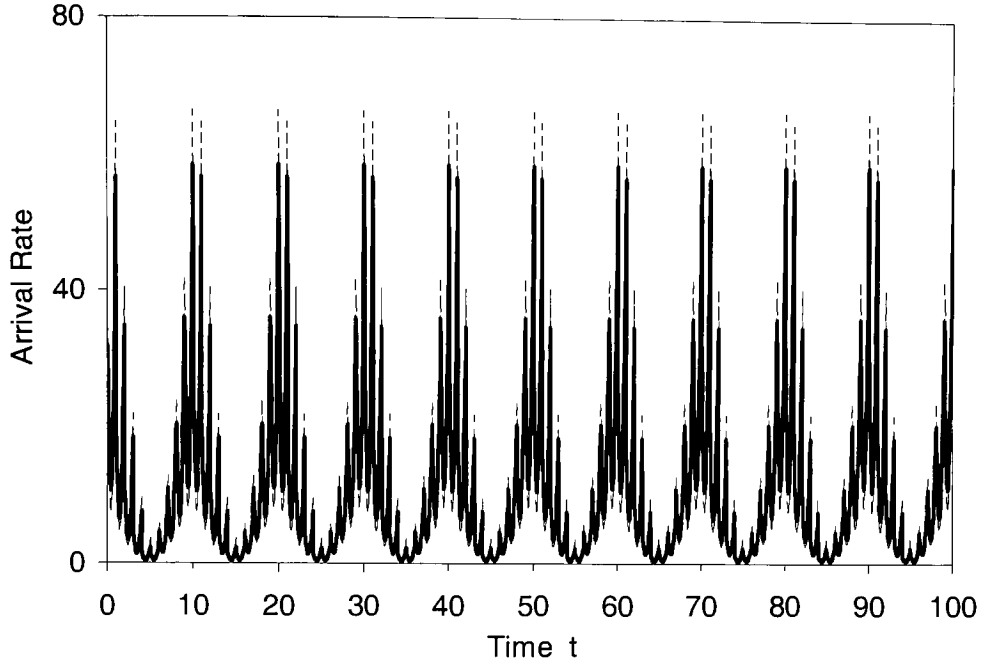

Figure 5.9 90% Tolerance Intervals for $\lambda(t), t \in [0,100]$, in Case 5.

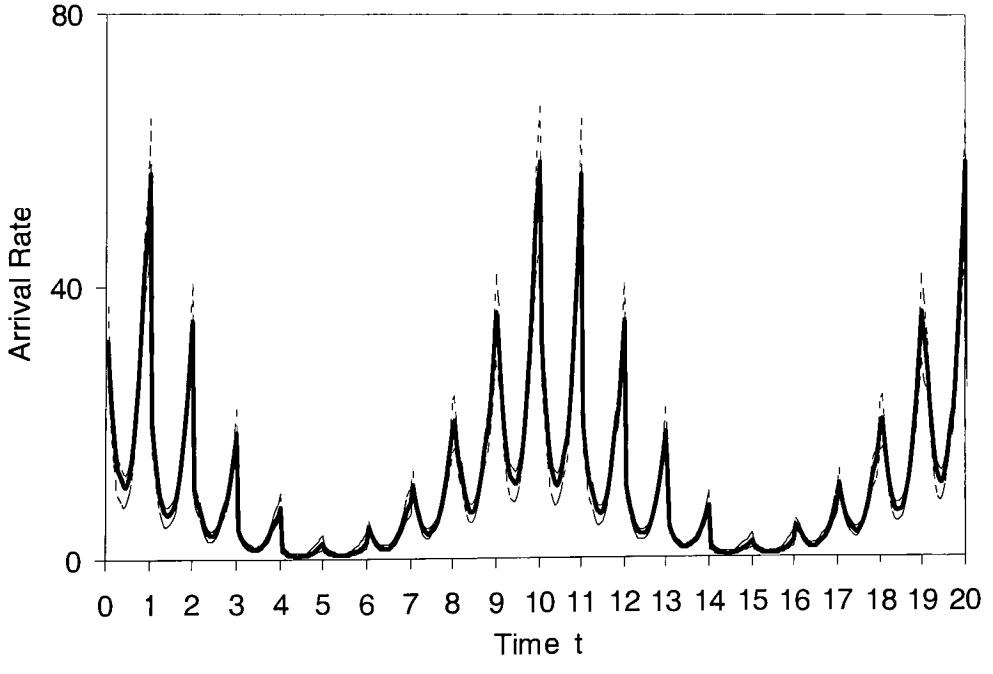

Figure 5.10 90% Tolerance Intervals for $\lambda(t), t \in [0,20]$, in Case 5

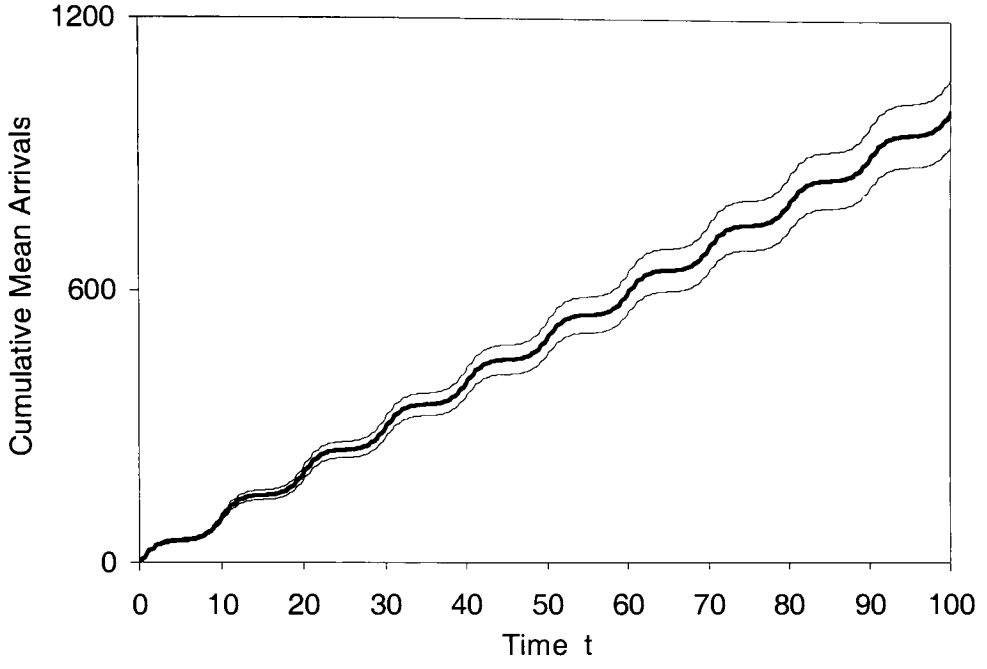

Figure 5.11 90% Tolerance Intervals for $\mu(t), t \in [0,100]$, in Case 5.

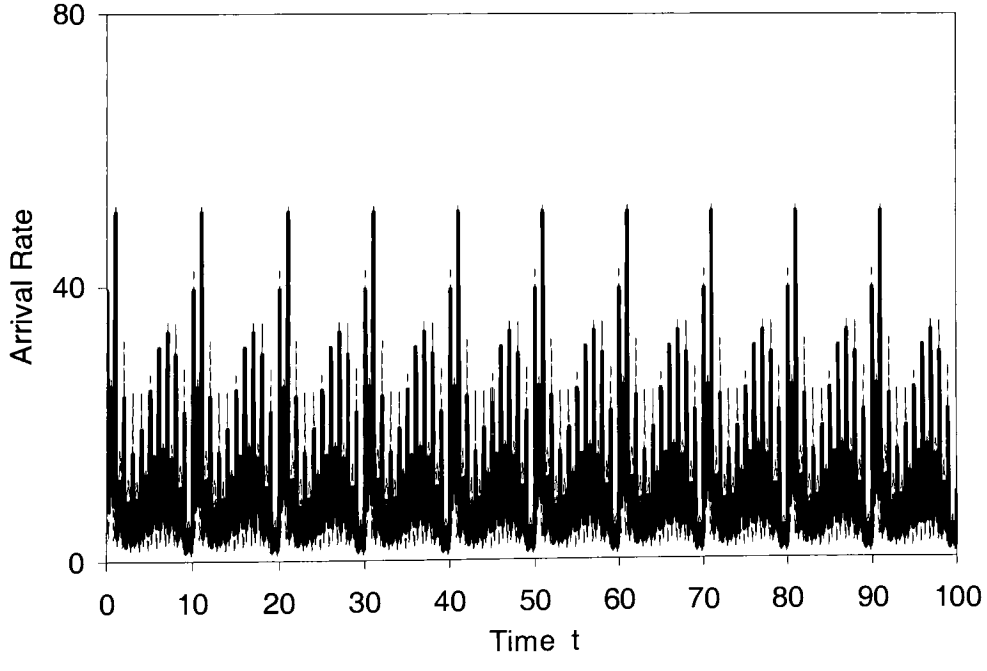

Figure 5.12 90% Tolerance Intervals for $\lambda(t), t \in [0,100]$, in Case 6.

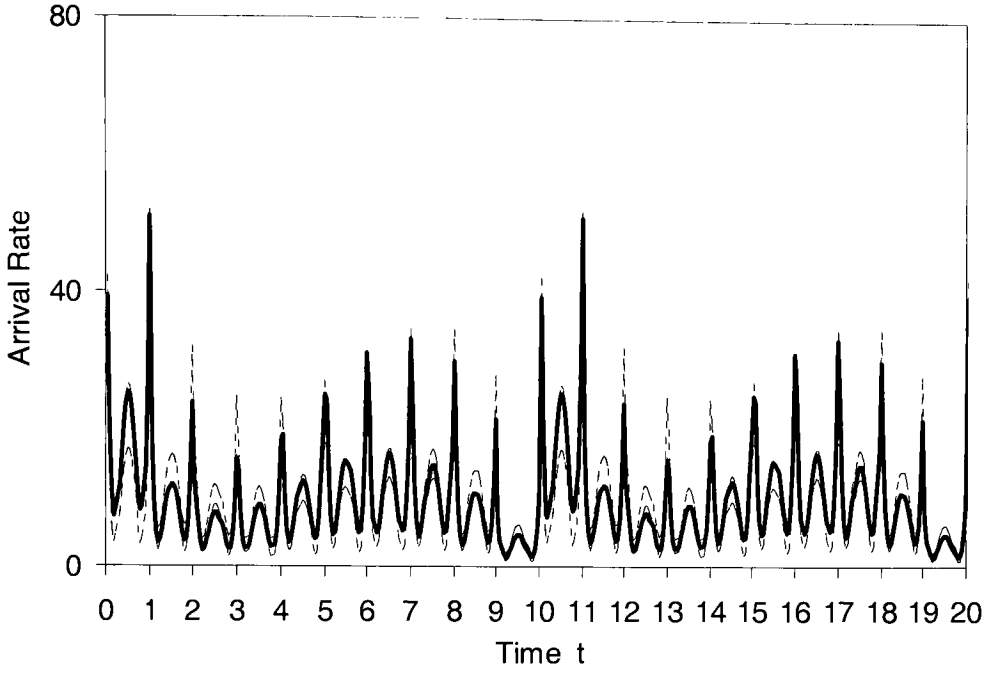

Figure 5.13 90% Tolerance Intervals for $\lambda(t), t \in [0,20]$, in Case 6.

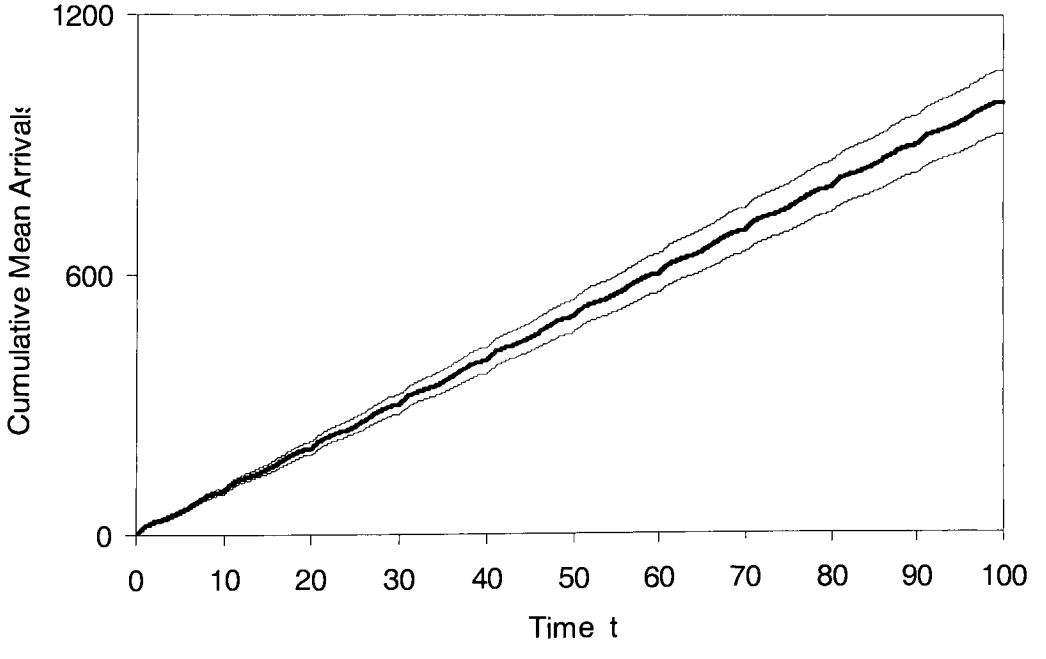

Figure 5.14 90% Tolerance Intervals for $\mu(t), t \in [0,100]$, in Case 6.

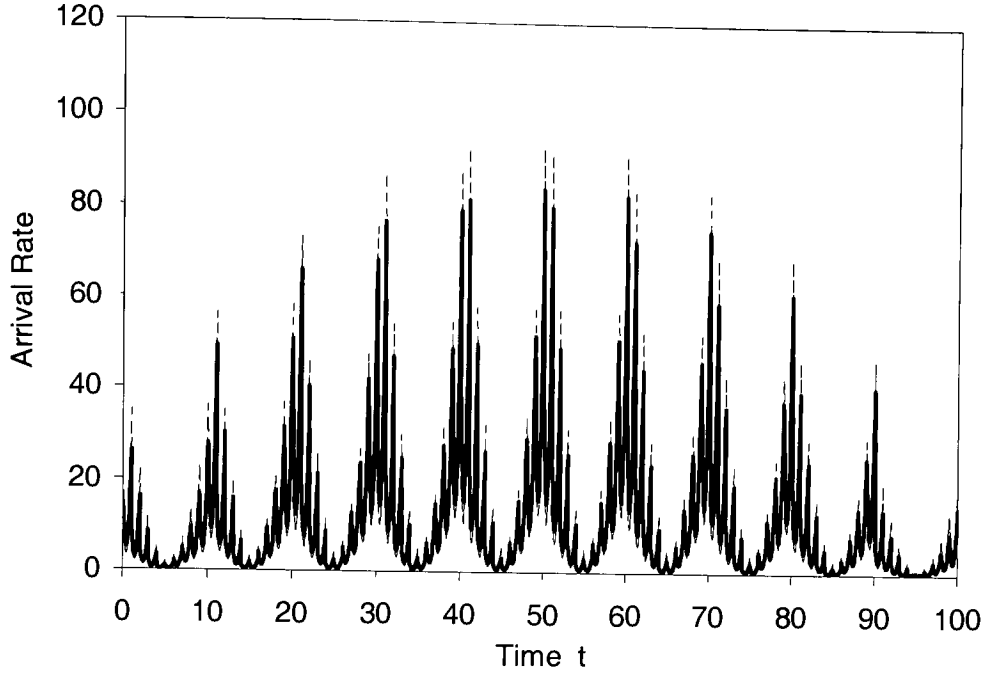

Figure 5.15 90% Tolerance Intervals for $\lambda(t), t \in [0,100]$, in Case7

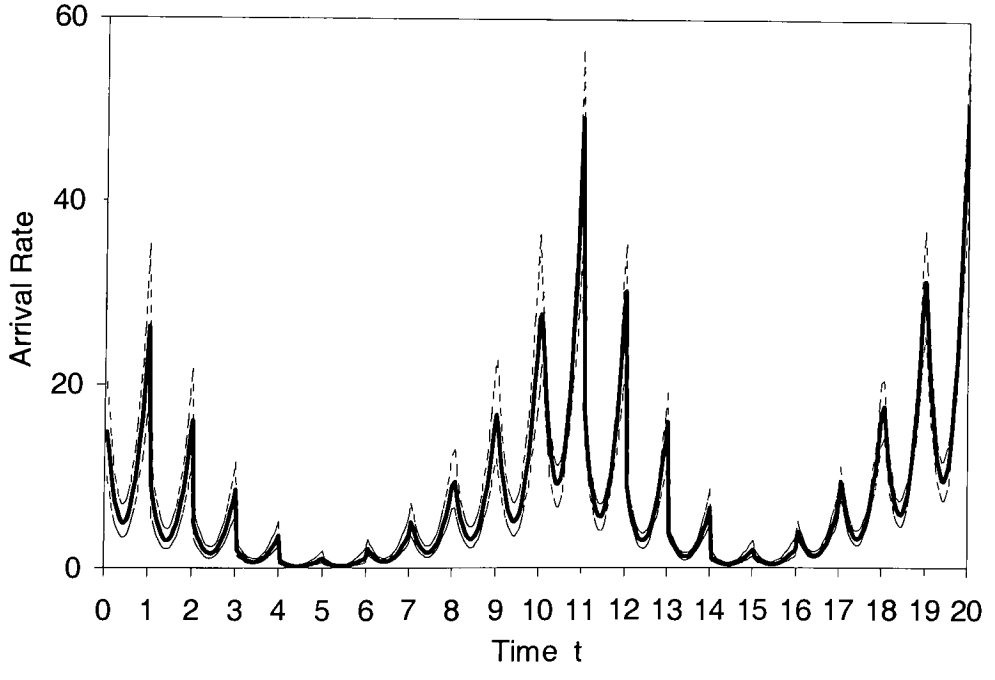

Figure 5.16 90% Tolerance Intervals for $\lambda(t), t \in [0,20]$, in Case 7

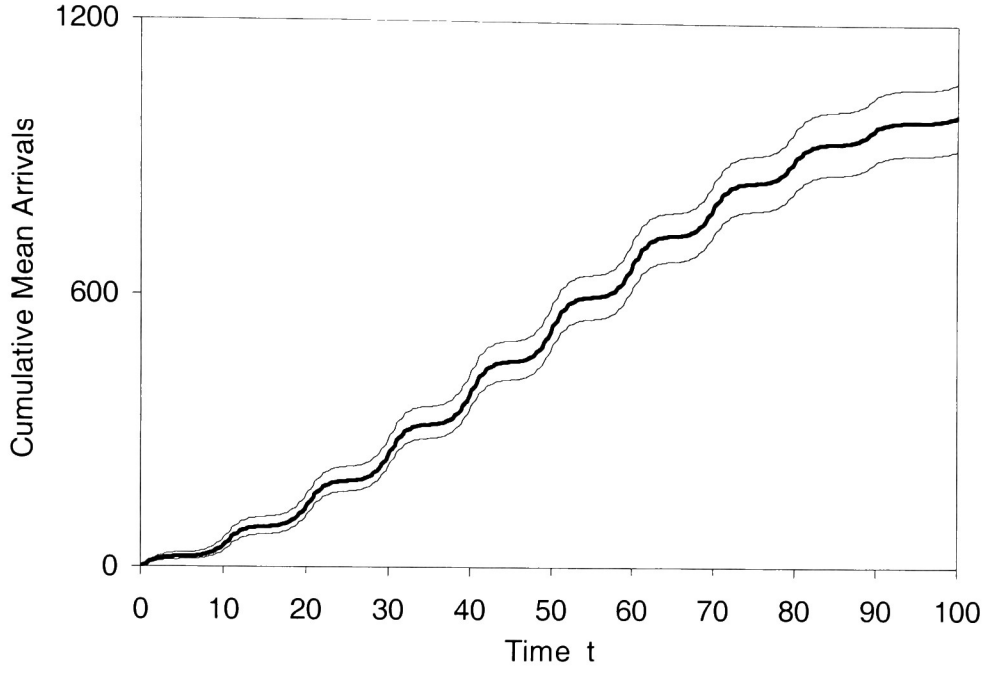

Figure 5.17 90% Tolerance Intervals for $\mu(t), t \in [0,100]$, in Case 7.

