Efficient Simulation of Non-Poisson Non-Stationary Point Processes to Study Queueing Approximations

Ni Ma [1], Ward Whitt [2]

Abstract

A nonstationary point process can be efficiently simulated by exploiting a representation as the composition of a rate-one process and the cumulative arrival rate function, provided that an efficient algorithm is available for generating the rate-one process, as is the case for stationary renewal processes, Markov modulated Poisson processes and many other processes. Overall efficiency can be achieved by constructing a table of the inverse cumulative arrival rate function when it is not explicitly available.

Keywords: simulation, nonstationary point process, time-varying arrival rate, inverse function, queues with time-varying arrival rates, service system

1. Introduction

Since service systems typically have arrival rates that vary strongly by time of day, e.g., see Figures 1, 8 and 9 in [1], there is interest in developing stochastic models with time-varying arrival rates. The most common arrival process model for this purpose is the nonhomogeneous Poisson process (NHPP), but there also is evidence from arrival data that an appropriate arrival process model with time-varying rates often should not be an NHPP; see [2, 3, 4, 5, 6, 7] and references therein.

Thus we want to create non-Poisson nonstationary arrival process models and study the performance of associated queueing models with those arrival

[1]Industrial