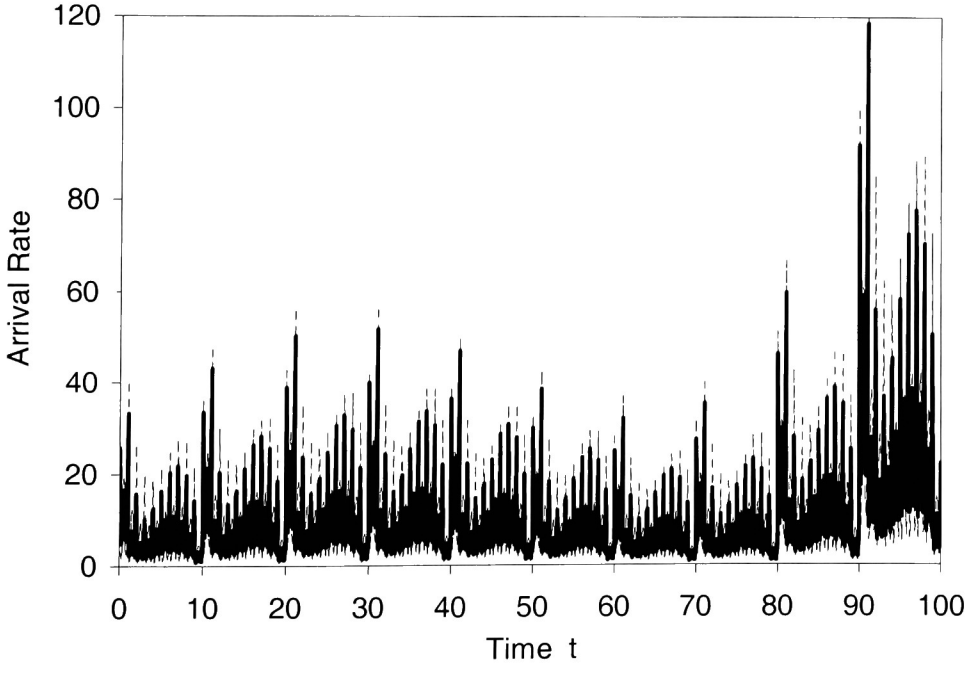

Figure 5.18 90% tolerance intervals for $\lambda(t), t \in [0,100]$, in Case 8.

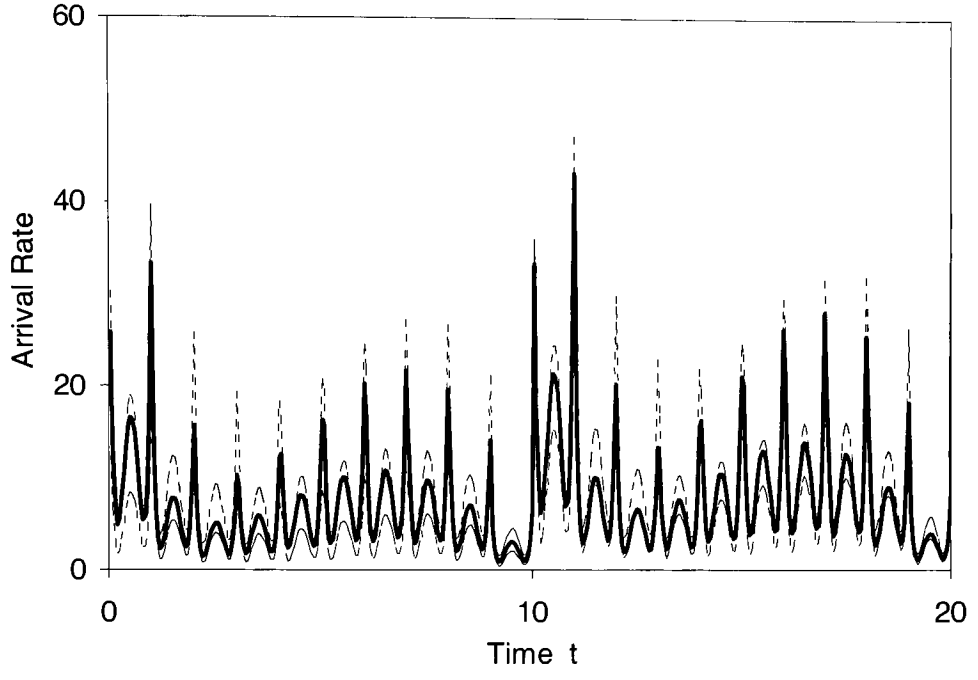

Figure 5.19 90% Tolerance Intervals for $\lambda(t), t \in [0,20]$, in Case 8.

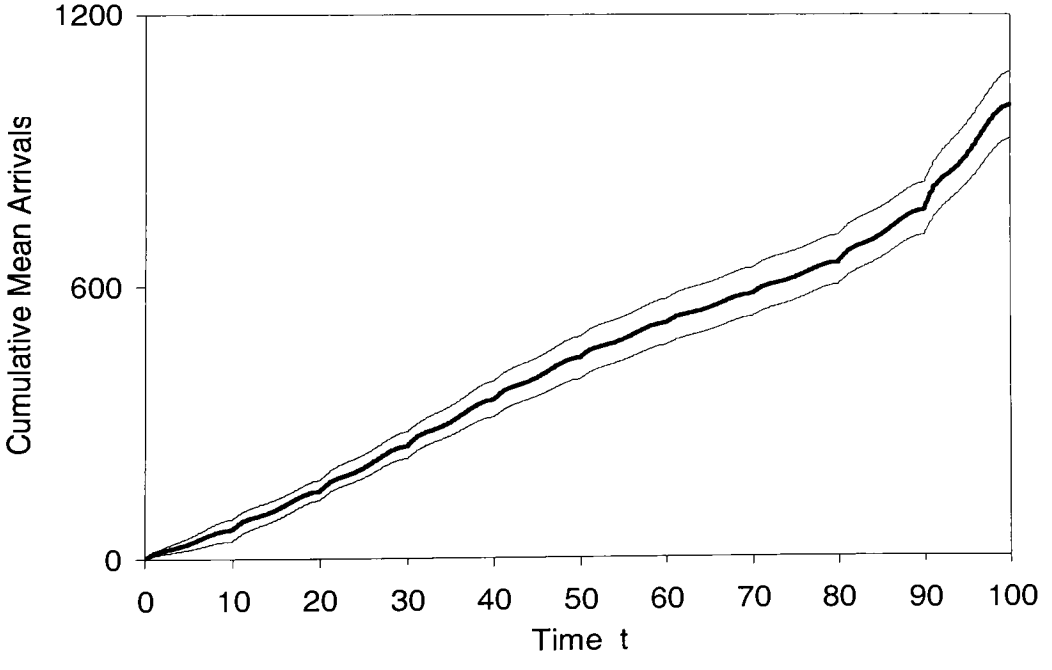

Figure 5.20 90% Tolerance Intervals for $\mu(t), t \in [0,100]$, in Case 8.

The plots of the 90% tolerance bands about the theoretical rate functions designate that the LRT procedure is consistently able to fit a reasonable rate function to the underlying NHPP. The plots of the tolerance bands for LRT procedure are widest at peaks and valleys of the arrival rate.

The plots of the 90% tolerance bands about the theoretical mean value functions also designate that the LRT procedure is consistently provide reasonable estimates for the underlying NHPP. In the plots, the widths of the tolerance bands increase over time. That is because the error is cumulative over time, the estimation error increases as the mean-value function is plotted over time.

The multiresolution procedure is automated in Chapter 4. The automated procedure is evaluated in this chapter. Now, to make this procedure practical and user friendly, a graphical user interface is needed to interact with the user and to provide results. Next objective of this thesis is to develop a web-based infrastructure for implementation of the multiresolution procedure on the internet.

6. WEB-BASED IMPLEMENTATION OF MULTIRESOLUTION PROCEDURE FOR MODELING NHPPS

In order to disseminate the automated multiresolution procedure so that simulation practitioners are able to readily make use of the techniques for modeling and simulating NHPPs, a web-based input modeling environment is developed. This web-based software is designed to allow the user to fit an NHPP to observed data and to generate realizations of the fitted NHPP that could be used in simulation experiments. Furthermore, the web-based implementation will allow the user to utilize the fitting procedures any where in the world without the need to download or install specialized input modeling software.

This chapter discusses design and development of the web-based multiresolution procedure. In summary, the web-based input modeling software involves the following sequence of activities. Upon accessing the input modeling web-page, the data file and the information about the parameters such as number of resolutions, maximum degree to be fitted at each resolution, length of each resolution, etc. are provided by the client on the Internet. The data file is passed to the server through the network. The server executes the necessary application on the server and conveys the results to the client by means of graphs and links to output files containing the fitted NHPP parameters. Once the NHPP has been fit to the data, the user has the option of generating realizations of the NHPP. The generated data are stored in files and provided to the user over the Internet.

6.1 Application Requirements

The requirements of a web-based multiresolution procedure are as follows:

1. It should have capabilities to display the graphs accurately.
2. User should be able to change the input data, which includes number of resolutions, maximum degrees to be fitted at each resolution, etc.
3. The output should include the details of fitted polynomials and mean value function with graphical demonstration.
4. Environment should let the user save the results in proper format.
5. User interactive environment should allow the user to change the parameters of the multiresolution procedure while generating data.
6. The data generation and data fitting should be accurate. This includes proper random number generator, exact representation of data, and precise data manipulation.

The technical issues that need to be addressed while creating this web-based application are as follows:

1. Security: Security is the most significant issue while creating web-based application. The data communication between the client and server should be secure.
2. Speed: This factor is more dependent on the network traffic at the particular time, CPU speed of the client m/c, Internet speed and other factors. But still while designing environment, it can be make sure that during favorable conditions, our application response time is optimal.
3. Platform Independence: This issue has been discussed and solved by the web designers by using different technologies such as Java and by testing their applications on different platforms, on different browsers etc.

4. The application should be capable of handling multiple users at a time.
5. The environment should be use-friendly. This includes proper documentation of the processes, online availability of references and online help.

6.2 Software Selection

In this section, concepts of web-based applications and current technologies available in the market are discussed and the appropriate ones are selected to fulfill the requirements discussed in the previous section. In order to develop the web-based multiresolution procedure, both static and dynamic web-pages should be considered. Static and dynamic web-pages have their own advantages and disadvantages. To receive maximum benefits, pros and cons of various technologies from both the areas are elaborated below.

6.2.1 Static Web Pages

A static web-page is a page whose content encompasses some HTML code that types directly into a text editor and saved as an .htm or .html file. Thus, the designer of the page has already completely determined the exact content of the page, in HTML, before any user visits the page. The content and appearance of a static web page is always same regardless of the user (who visits the page), the time (when they visit), the path (how they arrive at the page) or any other factor.

A static web page is served to the client in five steps namely the author writes HTML, the client requests web page, the web server locates .html file, the HTML stream (from .html page) is returned to the browser, and the browser processes HTML and displays page. In this process, the server or the browser is not doing much work as the content is completely determined before the page is even requested. The reason is, HTML offers no features for

personalizing the web pages. Further, the HTML code is not secure. Anybody can access and use the code. Though the static pages are fast, they are quite limited without any dynamic features.

6.2.2 Dynamic Web Pages

There are two different ways of creating dynamic web pages. They are client side dynamic web pages and server side dynamic web pages.

In the client side model, modules (or plug-ins) present in the browser perform all the work of creating dynamic pages. The HTML code is passed to the browser along with a set of instructions, which is referenced from the HTML code. The browser generates the page dynamically when it is requested. This model serves the web-page in six steps namely the author writes instructions, the clients requests web page, the web server locates HTML and instruction file, the HTML and instructions are returned to the browser, a module in the browser processes instructions and turns them into HTML, and finally the browser processes HTML and displays page. The client side models have not been preferred in recent times, as they take a long time to download. A second disadvantage is that each browser may interpret the instructions differently. Another major drawback is a difficulty in writing client-side code that uses server-side resources such as databases. Also all code for client side scripting is available to everybody, which can be undesirable. Each client side technology relies on a module built into the browser to process the instructions. The client side technologies are a mixture of scripting languages, controls, and fully fledged programming languages. Two scripting languages, JavaScript and VBScript, two controls, ActiveX control, Java Applet and one fully-fledged language, Curl are considered. The WBMP application requires

displaying various graphs in the output by performing calculations. To create an applet the best applet technology is selected as a client side tool.

In the server side model, web-server processes all the instructions to create a dynamic page. This model creates a dynamic page in six steps namely the author writes instructions, clients requests web page, the web server locates instruction file, the web server processes instructions to create HTML, the HTML stream is returned to browser, and finally the browser processes HTML and displays page. All the processing is done on the server before the page is sent to the browser. One key advantage this has over the client side model is that only HTML code describing the final page is transferred to the browser. This means that the logic is not visible to the user and it can be assumed that all the browsers will only display the HTML code. Each server side technology relies on a modular attachment added onto the web server rather than the browser. Server side code is processed on the server, and only HTML, and any client side script is sent back to the browser by the web server. Five server side technologies namely CGI, JSP, ColdFusion, PHP and ASP.NET are considered. ASP.NET is selected as it is the latest and most powerful server-side technology for creating dynamic web pages (Robinson et. al., 2001). To develop WBMP, Windows 2000 is utilized as an operating system and IIS 5.0 as a web server. Further, the original algorithm for the multiresolution procedure is written in Fortran, which is the most used language for statistical algorithms and fast processing. The server side process deals with this Fortran executable file, which is a WIN32 application. ASP.NET technology is very suitable for Windows Operating system and for handling WIN32 applications.

6.3 Application Design and Development

The folder ASPNET, which is the base folder for this application, includes six directories namely Operation, Clients, Fortranexe, Images, References and RunFortranExecutable. Operation directory has all the files such as HTML (.html), ASP.NET (.aspx) and JAVA Applet (.java, .class) related to the web-based processing. Clients folder is created to store data arrived from the client, to perform the processing and to save the results. The IP Address of a client machine is used as a unique name for that client folder. Fortranexe folder stores the Fortran executable files for estimation (FitData.exe) and data generation (GenerateData.exe). The data files to demonstrate an example are also present in the Fortarnexe folder. The Images folder encompasses all the images displayed in the web pages. The References folder has all the files and documents containing information about references, research papers and help. Finally, the RunFortranExecutable folder contains a VB.NET application, which is utilized by ASP.NET environment to run the Fortran executable files. The flowcharts for the web-based application for data generation and data fitting are shown in Figures 6.1 and 6.2. This section is divided into three parts namely input data, estimate data and output results. The input data section concentrates on creation of input files for the Fortran program and other issues of web based data management. The estimate data section elaborates on executing the Fortran executable and obtaining results. Finally, the output results section discusses about reading the data from the output files generated from the Fortran program and sending the results to the applet.

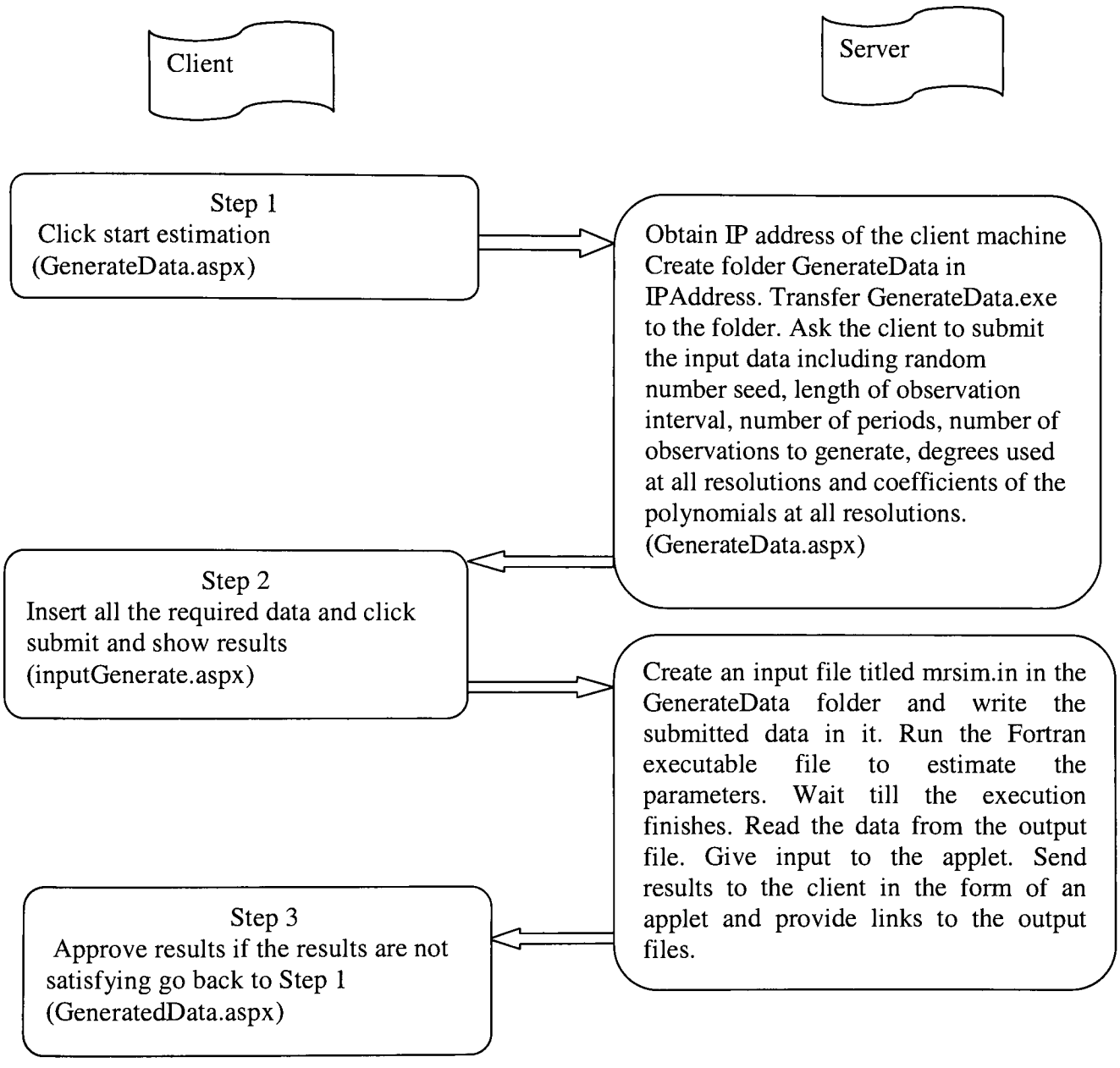

Figure 6.1 Flowchart for Web-based Data Generation

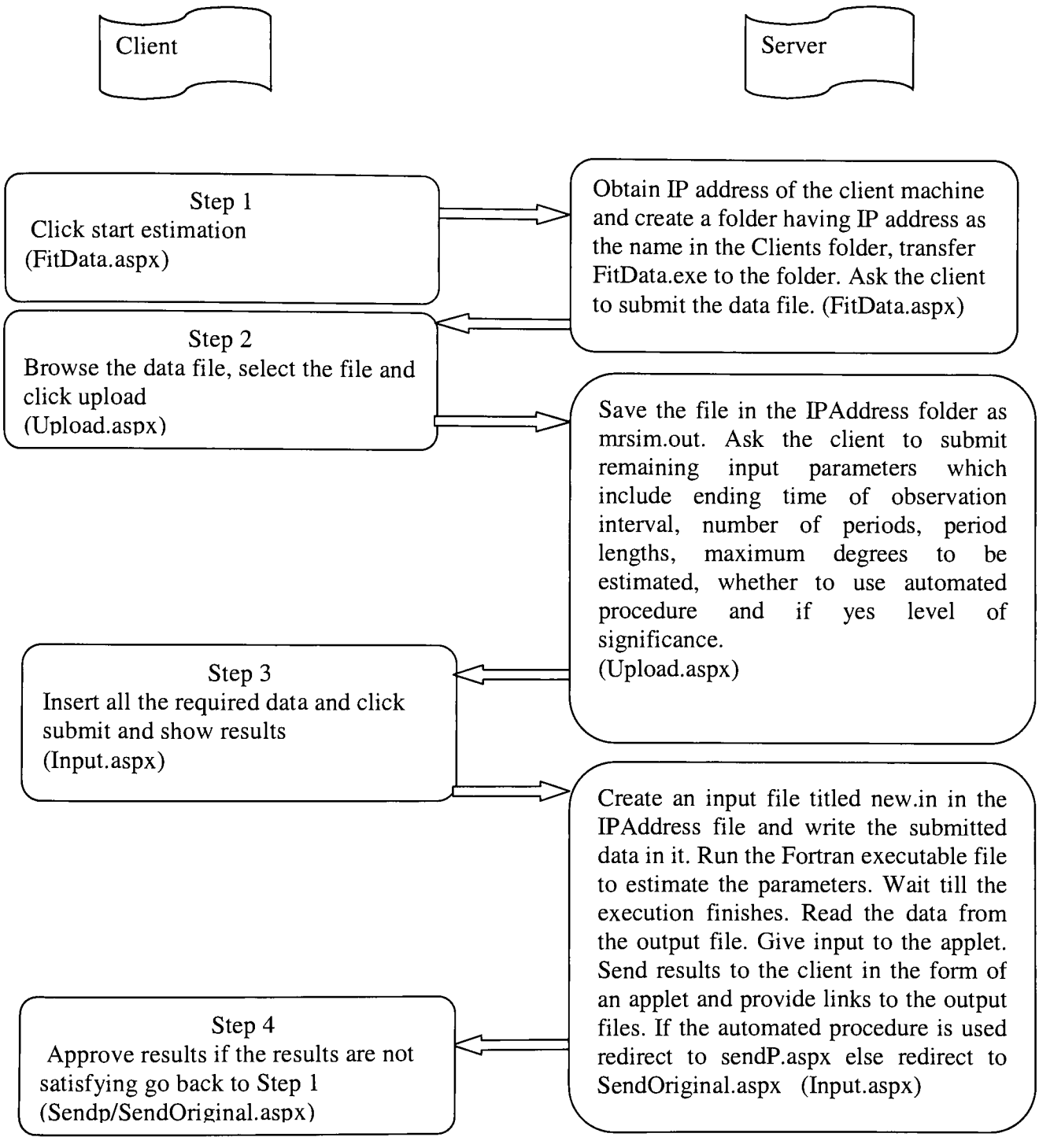

Figure 6.2 Flowchart for Web-based Data Fitting

6.3.1 Input Data

After creating folder for a specific client and transferring the respective executable file (GenerateData/ FitData), the next task is to obtain inputs for the estimation procedure.

The Fortran executable of data fitting requires two input files namely mrsim.out (arrival data) and new.in (estimation parameters). The web-interface for uploading the arrival data file is shown in Figure 6.4. The web-interface for inputting estimation

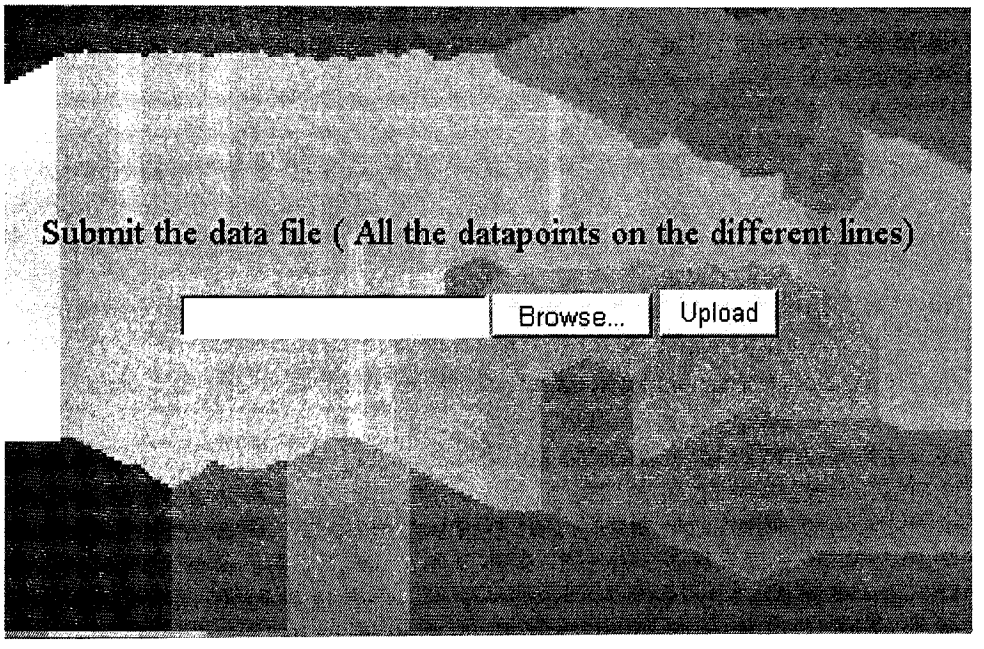

Figure 6.3 Web-Interface for Data File Uploading

parameters is shown in Figure 6.4. The user will enter values such as ending time interval, all the period lengths and their details. If the user is interested in using the automated procedure, he will be asked to provide the significance level. The created text file new.in is displayed in Appendix B.

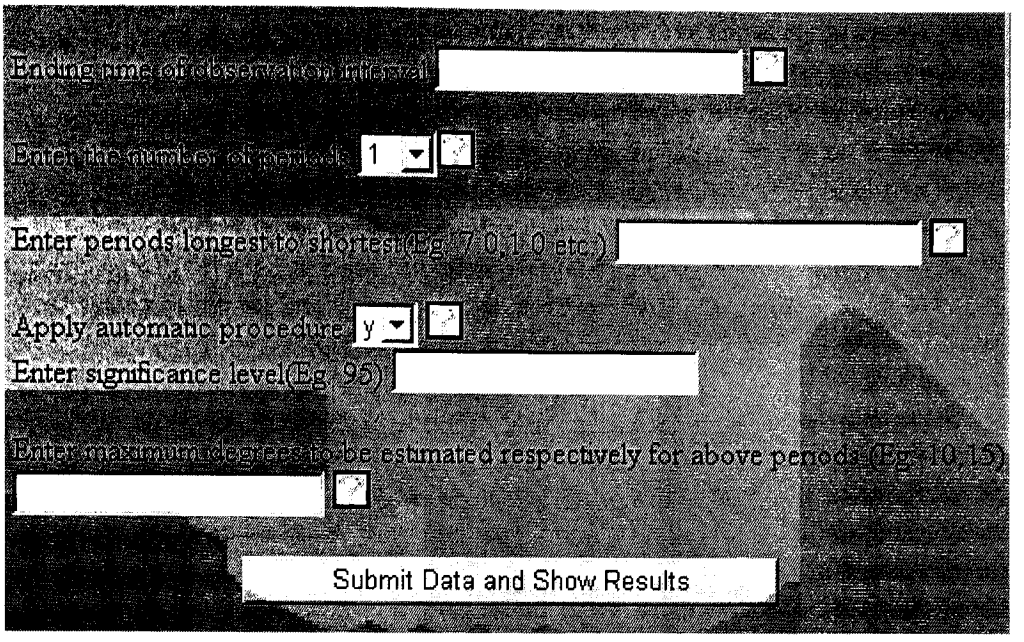

Figure 6.4 Web-Interface for Data Fitting Parameter Entry

In the case of generate data, the Fortran executable requires only one input file namely mrsim.in. The web interface for the entering the data generation parameters is shown in Figure 6.3. The user submits values including random seed, number of observations, all resolutions and their details. The created text file, mrsim.in, is shown in Appendix C. If the user wishes to use the fitted parameters to generate data, he only needs to provide two inputs including random number seed and expected number of arrivals. All other required parameters are grabbed automatically from the output file of the fitted program.

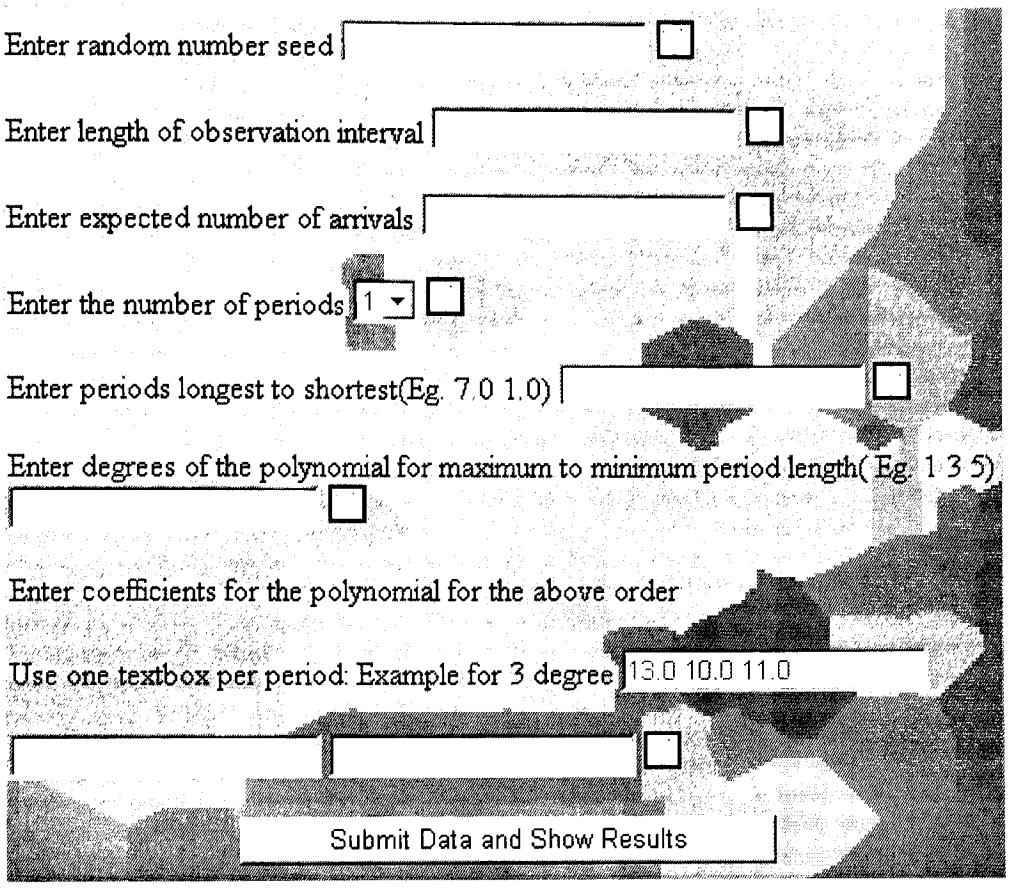

Figure 6.5 Web-Interface for Data Generation Parameter Entry

All these functionalities are obtained using HTML server controls (Chapter 3, Robinson et. al, 2001), System.IO class of .NET Framework and regular HTML.

6.3.2 Estimate Data

Once all the input data files are created, the next task is to run the Fortran executable file. To run the Fortran executable, a VB.NET Application (RunFotranExecutable) is developed. The VB.NET Application executes command prompt at the start. Then, it enters into the client directory through the command prompt and runs the Fortran Executable. Once the process is executed, the application exits the command prompt and terminates itself. The command prompt is executed using System.Diagnostics.Process Class (Jones, 2002). The same class is used to control the VB.NET application from ASP.NET page. The program for

running the Fortran executable is attached in Appendix A. All the exception handling code is added in the actual program to handle unusual situations.

6.3.3 Output Data

The execution of Fortran code results into output files, which are further used for displaying results.

The GenerateData.exe stores all the generated data into a file titled mrsim.out. This data is read from the file and send to the graph-plotting applet as parameters. The java class used for this purpose is GenerateData.class. The java class reads the parameters as x and y co-ordinates and plots them both using line graph and point graph. An example of plotted graph is shown in Figure 6.6 where data is plotted using both line and point graph. The user can see and save the data using the link 'Show Generated Data'.

The FitData.exe creates three output files namely mrfit.out, mvf.data and plots.data. The mrfit.out file has information about all the resolutions, fitted degree at each resolution and coefficients for the polynomials. The mvf.data file has original data points and fitted mean value function data points. Finally, plots.data file has all the empirical data points at each resolution. The ASP.NET page reads all the data from these three files and sends them as parameters to an applet. In the case, where the automated procedure is used, the applet is created using Graph.class. In the other case, where the automated procedure is not used, the applet is created using GraphOriginal.class. Only difference between these two classes is that Graph.class has an ability to read the parameters related to the transformed data along with the original data, whereas GraphOriginal.class is designed to read parameters related to only the original data. Along with the facility of plotting direct x and y coordinates, these classes can also estimate polynomial values by using the polynomial coordinates. The user can

select resolution number or mean value function, data type (original or transformed) and watch the graphs. The web-interface is shown in Figure 6.7. All the Java classes use Java AWT classes to display the graphics (Schildt, 2000).

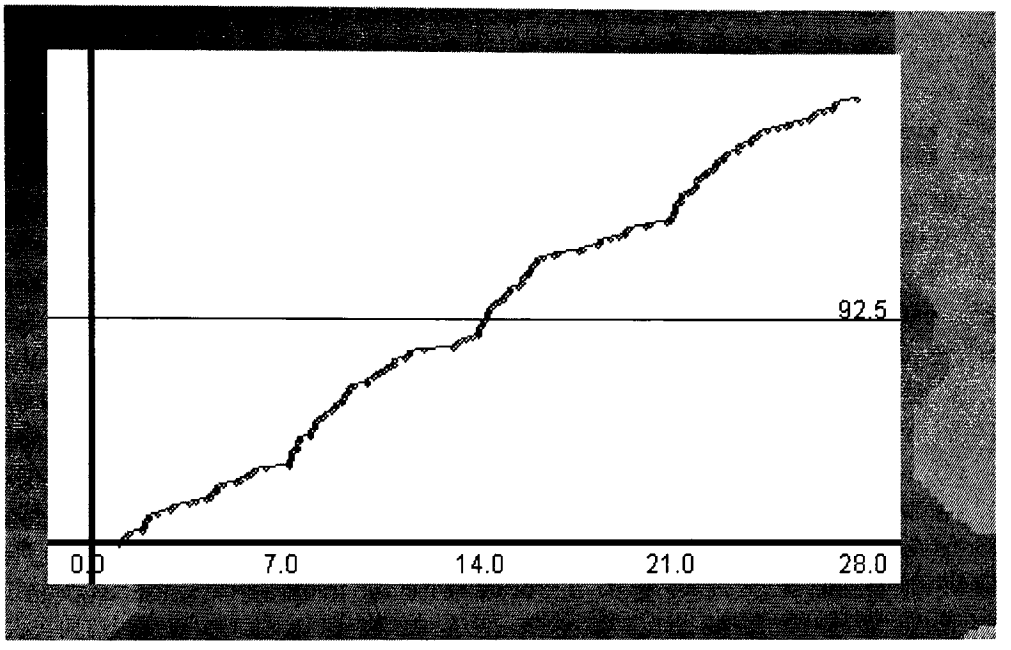

Figure 6.6 Output Graph from Data Generation Algorithm

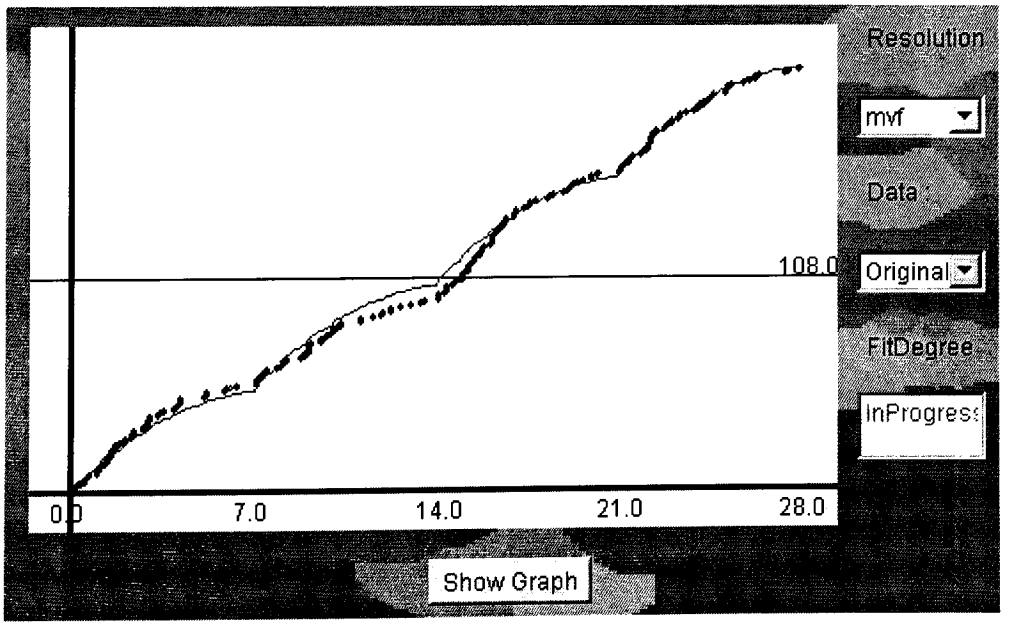

Figure 6.7 Output Graph from Data Fitting Algorithm

6.4 Web-Based Multiresolution Procedure (WBMP): An Example

Recall the example of arrival process whose instantaneous arrival rate over a time horizon of 28 days, which contains two cyclic effects including weekly and daily cyclic effects - that is, periodic rate components with periods of 1 week and one day.

To start accessing web-page go to www.rit.edu\~kuhl1\simulation .

6.4.1 Generate Data

Following steps are used to generate data from the WBMP tool.

1. Press any of the two "Generate Data" buttons from the screen as shown in Figure 6.8.

2. Press generate data button on the screen shown in Figure 6.9.

3. Enter parameters in screen shown in Figure 6.10 as given in Appendix D. Press " Submit Data and Show Results" button.

4. See results shown in Figure 6.11. Press "SHOW GENERATED DATA". The generated data will open in a new window as shown in Figure 6.12.

Copy, paste and save the data in a folder on the client side as a text file to use in simulation. This data file is used to demonstrate the data-fitting algorithm.

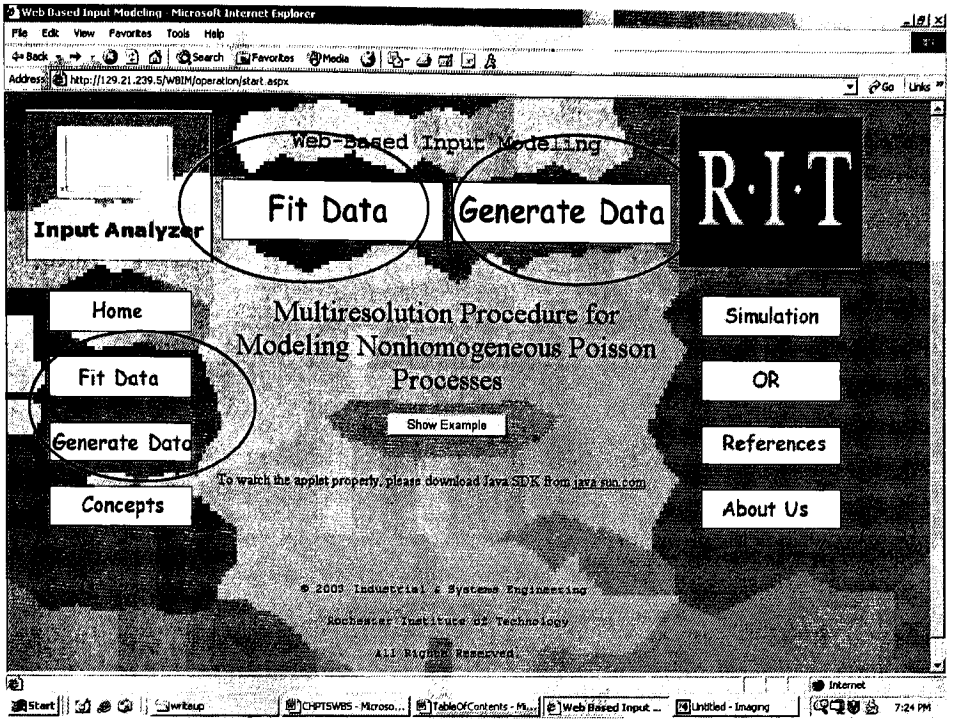

Figure 6.8 Start Up Screen for WBIM Application.

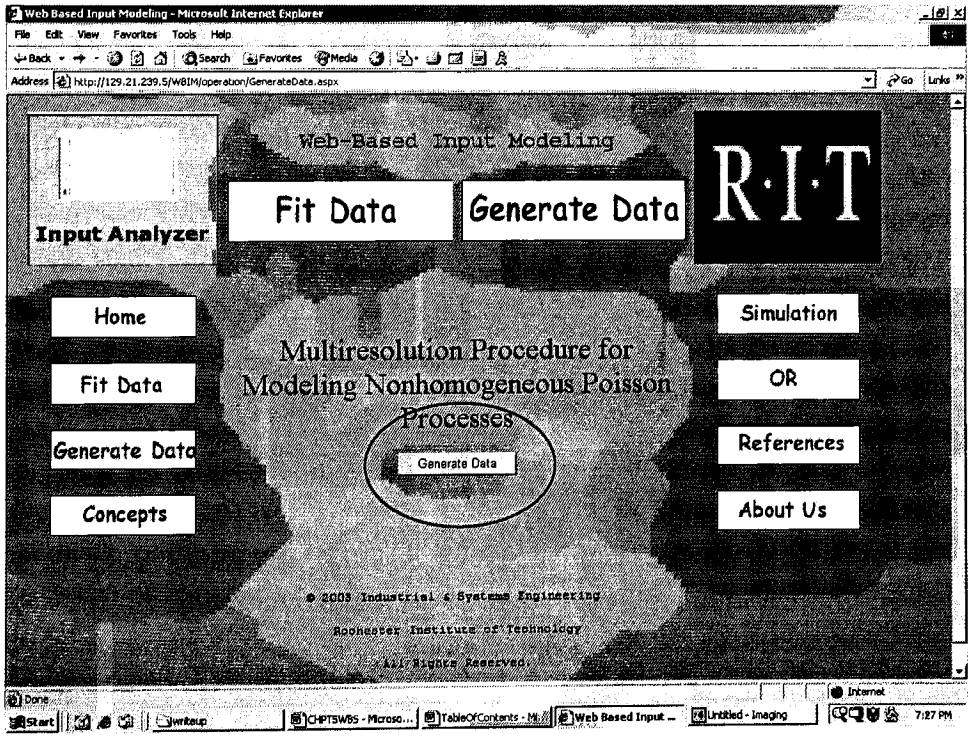

Figure 6.9 Data Generation Algorithm: Start Up Screen

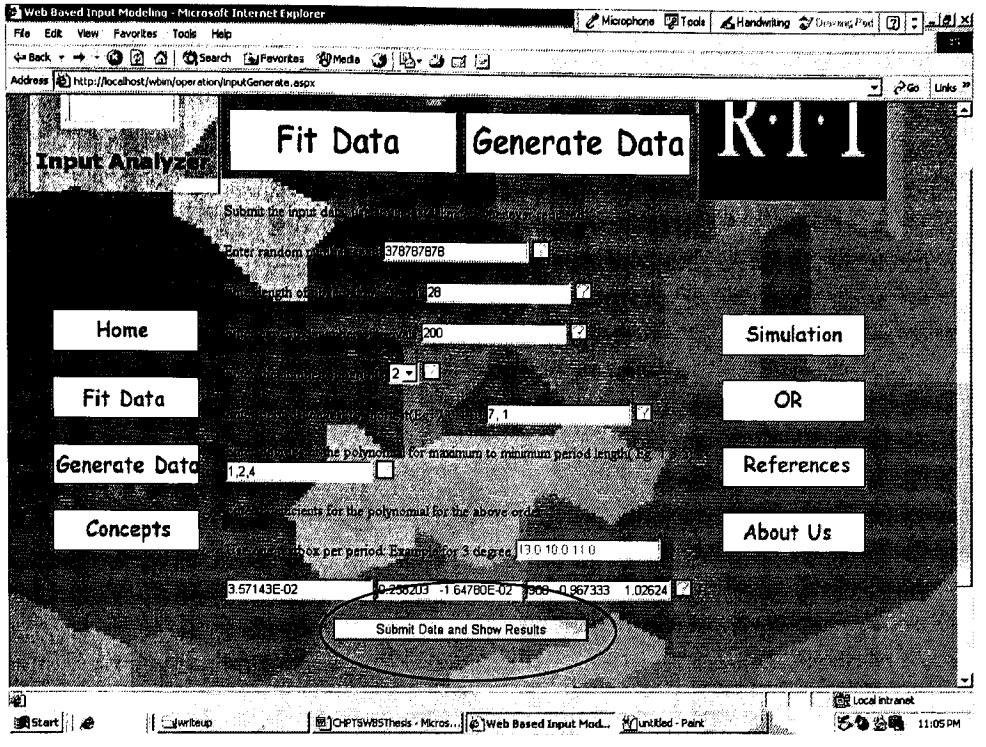

Figure 6.10 Generate Data Algorithm: Parameter Entry Screen

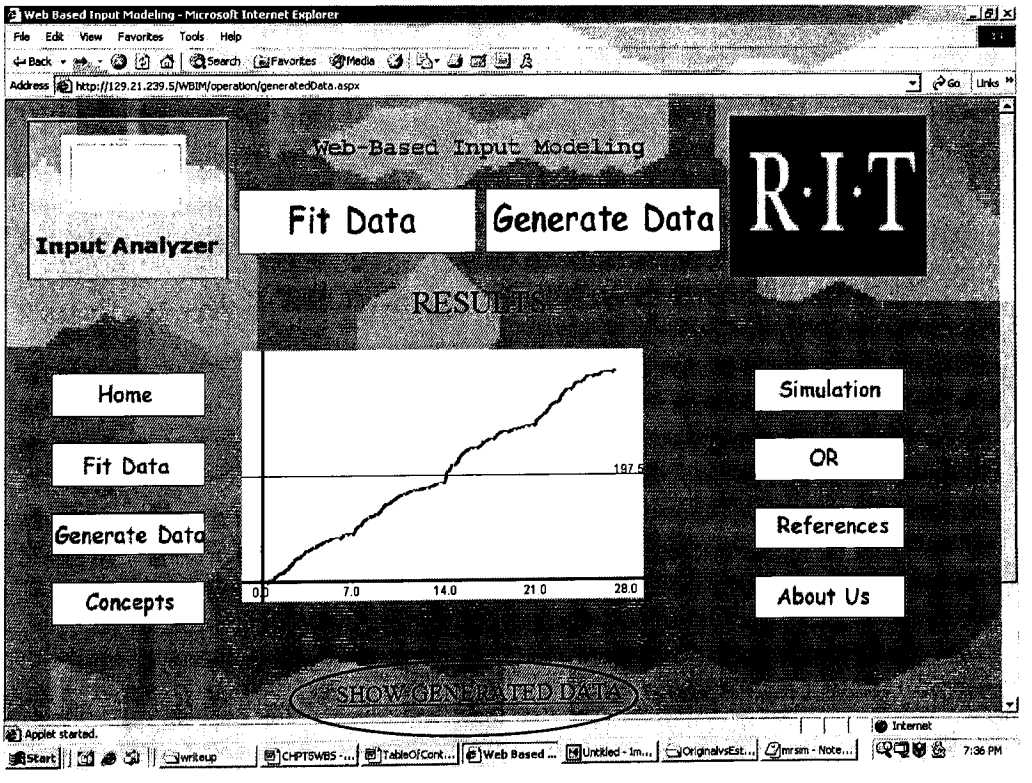

Figure 6.11 Generate Data Algorithm: Output Screen

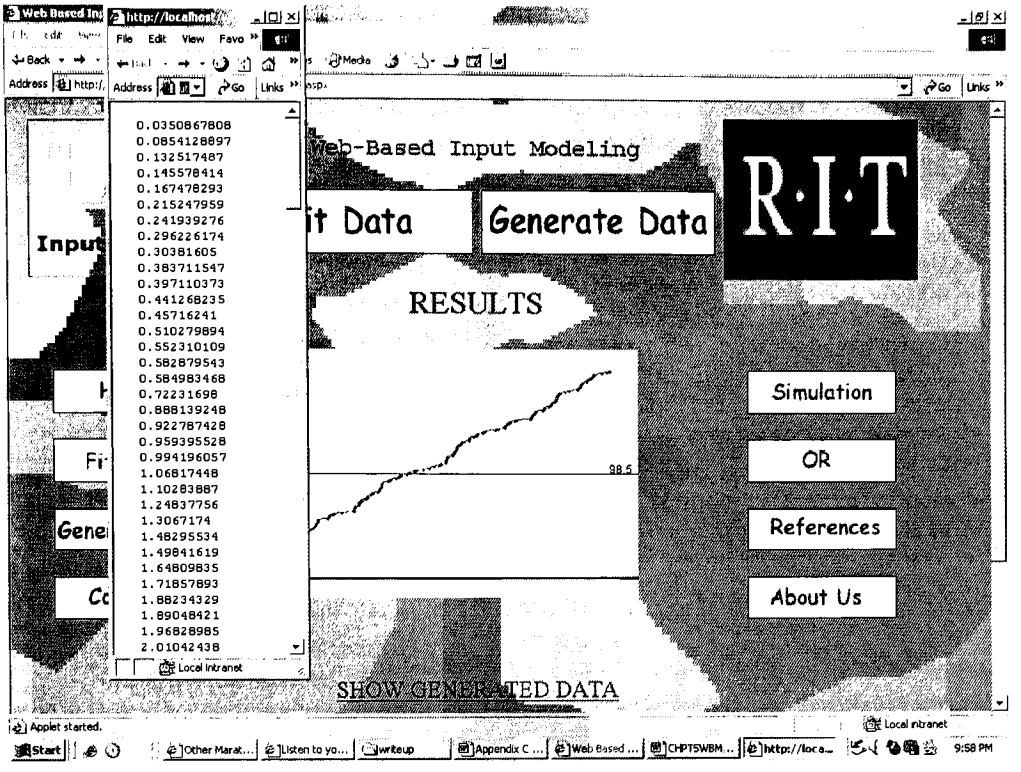

Figure 6.12 Generate Data Algorithm: Generated Data File

6.4.2 Fit Data

Following steps are used to fit multiresolution procedure to the given et of data utilizing the WBMP tool.

1. Press any of the two "Fit Data" buttons from the screen as shown in figure 6.8.

2. Press "Start Estimation" button on the screen shown in Figure 6.13.

3. Press browse and select the input data file generated in the last section from the local machine as shown in Figure 6.14, click open, and then click upload.

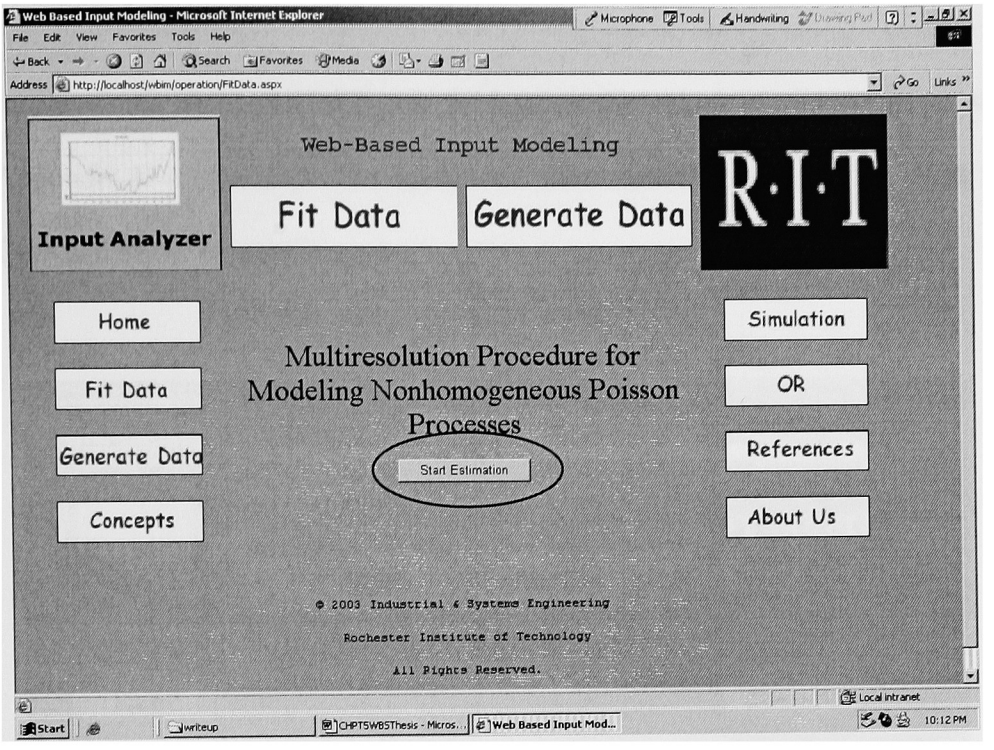

Figure 6.13 Data Fitting Algorithm: Start Up Screen

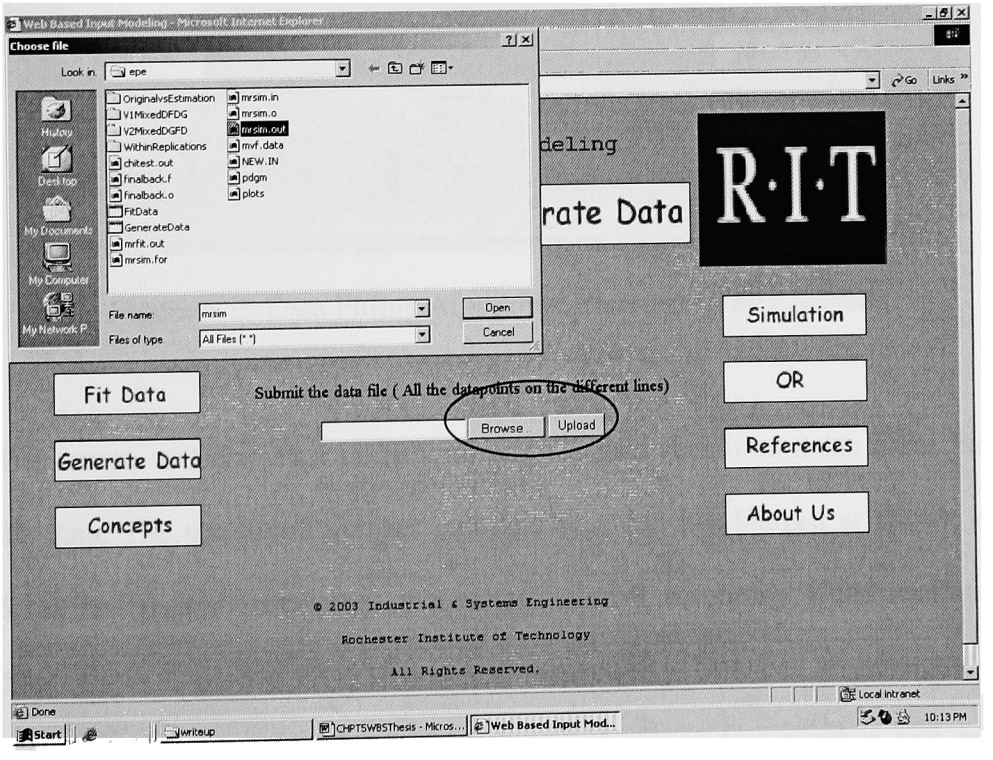

Figure 6.14 Data Fitting Algorithm: Data File Upload Screen

4. In the screen displayed in Figure 6.15, enter the values of parameters as shown in Appendix E in the respective textboxes. After entering these values, click "Submit Data and Show Results" button.

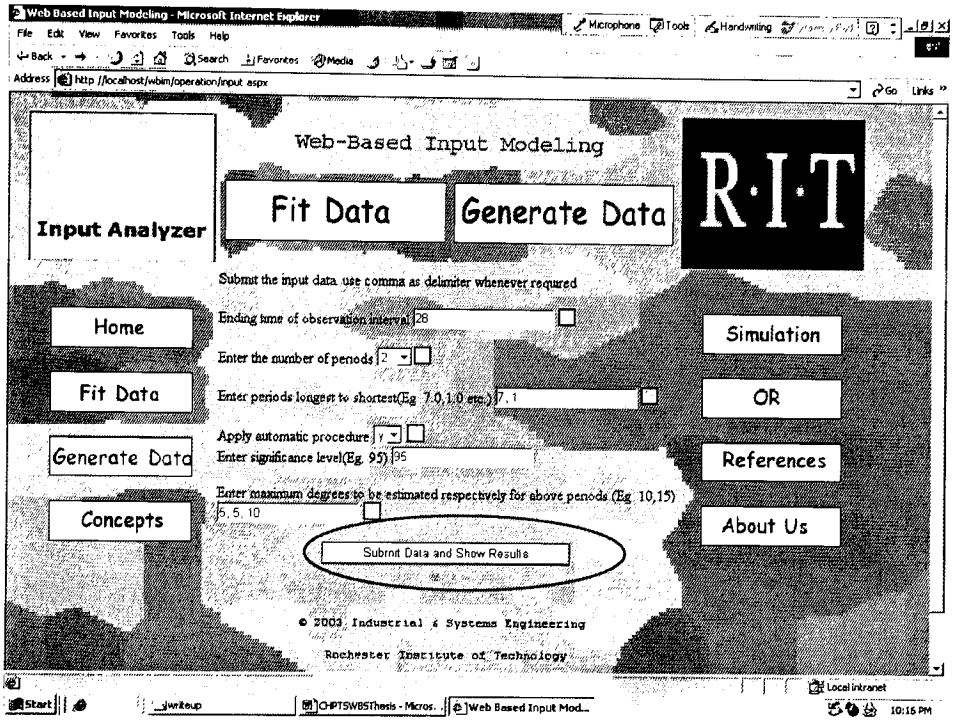

Figure 6.15 Data Fitting Algorithm: Parameter Entry Screen

5. The screen will display a graph with original data and fitted mean value function to the data. The resolution number and data type can be changed as shown in Figure 6.16. The graphs for original data at resolution 0, resolution 1 and resolution 2 are displayed in Figures 6.17, 6.18 and 6.19 respectively. Data can be generated from the fitted functions by providing parameters such as random number and number of

observations. It follows the data generation algorithm discussed before by obtaining the other required parameters automatically from the data fitting program.

The fitted mean value function data and information about fitted polynomials can be seen and obtained by clicking "SHOW MVF DATA FILE" and "SHOW OUTPUT FILE" respectively as shown in figures 6.20 and 6.21.

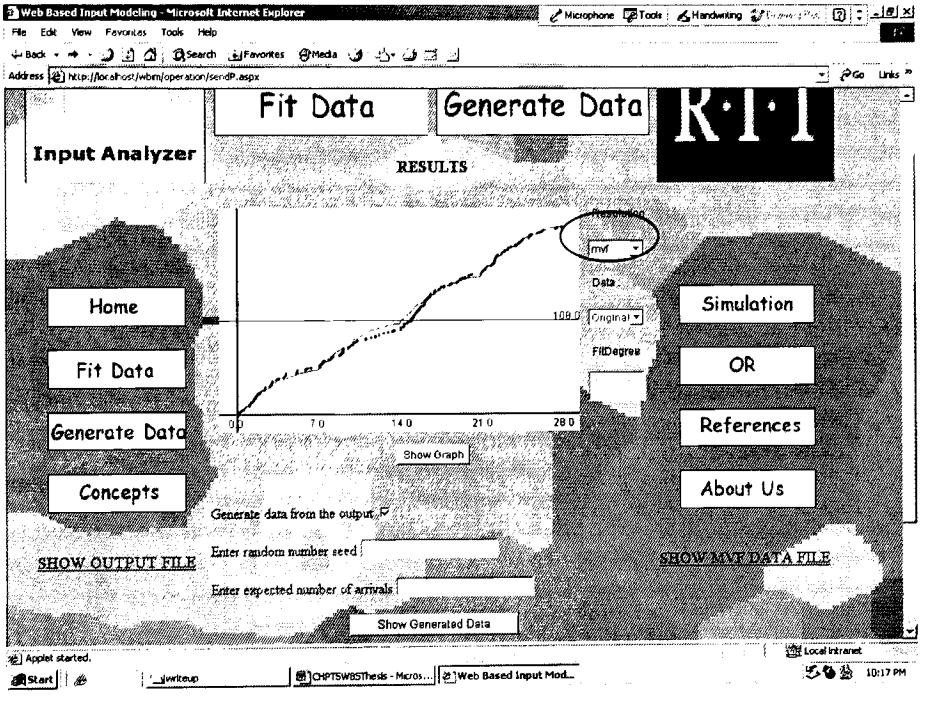

Figure 6.16 Data Fitting Algorithm: Output Screen with Original Data v/s Estimated Mean Value Function

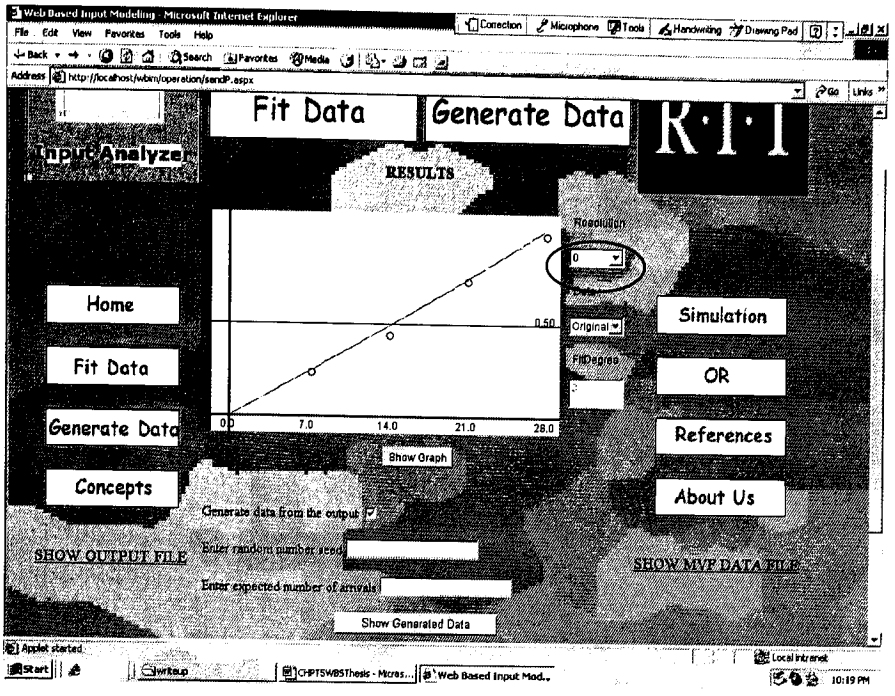

Figure 6.17 Data Fitting Algorithm: Output Screen at Resolution 0

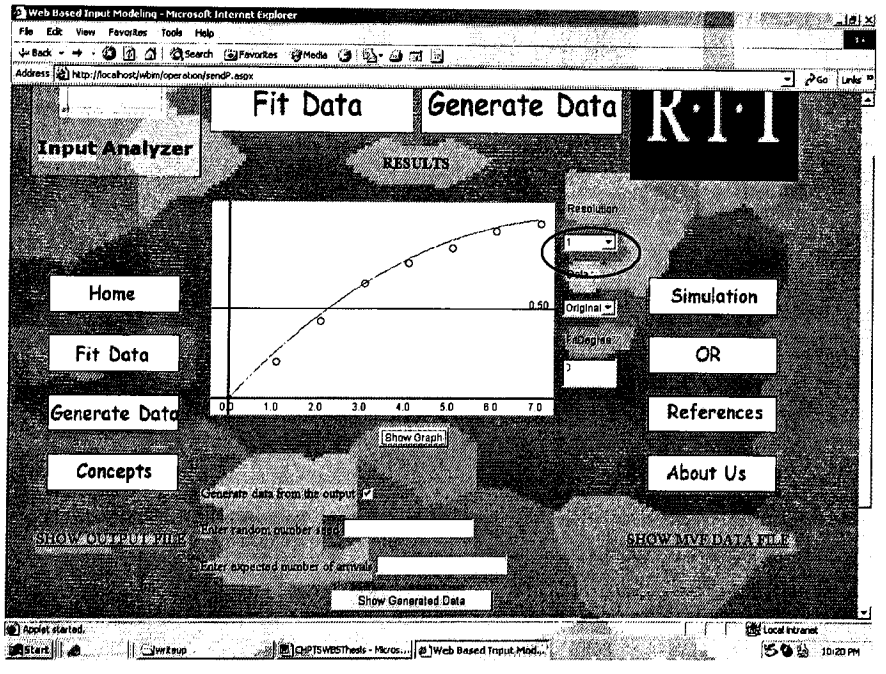

Figure 6.18 Data Fitting Algorithm: Output Screen at Resolution 1

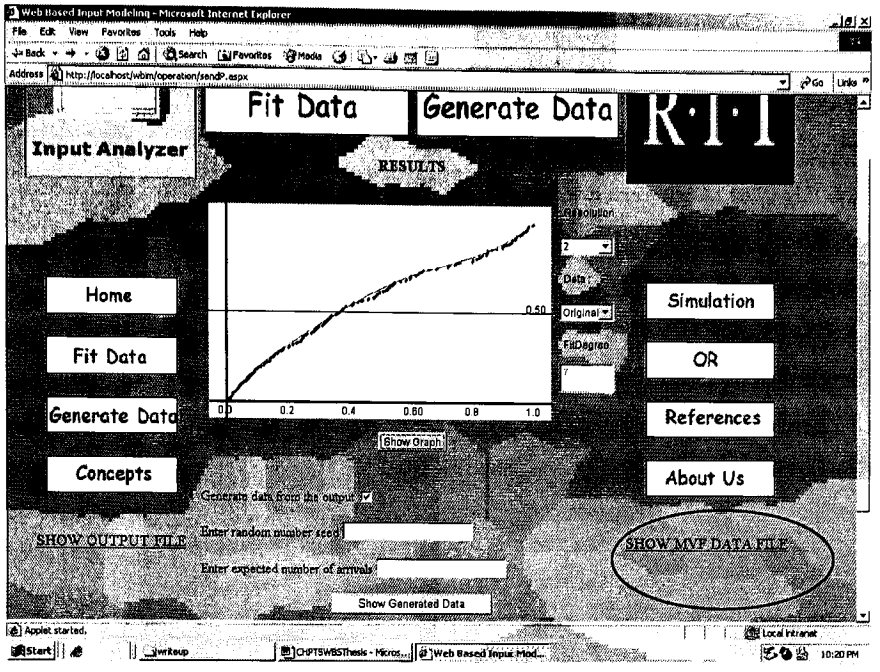

Figure 6.19 Data Fitting Algorithm: Output Screen at Resolution 2

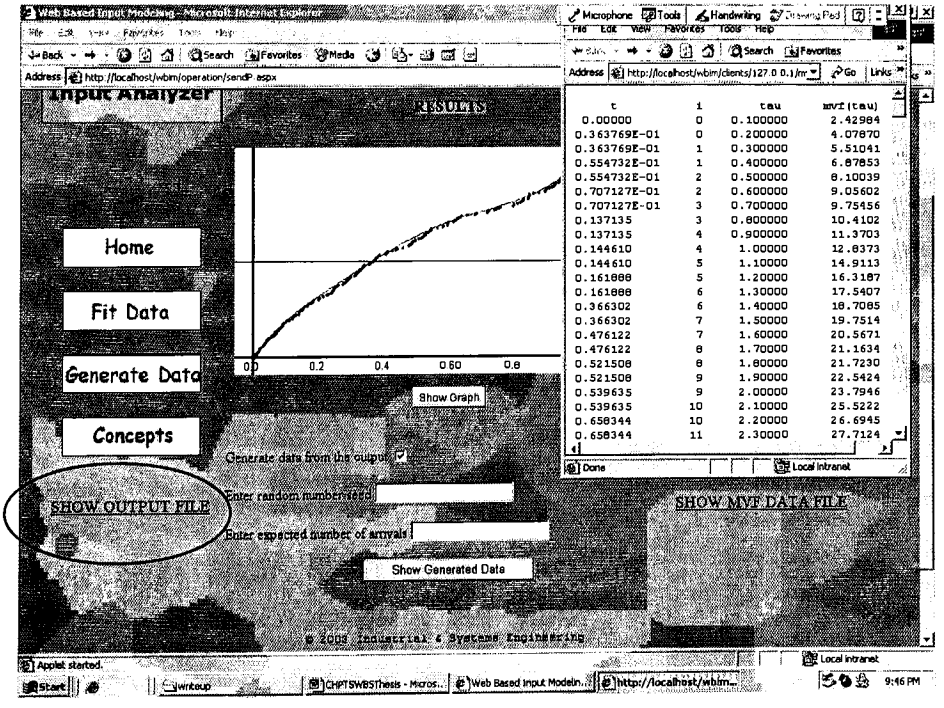

Figure 6.20 Data Fitting Algorithm: Mean Value Function Data File

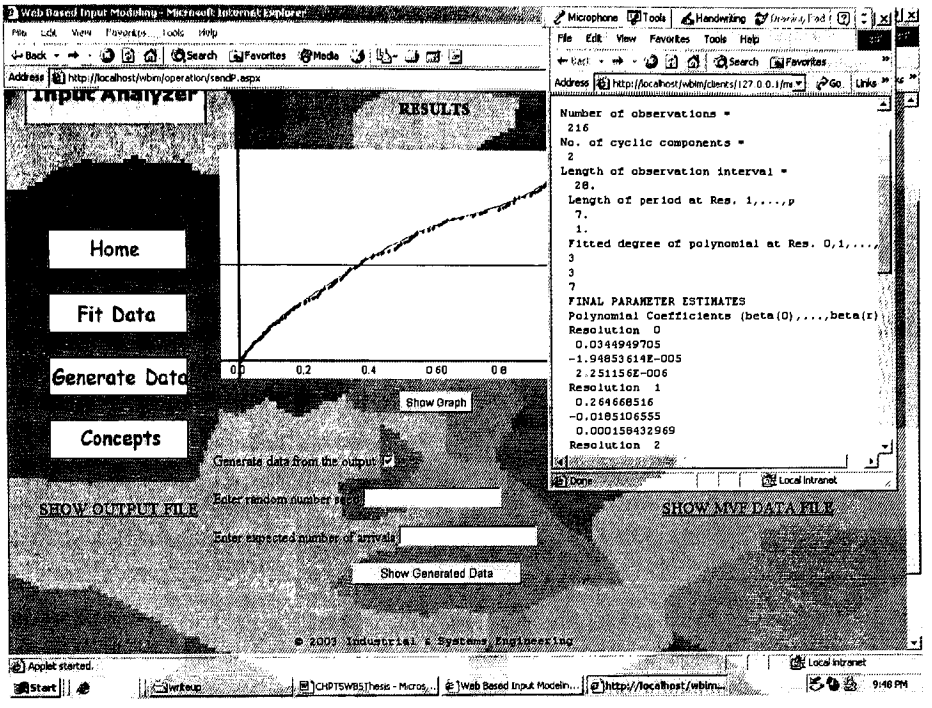

Figure 6.21 Data Fitting Algorithm: Output Data File

7. WEB-BASED SIMULATION ENVIRONMENT: A CONCEPT

As discussed in Chapter 3, web-based simulation has growing interest among the simulation community. However, most research focuses on one particular area of web-based simulation such as simulation language development in JAVA or XML, or a specific application such as supply chain management. This thesis investigates the concept of an integrated web-based simulation environment that could serve as an infrastructure for conducting simulation modeling and analysis over the Internet.

In this thesis, the basic concept of web-based simulation environment is demonstrated. As will be evident from the discussion, full development and implementation of the entire environment is a monumental task, which is beyond the scope of this research. This section discusses general issues related to web-based simulation, designs a conceptual model for web-based simulation and illustrates some of the modules with examples.

7.1 General Issues

The infrastructure of a web-based simulation environment is designed to advance and promote simulation and animation techniques, tools and models on the web by exploiting the web's platform independence and global access. One of the main advantages of a web-based simulation environment will be the reusability of programs that can be utilized by providing various simulation repositories over the web. The web can provide easy access to these repositories, which can hold modeling components, past simulations, real data archives, etc. These repositories can then be used selectively by simulation practitioners to build models and conduct experiments. To create such repositories, current web-based simulation languages such as SimJAVA, JSIM, etc. or a high level language such as JAVA can be used.

A second advantage to a web based simulation environment would be to develop environments with coherent web based support for collaborative model development; dynamic multimedia based documentation, as well as distributed execution. The ability to access multimedia on the web introduces greater potential for the use of video to show problem scenarios, and for interaction with stakeholders situated at remote locations. For example, when the running model hits an unknown combination of circumstances, an expert stakeholder might be able to determine the successful rules for advance.

By using current web technologies, this thesis work proposes to develop an infrastructure for a web-based simulation environment to incorporate the characteristics required for simulation modeling, analysis, and experimentation. The main focus of the planned simulation environment is to combine current web-based simulation methodologies and web-technologies to create a user-friendly environment and to utilize the benefits of a web-based system. A list of recommendations for simulation methodologies given by Kuljis (2000) can also be applied to our proposed web based simulation environments. Those recommendations are as follows.

1. A facility to design and/or choose a problem domain;
2. Support for data input/model specification;
3. Support for visual simulation;
4. Support for simulation statistics/results; and
5. User support and assistance.

All the above aspects will be considered while designing the web-based simulation environment. In our conceptual environment, rather than concentrating on the capabilities of

web technologies or the development of a web-based simulation language, the research work focuses on utilizing existing web-technologies and web-based simulation methodologies.

7.2 The Web-Based Simulation Environment Concept

As discussed in Chapter 1, any simulation study can be divided into six steps namely, problem formulation, creation of a conceptual model, input modeling, creation of a computer model, verification and validation, and experimentation. After formulating the problem and creating the conceptual model, the remaining tasks can be performed using the proposed web-based simulation framework. The web-based simulation framework is divided into four modules namely web-based input analysis, simulation model building, web-based verification and validation, and web-based output analysis. The concept of the web-based simulation environment is depicted in Figure 7.1.

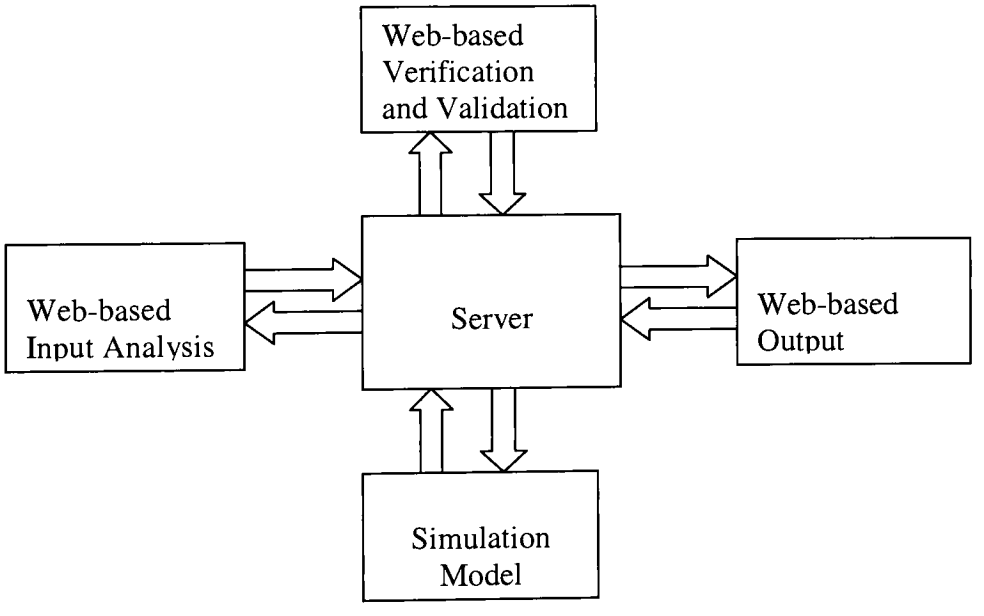

Figure 7.1 Proposed Web-Based Simulation Environment

The web-based environment will be housed on a web-server, which will serve as the simulation engine. The user can interact with the simulation environment through the four simulation modules.

The Web-based Input Analysis module will be used to fit distributions to observed data. The user will supply the data and interactively select the most appropriate distribution for the application. The model could then be returned to user through the web-page or be supplied to the simulation model.

The Simulation Model Building module will allow the user to interactively select a model from existing simulation models or components that have previously been developed and stored on the server, to modify an existing model, or to create a new model. The simulation model will then be submitted for execution and can be utilized for verification and validation. However, current web-technologies may be limiting in the near term. According to Wiedemann (2001), the internal structure and available functions of the current web are not ready for a creative development process. That means current development on the web does not support the creation of complex models. The user will be able to do use any web-based language or technology to build the model keeping in mind that the model should be platform independent.

The Web-based Verification and Validation module will allow the user to access the model from anywhere in the world. Furthermore, this will allow the model to be run by multiple users at the same time. Chat engines can be inserted inside this module so that the members can discuss the scenario. In addition, this module will interact with the server to display dynamic results. Animations, traces and performance measures of the model can be investigated by using this module, as well as making changes to model parameters.

The Web-based Output Analysis module can be used independently or along with verification and validation module. The user can analyze either already existing data or can create data using the verification and validation model and then analyze it on the web. Data is passed to the server, and then server will run the necessary algorithm and send back the results to the user.

7.3 Web-Based Simulation Environment: Design

The start page for web-based simulation environment is divided into four parts namely web-based input modeling, web-based output analysis, web-based verification and validation and simulation model uploading. The page is shown in Figure 7.2. All four parts are discussed below.

7.3.1 Web-Based Input Modeling

In this thesis, the concept of a web-based input-modeling environment has brought into existence. .In the context of simulation, there are two major tasks involved in input modeling. The first task is to fit a distribution to the data, and the second task is to generate data for the given parameters of a specific distribution. Further, the process to be modeled may be stationary or non-stationary. The general steps for input analysis can be summarized as follows:

1. Data collection,
2. Determine the type of model to be fit (discrete or continuous, or nonstationary process) depending on the type of application and data,

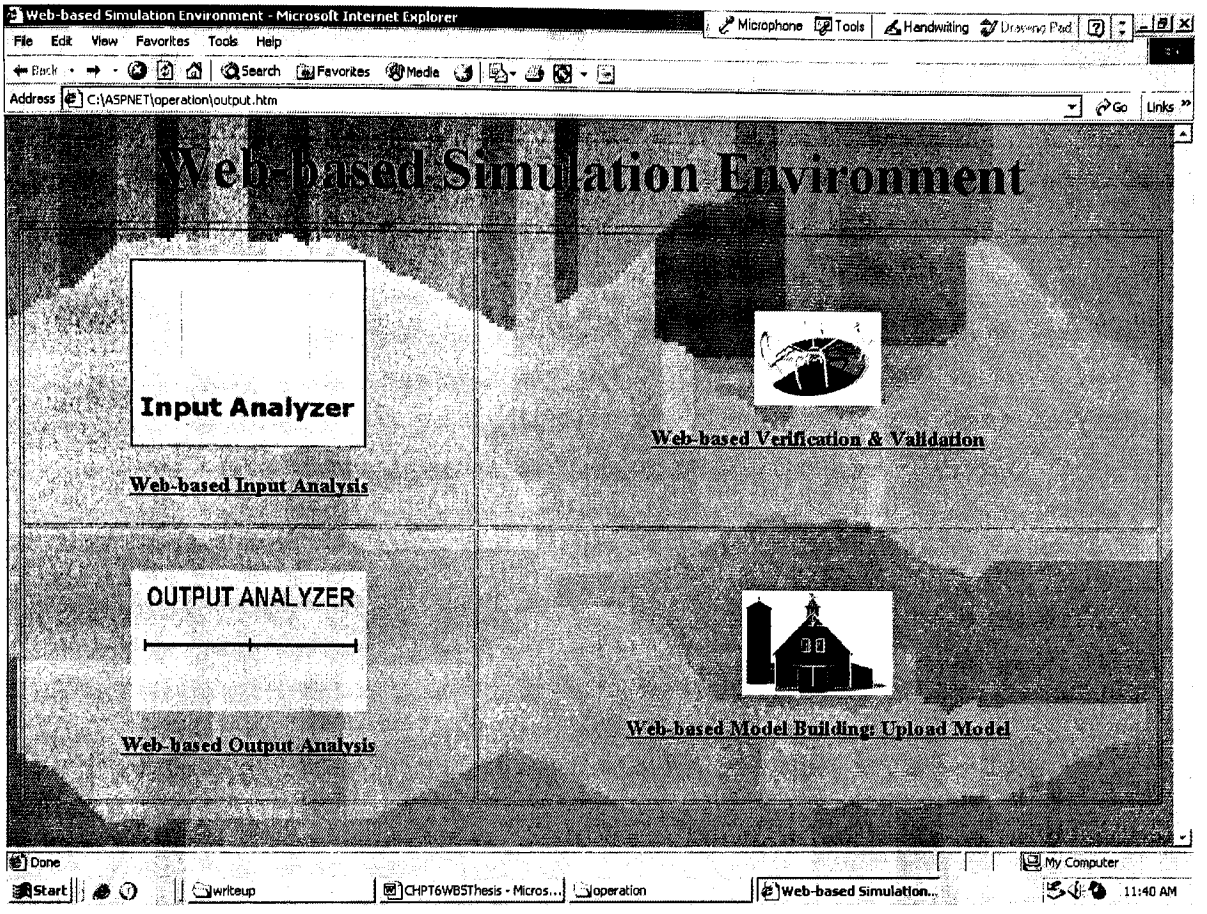

Figure 7.2 Web-based Simulation Environment Design

3. Display a graphical representation of the data such as a histogram or empirical function,
4. Fit a distribution or process to the data.
5. Perform Goodness of fit tests (Chi-square test and Konglomo-smirov test).

Theoretical distributions and stochastic processes will able to be fit on the web using web-based input analyzer. The input analyzer will allow the user to submit data files on the web and provide results including parameters, p-values and square errors on the web. This interaction is illustrated in Figure 7.2. The proposed web-based input analyzer will perform the tasks 2-5 listed above.

The requirements of a web-based input analyzer are as follows:

1. It should have capabilities to display the graphs accurately.
2. User should be able to change the number of intervals for the histogram while fitting data.
3. The output should include the details of fitting distributions such as square error, goodness of fit tests etc.
4. Environment should let the user save the results in proper format.
5. User interactive environment should allow the user to change the parameters of the distribution while generating data.
6. The data generation and data fitting should be accurate. This includes proper random number generator, exact representation of data, and precise data manipulation.

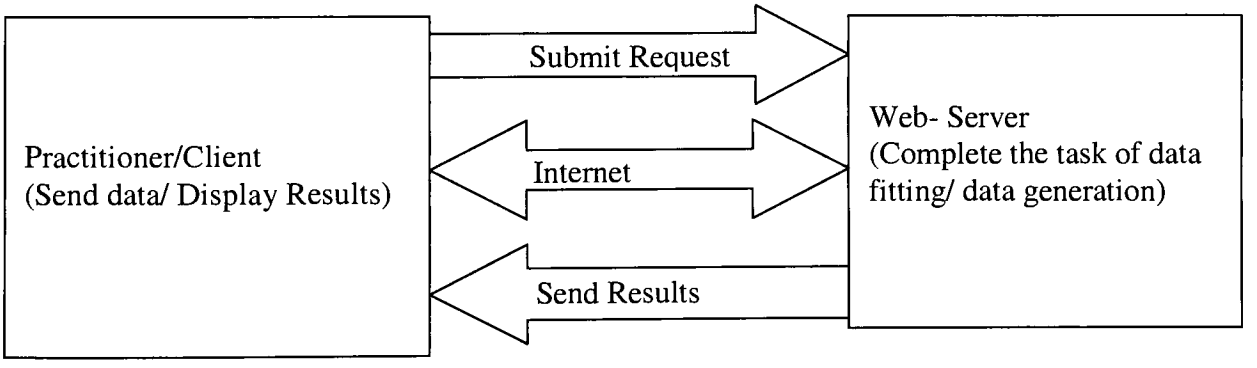

Figure 7.2 Input Modeling Environment.

A web-based input modeling (WBIM) environment, which can be set up to conduct a web-based input modeling or web-based statistical analysis for any algorithm developed in any language available in executable format on Windows 2000 platform, is developed in this

thesis. The environment has a modular design letting it to be easy to understand and modify. Once the set up for an application is completed, every time a client accesses the program, his IP address is stored in the server allowing greater security by tracking his activities. The environment is open source so that anybody can customize existing facilities and include additional features in it.

To demonstrate web-based input modeling environment, the example of the multiresolution procedure is selected. It is selected for illustration because it is one of the most complex input modeling algorithms. The complete set-up procedure can be found in Appendix E. The set-up procedure encompasses three steps. The first step is to modify web-pages to obtain data from the client on the internet. Next step is to set-up the process to run an executable on the server. The last step is to modify web-pages to send results back to the client. The same procedure is used in the next subsection to set up web-based output analysis environment.

7.3.2 Web-Based Output Analysis

The output analyzer can be developed same as the input analyzer having client server interaction. This thesis work develops a web-based output analysis environment to estimate confidence interval of the output data. Here, the application accepts an input file from the client having means of the replications in it as shown in Figure 7.3. Once the client clicks "Upload Data and Estimate CI" button, the data file is passed to the server. The output analysis algorithm is executed on the server, which estimates mean, standard deviation and confidence interval half-width at significance levels 90, 95 and 99. Further, the algorithm only considers first 30 replications for demonstration. These results are sent as inputs to the output applet and further sent back to the client. The application also allows the client to see

and save the output file. The resulting web-page is shown in Figure 7.4 and 7.5. The source code for output analysis can be found in "Output Analysis" folder on the CD.

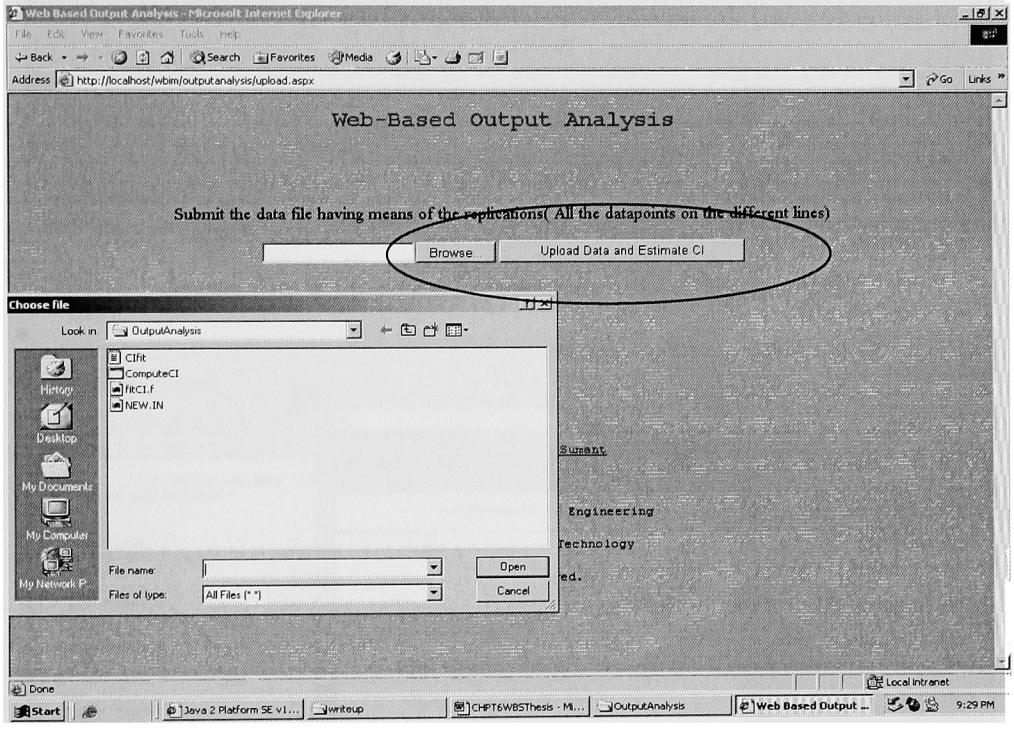

Figure 7.4 Web-based Output Analysis: Input Data File

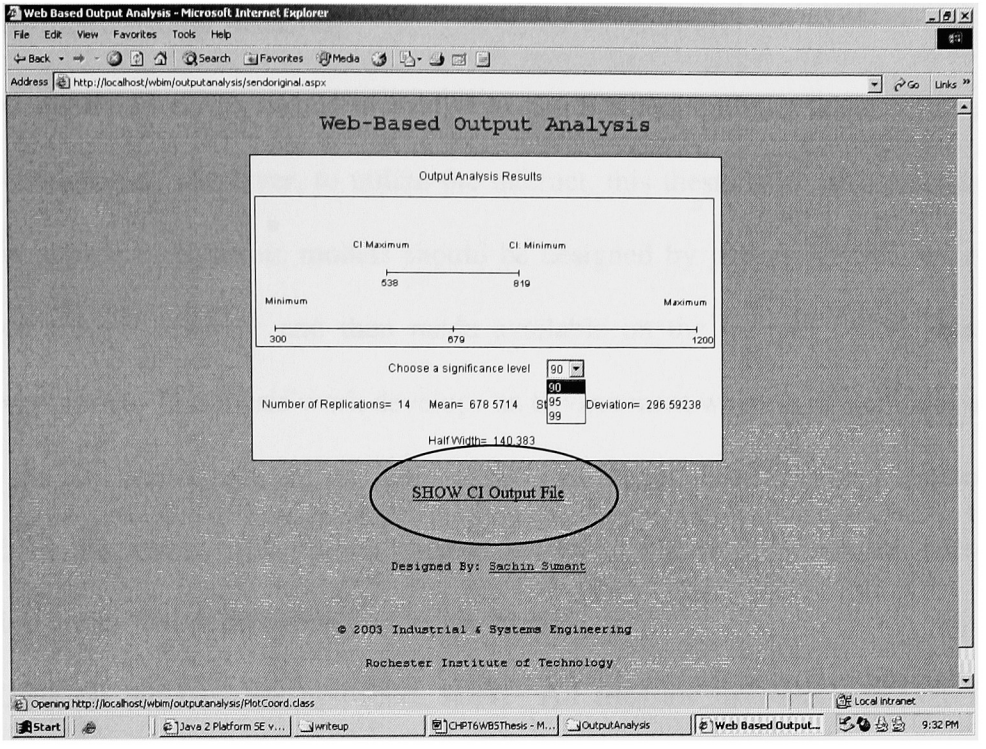

Figure 7.5 Web-based Output Analysis: Output Applet

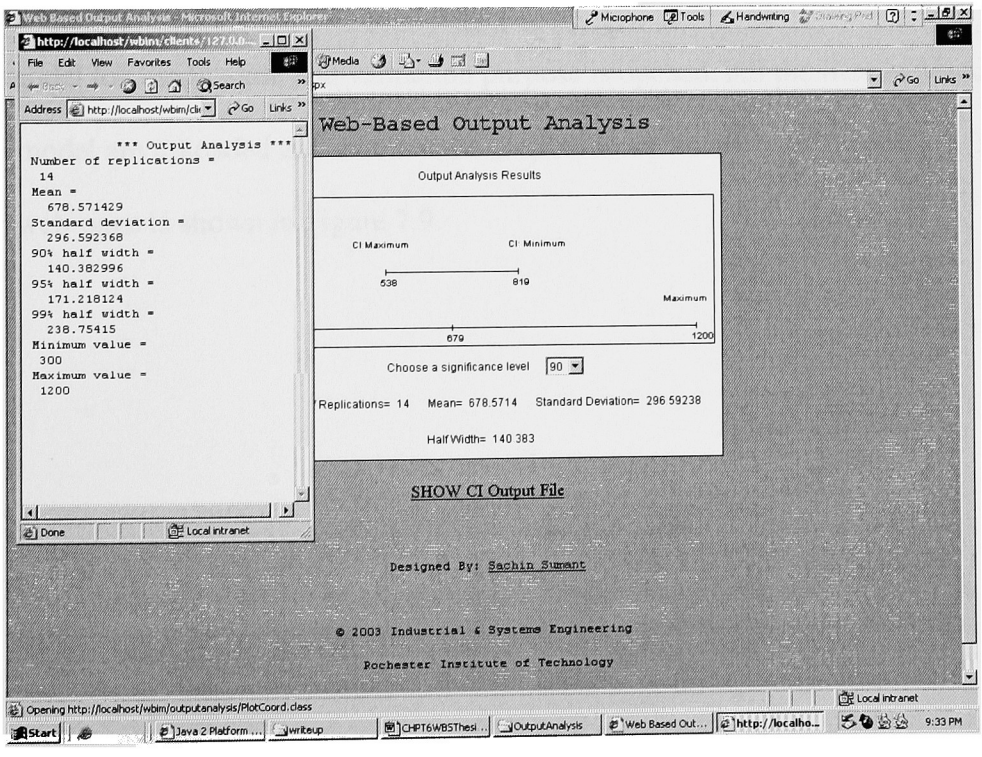

Figure 7.6 Web-based Output Analysis: Output File

7.3.3 Web-based Model Building

As discussed earlier in this section, web-based model building is not achievable with current technologies. However, to utilize the internet, this thesis work recommends that the animation applets or dynamic models should be designed by putting VRML or Java AWT objects on a local machine and then made available on the internet for verification and validation purpose. This thesis work develops an environment where a model build on a local machine in any web-language including applet, javascript, html, asp and asp.net can be uploaded to the server. The upload page is shown in Figure 7.7. All the files for this application are in "ModelBuilding" folder on the CD.

Once all the files are uploaded, select "All files are uploaded" as "y". Then, the program will ask for name of the output file as shown in Figure 7.8.

After the file is entered, press "Run the model" button and the page having the running model for verification and validation purpose is opened. All the files required for a car wash model are uploaded and name of the output file as "index.html" is entered. The resulting web-page is shown in Figure 7.9.

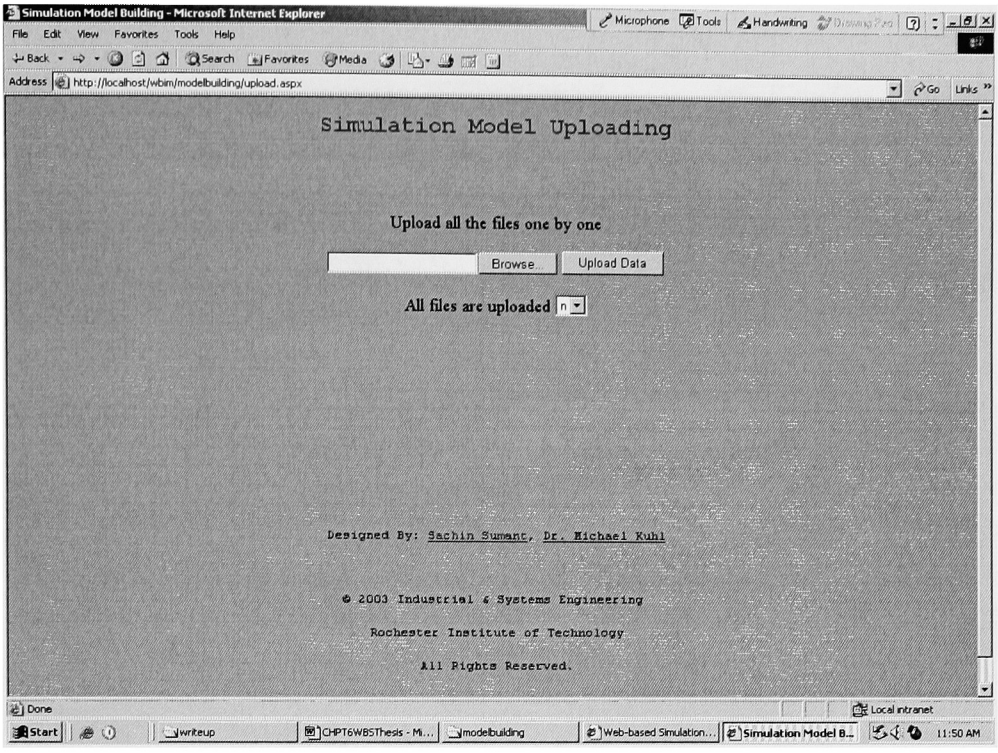

Figure 7.7 Webpage to Upload Files

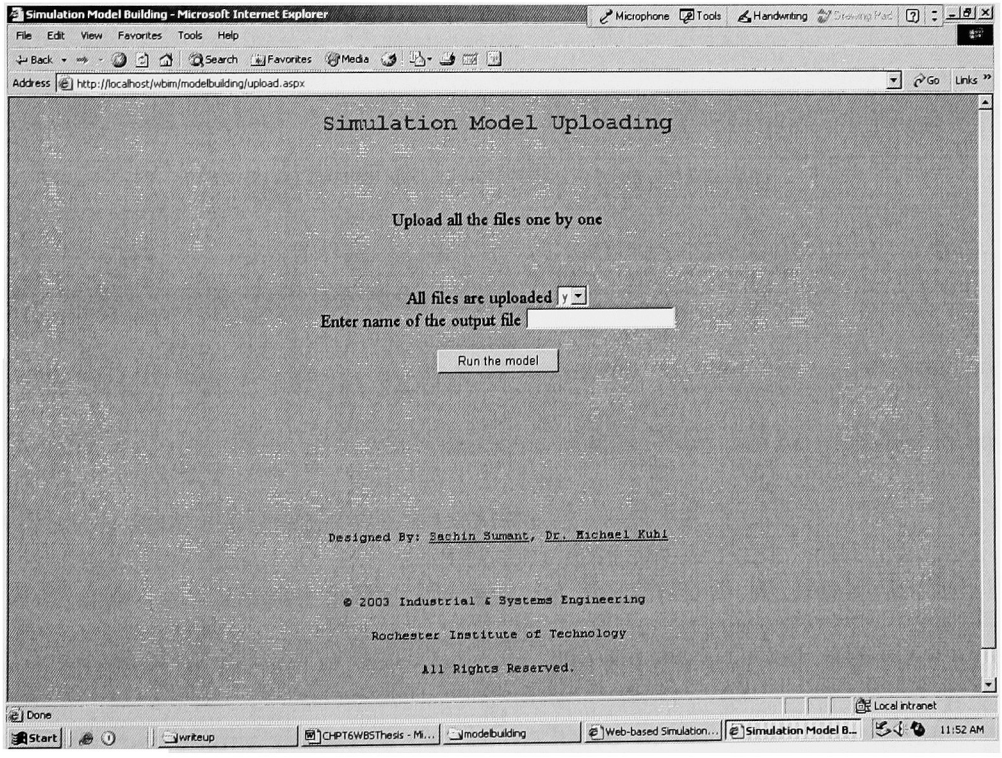

Figure 7.8 Webpage to Run the Uploaded Model

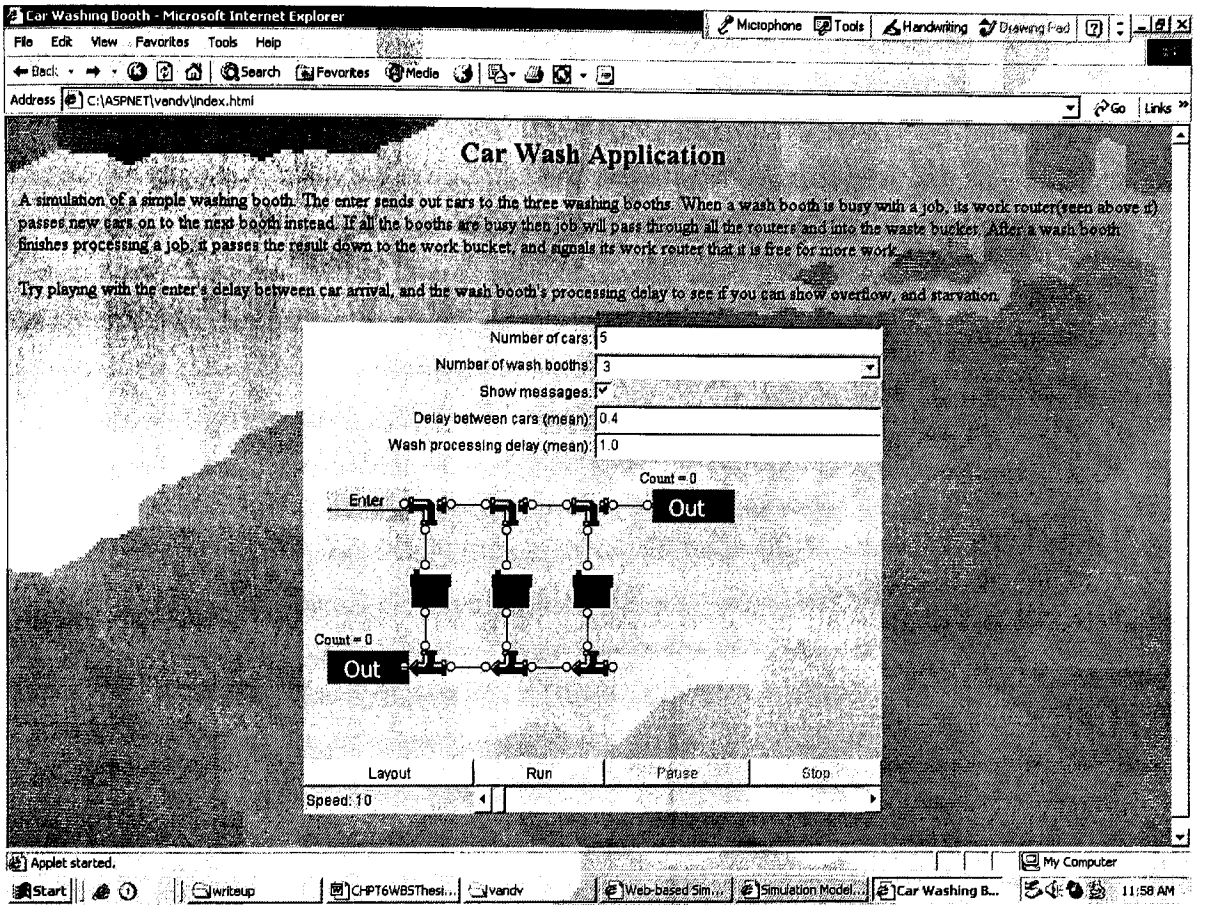

Figure 7.9 Running the Uploaded Model

7.3.4 Web-based Verification and Validation

User will be able to change number of resources or some already decided validation majors. Applet will run, show animation and give results. Plots can be included in animation, which will show what is happening in the system.

Web user can suggest some changes in the creation of model to the modeler. Nowadays, chatting on the net is very common. But for chatting one intermediate server is required. Free chat engine programs can be included in the web. User-friendly graphical interface will be created for the modeler. Anyone will be able to download the interface so

that anyone can download the applet, make changes in the applet and run the applet on his own terminal.

The java applets are created using any of the simulation languages mentioned above. The applet program and The VRML or Java AWT animation objects can be put on the web for reuse. To demonstrate web-based verification and validation, a simulation model applet for car washing is created using SimJava classes. SimJava is a toolkit for building working models. It is based around a discrete event simulation kernel and includes facilities for presenting simulation objects as animated icons on screen. Simkit provides a set of MODSIM II influenced Java classes, with no model animation but a better set of statistical classes than Simjava provides. JSIM supports a good graphical environment for displaying queues, and uses a Java database for storing results.

Since assumptions for car washing system match with static network of active entities, SIMJAVA is selected for creating applet. The applet is shown in Figure 7.10. The whole program with documentation can be found in "vandv" folder on the CD. The enter sends out cars to the three washing booths. When a wash booth is busy with a job, its work router (seen above it) passes new cars on to the next booth instead. If all the booths are busy then job will pass through all the routers and into the waste bucket. After a wash booth finishes processing a job, it passes the result down to the work bucket, and signals its work router that it is free for more work. The user can play with enter's delay between car arrival, and the wash booth's processing delay to see if it can show overflow, and starvation. Number of wash booths can be changed from 1-5.

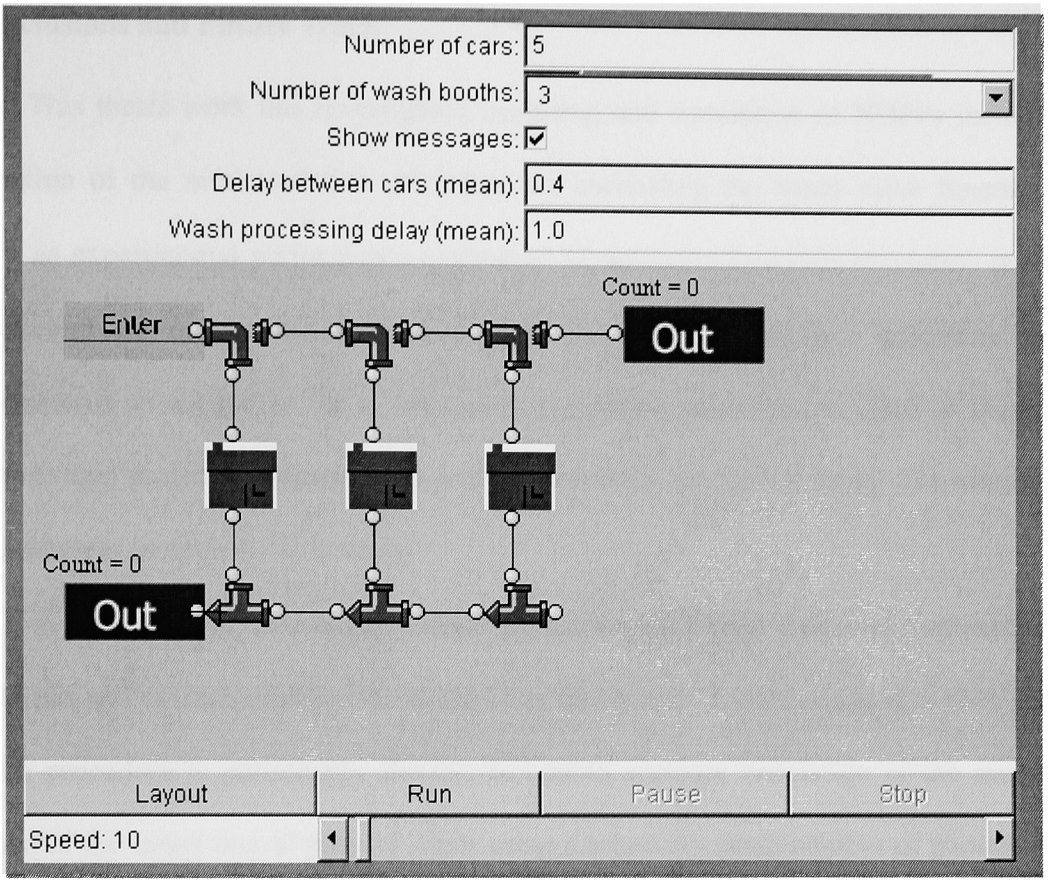

Figure 7.10 Web-based Verification & Validation: An Example

8. Conclusions and Future Work

This thesis work has investigated modeling and simulation of NHPPs including the automation of the mutiresolution procedure for estimating the mean value function of an NHPP, an experimental performance evaluation of the automated multiresolution procedure, the development of a web-based environment for the multiresolution procedure, and the demonstration of a concept for a web-based simulation environment. Each of these issues have provided useful simulation tools and opportunities for further study. The contributions and challenges include the following.

A completely automated procedure for estimating arrival processes that exhibit long-term trends and nested periodic effects has been developed. Evidence has also been provided that the procedure is sufficiently flexible to enable accurate estimation of the mean value function of the target arrival process while using a relatively small number of parameters that can be efficiently estimated from the history of process. The experimental performance evaluation further reveals accuracy of the automated procedure. It demonstrates not only the accuracy in fitting polynomials but also the ability of the automated procedure to handle highly complex data and parameters.

Future extensions of the multiresolution procedure will include (*a*) a graphical user interface for invoking the procedure in popular simulation software systems; and (*b*) efficient software for generating independent replications of a fitted arrival process. Another direction for future work is to develop statistical tools for validating the basic assumptions of the multiresolution procedure in arrival processes to which the procedure might be applied.

The design and development of the web-based multiresolution procedure for modeling nonhomogeneous Poisson processes provides a highly useful, accessible and interactive tool to fit and generate data having trends and periodic effects.

Future work includes optimizing the tool by adding more features such as allow the user to save the graphs, create login and passwords for regular visitors to store and maintain their history.

The Web-based Simulation Environment is presented as a concept rather than a completed project. This thesis work has discussed current research in web-based simulation environment. Current web-based simulation languages require hard core Java programmer to understand and implement the model. The web-based simulation environment should promote and spread simulation and animation techniques, tools and models on the web to utilize platform independence and global access.

The infrastructure for web-based input modeling is designed so that it can be used to deploy any statistical analysis or input modeling algorithm. The web-based input modeling infrastructure can run any executable (Win32.exe) file programmed in any language including Fortran, C, C++ etc., which is compiled on Windows platform. The user needs to be familiar with any server side language such as ASP, ASP.NET or JSP. However, since WBIM is developed in ASP.NET, it is recommended for use so that the user only requires updating some lines of code to interact with the client on the internet. Future work involves facilitating as many input modeling algorithms and statistical analysis tools to the internet users utilizing the WBIM infrastructure.

The output analysis environment can provide an infrastructure to create more sophisticated output analysis tool having more flexibility in terms of significance levels,

number of replications and format of input data. In addition, data optimization tools can be developed to provide complex data analysis to the decision makers.

The web-based verification and validation concept is illustrated by an example. However, to obtain the web-based verification and validation concept into reality requires large amount of efforts as compared to web-based input modeling and web-based output analysis. Interesting areas for future web based simulation packages to develop are cooperative usage and creation of simulations, component reuse using the JavaBeans model and applets, distributed simulations, and more dynamic interaction and interpretation of simulation results. Online applets, classes for creation, documentation help for different types of application will be put on for global access.

Lastly, this thesis work also reveals that that web-based simulation model building is going to be a time consuming and non-practical task by using the current web-based technologies and tools. Upcoming technologies might provide facilities to work with the complexity of the simulation model building. Then, web-based simulation modeling building will be possible.

Future work for maximum utilization of internet for simulation modeling involves integration of all the modules including web-based input modeling, web-based output analysis, web-based model building and web-based verification and validation.

REFERENCES

Alfonseca, M., J. Lara, and H. Vangheluwe. 2001. Web based simulation of systems described by partial differential equations. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J.S. Smith, D. J. Medeiros, and M. W. Rohrer. 629-636.

Banks, J. 1998. *Handbook of simulation: Principles, methodology, advances, applications, and practices*. New York: John Wiley & Sons, Inc.

Bard, Y.1974. *Nonlinear parameter estimation*. New York: Academic Press.

Bhat, U. N. 1972. *Elements of applied stochastic processes*. New York: John Wiley & Sons, Inc.

Brieman, L. 1973. *Statistics: with a view toward applications*. Boston: Houghton Mifflin Company.

Casella, G., R. L. Berger. 1990. *Statistical inference*. Belmont: Duxbury Press.

Cinlar, E. 1975. *Introduction to stochastic processes*. New Jersey: Prentice-Hall, Inc.

Cox, D. R., V. Isham. 1980. *Point processes*. New York: Chapman & Hall.

Feldman, R. M., C. Vadez-Flores. 1996. *Applied probability and stochastic processes*. Boston: PWS Publishing Company.

Fishwick, P. A. 1996. Web based simulation: some personal observations. In *Proceedings of the 1996 Winter Simulation Conference*. 772-779.

Fred Howell, Ross McNab. Institute for Computing Systems Architecture Division of *Informatics, University of Edinburgh.* "SIMJAVA". *<http://www.dcs.ed.ac.uk/home/hase/simjava/> (15 December, 2001)*

Guru, A., P. Savory, and R. Williams . 2000. A web based interface for storing and executing simulation models. In *Proceedings of the 2000 Winter Simulation Conference*, ed., J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick. 1810-1814.

Healy, K. J., and R. A. Kilgore.1997. Silk: A Java based process simulation language. In *Proceedings of the 1997 Winter Simulation Conference*, ed., S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson 475-482.

Jain, S., C. Lim, B. Gan, and Y. Low. 1999. Criticality of detailed modeling in semiconductor supply chain simulation. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. farrington, H.B. Nembhard, D. T. Sturrock, nad G. W. Evans,888-896.

Jasna Kuljis, R. J. Paul. 2000. A review of web based simulation: whether we wander?. In *Proceedings of 2000 Winter Simulation Conference*, ed., J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick. 1872-1881.

Johnson, M. A., S. Lee, and J. R. Wilson. 1994. Experimental evaluation of a procedure for estimating NHPPs having cyclic behavior. *ORSA Journal on Computing*, 6:4, 56-68

Kao, E. P. C., S Chang. 1988. Modeling time-dependent arrivals to service systems: a case in using a piecewise-polynomial rate function in a NHPP. *Management Science*, 34, 1367-1379

Kelton, W. D., R. P. Sadowski, D. A. Sadowski. 2002. *Simulation with Arena.* New York: The McGraw-Hill Companies, Inc.

Kilgore, R. 2001. Open source simulation modeling language. In *Proceedings of 2001 Winter Simulation Conference*, ed., B. A. Peters, J.S. Smith, D. J. Medeiros, and M. W. Rohrer, 607-613.

Kilgore, R. and K. Healy. 1998. Introduction to Silk and Java based simulation. In *Proceedings of 1998 Winter Simulation Conference*, ed, D. J. Medeiros, E. F. Watson, J. S. Carson and M. S. manivannan.327-334.

Klein, U., S. Straburger and J. Beikirch. 1998. Distributed simulation with JavaGPSS based on the high level architecture. In *Proceedings of the 1998 International Conference on Web-based Modeling & Simulation*, ed. P. A. Fishwick, D.R.C. Hill and R. Smith, 85-90.

Knight, K. 2000. *Mathematical statistics.* New York: Chapman & Hall/CRC.

Kuhl, M. E., J. R. Wilson. 1997. Estimating and simulating Poisson processes having trends or multiple periodicities. *IIE transactions*, 29, 201-211.

Kuhl, M. E., J. R. Wilson. 1999. User's manual for mrfit: software for estimating Nonhomogeneous Poisson processes having trends or nontrignometric cyclic effects. Technical report, Department of Industrial Engineering. North Carolina State University, Raleigh, NC.

Kuhl, M. E., J. R. Wilson. 2000. Least square estimation of nonhomogeneous Poisson processes. *Journal of Statistical Computer Simulation*, 67(2000) 75-108.

Kuhl, M. E., J. R. Wilson. 2001. Modeling and simulating Poisson processes having trends or nontrignometric cyclic effects. *European Journal of Operational Research*, 133 (2001) 566-582.

Kuhl, M. E. 1997. Modeling, estimation and simulation of nonstationary point processes. Ph. D. Dissertation, Department of Industrial Engineering, North Carolina State University, Raleigh, NC.

Law, A. M., and Kelton, W. D. 2000. *Simulation modeling and analysis*. Boston: The McGraw-Hill Companies Inc.

Lazeolla, G. and A. D'Ambrogio. 1998. A web based environment for the reuse of simulation models. In *Proceedings of the 1998 International Conference on Web-based Modeling & Simulation*, ed. P. A. Fishwick, D.R.C. Hill and R. Smith, 37-42.

Lee, S., J. R. Wilson, and M. M. Crawford. 1991. Modeling and simulation of a NHPP having cyclic behavior. *Communications in Statistics-Simulation and Computation*, 20, 777-809

Leemis, L. 1991. Nonparametric estimation of the cumulative intensity function for a Nonhomogeneous Poisson process. *Management Science,* 37,886-900.

Lewis, P. A. W., G. S. Shedler. 1776a. Simulation of NHPPs with log linear rate function. *Biometrica*, 63, 501-505.

Lewis, P. A. W., G. S. Shedler. 1776b. Statistical analysis of non-stationary events in a database system. *IBM Journal of Research and Development*, 20,465-482.

Lewis, P. A. W., G. S. Shedler. 1779. Simulation of NHPPs with degree two exponential polynomial rate function. *Operations Research*, 27:5, 1026-1040.

Lorenz, P., H. Dorwarth, K. Ritter, and Schriber. 1997. Towards a web based simulation environment. In *Proceedings of the 1997 Winter Simulation Conference*, ed., S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson, 1338-1344.

Miller, J. A., Y. Ge, and J. Tao. 1998. Component based simulation environments: Jsim as a case study using Java Beans. In *Proceedings of 1998 Winter Simulation Conference*, ed, D. J. Medeiros, E. F. Watson, J. S. Carson and M. S. manivannan. 373-381.

Nahi, N. E. 1969. *Estimation theory and applications*. New York: John Wiley & Sons, Inc.

Nelson, B. L. 1995. *Stochastic modeling, analysis and Simulation*. New York: McGraw-Hill Higher Education.

Nyhoff, L., S. Leestma. 1988. *FORTRAN77 for engineers and scientists.* Second edition. New York: Macmillan Publishing Company.

Pidd, M., N. Oses, and R. Brooks. 1999. Component based simulation on the web. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. farrington, H.B. Nembhard, D. T. Sturrock, nad G. W. Evans.1438-1444.

Pritsker, A. A. B. 1998. Life and death decisions: organ transplantation allocation policy analysis. *OR/MS today*, 25, 22-28.

Ratkowsky, D. A. 1990. *Handbook of nonlinear regression models.* New York: M. Dekker, Inc.

Ross, S. M. 1989. *Introduction to probability models.* Fourth edition. San Diego: Academic Press, Inc.

Sarjoughian, H. S. and B. P. Ziegler. 1998. DEVSJAVA: basis for a DEVS based collaborative M&S environments. In *Proceedings of the 1998 International Conference on Web-based Modeling & Simulation*, ed. P. A. Fishwick, D.R.C. Hill and R. Smith, 29-35.

Tao, Y. and S. Guo. 2001.The design of a web based training system for Simulation analysis. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J.S. Smith, D. J. Medeiros, and M. W. Rohrer, 645-652.

Taylor, H. M., S. Karlin. 1994. *An introduction to stochastic modeling.* San Diego: Academic Press, Inc.

Walpole, R. E., R. H. Myers and S. L. Myers. 1998. *Probability and Statistics for Engineers and Scientists.* Prentice Hall, Inc.

Wiedemann, T. 2001.Simulation application service providing. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J.S. Smith, D. J. Medeiros, and M. W. Rohrer, 623-628.

Appendix A: VB.NET Program to Run an Executable on the Server

```
Sub Main()
// Read command line arguments namely client's folder name/ip address and name of
// Fortran executable file.
    Dim arguments() As String = Environment.GetCommandLineArgs()
// Define and initialize a process
    Dim myProcess As Process = New Process()
    Dim s As String
// Assign process name as command prompt and assign other required parameters for
// execuation of the process
    myProcess.StartInfo.FileName = "cmd.exe"
    myProcess.StartInfo.UseShellExecute = False
    myProcess.StartInfo.CreateNoWindow = True
    myProcess.StartInfo.RedirectStandardInput = True
    myProcess.StartInfo.RedirectStandardOutput = True
    myProcess.StartInfo.RedirectStandardError = True
// Start running command prompt invisibly on the server side
    myProcess.Start()
//  Define and stream writers and readers to write commands in the command prompt
    Dim sIn As StreamWriter = myProcess.StandardInput
    sIn.AutoFlush = True
    Dim sOut As StreamReader = myProcess.StandardOutput
    Dim sErr As StreamReader = myProcess.StandardError
```

```
// Go into the client's directory
        sIn.Write("cd " + arguments(1) & _
            System.Environment.NewLine)
// Write the name of Fortran executable
        sIn.Write(arguments(2) & System.Environment.NewLine)
// Run the Fortran executable and exit
        sIn.Write("exit" & System.Environment.NewLine)
        s = sOut.ReadToEnd()
        If Not myProcess.HasExited Then
           myProcess.Kill()
        End If
  // close all the processes, readers and writers
        sIn.Close()
        sOut.Close()
        sErr.Close()
        myProcess.Close()
    End Sub
```

Appendix B: Input Data for Web-Based Data Fitting Algorithm

/* Ending time of observation interval */

28

/* number of periodic components */

2

/* Periods longest to shortest */

7,1

/* Apply automatic procedure to determine the degree of polynomial */

y

/* Significance level */

95

/* Max. Degree to be fitted at each resolution */

5, 5, 10

Appendix C: Input Data for Data Generation Algorithm

/* Random number seed */

378787878

/* Length of obsevation interval (S) */

28.0

/* Expected number of arrivals in [0,S] */

200

/* Number of periodic components */

2

/* Periods (Longest to shortest) */

7.0, 1.0

/* Degree of Polynomial */

1, 2, 4

/* Polynomial Coefficients */

1st TextBox Enter: 3.57143E-02

2nd TextBox Enter: 0.258203 -1.64780E-02

3rd TextBox Enter: 1.83200 -0.890900 -0.967333 1.02624

Appendix D: Set Up Procedure for Web-Based Input Modeling Environment

The software required for this application is:

1) Windows 2000 Professional
2) IIS Server: Free with any Windows 2000 CD.
3) .NET framework SDK: Download is free from www.microsoft.com.
4) Java 1.3.1 SDK, if the user wants to display results in the form of an applet (Optional): Download is free from www.java.sun.com.
5) Application executable (data analysis algorithm) in the form of .exe which is compiled on Windows 2000 Professional.

There are no specific hardware requirements for this application. However, sometimes any of the above software might have any requirements, which can be found from its documentation.

The name "user" is used for the person who is going to use the infrastructure for setting up an application and "client" is used for the person who is going to utilize the application. To start setting up the application, download the aspnet.zip file from the CD, select download source files. Or copy all folders from "Web-based Input Modeling" folder on the CD to a new folder called ASPNET on C drive. After unzipping the folder, the user will find a folder called ASNET having five subfolders namely operation, images, clients, executable, runexecutable. Store the ASPNET folder in C: drive. Set up a virtual directory from IIS server mapping to this folder. Make sure the names of all the folders are not changed. The virtual directory is named as "WBIM". Any name can be given to this directory. As soon as the client accesses the first web-page start.aspx, it's IP address is grabbed and a subfolder utilizing the IP address as name is created in the "Clients" folder. The program for this functionality is Sub page_Load()

```
// Request user IP address
    Session("Folder")= Request.UserHostAddress
// Create a directory
    System.IO.Directory.CreateDirectory("C:\ASPNET\clients\"+Session("Folder"))
```

There are three main steps to finish the setup. These steps are as follows:

1) Set up web-pages for obtaining data from the client.

In the operation folder, there are three pages namely fitdata, input.aspx and upload.aspx, which can be used to get the data receiving functionality. When the client clicks startEstimation button, the fitdata page copies the executable from Executable folder to the client's folder using following code:

```
Session("Folder") = Request.UserHostAddress
Try
// Change name of the executable (FitData.exe) file as needed
System.IO.File.Copy("C:\ASPNET\Executable\FitData.exe","C:\ASPNET\clients
\"+Session("Folder")+"\FitData.exe")
  Catch Except As IOException
  End Try
```

Input.aspx can be used to allow user to enter input parameters. For example, the ending time of the interval is entered in a textbox and program for putting textbox into the web-page is,

```
<asp:label id="message1" text="Ending time of observation interval" runat=
"server"/>
 <asp:textbox id="endingtime" runat="server" />
```

Similarly, program for a dropdownlist for entering number of periods is,

```
 <asp:label id="message" text="Enter the number of periods"
runat="server"/>
 <asp:dropdownlist id="list1" runat="server">
 <asp:listitem>1</asp:listitem>
 <asp:listitem>2</asp:listitem>
 <asp:listitem>3</asp:listitem>
 <asp:listitem>4</asp:listitem>
```

```
<asp:listitem>5</asp:listitem>
<asp:listitem>6</asp:listitem>
<asp:listitem>7</asp:listitem>
<asp:listitem>8</asp:listitem>
<asp:listitem>9</asp:listitem>
<asp:listitem>10</asp:listitem>
</asp:dropdownlist>
```

Once the client enters all the information and clicks submit data, the btnSub_Click function gets called. This function will create a text file in the client's folder. The program for this functionality is,

```
// Read user's ip address
Session ("Folder") = Request.UserHostAddress
// Create an input file in the client's folder
// Name of the file (NEW.IN) should be changed according to the requirement.
Dim writer As StreamWriter =
File.CreateText("C:\aspnet\clients\"+Session("Folder")+"\new.in")
// Write data  in the input file
writer.WriteLine("/* Ending time of observation interval */")
writer.WriteLine(Request.Form("endingtime"))
writer.WriteLine("/* number of periodic components */")
writer.WriteLine(Request.Form("list1"))
```

The first task the user needs to do is add, delete or modify any ASP.NET controls such as textbox, dropdownlist, label, etc as per the requirement. The second task is to add, delete and modify the file writing program to suit to respective input file. The user can create any number of input files depending on the need.

Upload.aspx can be used to upload a datafile from client's machine. The client browses a file and clicks upload. Following code gets executed on the server side and the file is saved in the clients folder:

```
// read user's ip address
Session("Folder")= Request.UserHostAddress
If fupUpload Is Nothing Then
Else
// Name of the file (MRSIM.OUT) should be changed according to the
requirement
fupUpload.PostedFile.SaveAs("C:\ASPNET\clients\"+Session("Folder")+"\MRSI
M.OUT").
```

This page can be used directly without any major changes.

2) Set up the system to run an executable at the server.

Store a copy of an executable in the "executable" folder. When a client's folder is created, a copy of the executable is created in it. Next, in the input .aspx, after an input file is created, the executable is run in the clients folder by using following code.

```
// Request the client's IP address
        Session("Folder")= Request.UserHostAddress
// Run the executable in the client's folder
// Change the name of the executable according to the requirement
Dim myProcess As Process
=System.Diagnostics.Process.Start("C:\aspnet\RunExecutable\bin\RunFortranExe
cutable.exe","C:\aspnet\clients\"+Session("Folder")+" FitData")
myProcess.WaitForExit()
```

3) Set up web-pages to send results back to the client.

To show the results, SendP.aspx is used. SendP.aspx has a link to the output file as shown below:

```
// Change the name of the output file (mrfit.out) as per the requirement
<p align="center"><div align="center"><font size="4">
```

```
<a Href=<%="..\clients\"+Session("Folder")+"\mrfit.out"%> Target=
"_blank">SHOW OUTPUT FILE</a></font></div></p>
```

The user can add any number of links to any number of output files stored in the clients folder by using above code.

The same steps can be followed to set up any number of executables ensuring that the file names are not matching for different executables. Even if they match, the infrastructure allows you to create different folders in the client's folder for different executables. This concept is used to set up the input modeling environment for data generation program. In the webpage GenerateData.aspx, a folder titled "GenerateData" is created in the client's folder using following code:

```
// Obtain the client's IP address
Session("Folder") = Request.UserHostAddress
// Create a folder named GenerateData in the client's directory
System.IO.Directory.CreateDirectory("C:\ASPNET\clients\"+Session("Folder")
+"\GenerateData")
Try
// Copy the GenerateData executable in the GenerateData Folder.
System.IO.File.Copy("C:\ASPNET\Executable\GenerateData.exe","C:\ASPNE
T\clients\"+Session("Folder")+"\GenerateData\GenerateData.exe")
  Catch Except As IOException
  End Try
```

The other web-pages used for this application are inputGenerate.aspx to obtain input data and execute the application, and generatedData.aspx to display the output file. The important change in the approach is that generateData folder name is added in the respective paths for input, output and executable files.

To see the full functional application visit http://www.rit.edu/~kuhl1/simulation with many added functionalities such as displaying output in the form of graphs, allowing generation of data from the fitted functions etc.

Appendix E: CD-ROM Containing All Programs

Folder 1. ASPNET: Files related to web-based simulation environment including web-based input modeling, web-based output analysis, web-based model uploading and web-based verification and validation.

Folder 2. FortranPrograms: Programs related to multiresolution data fitting and data generation along with programs for experimental performance evaluation

Folder 3. Web-basedInputModeling: Programs to set up web-based input modeling environment

Producer for
Building complex processes
K. Sunny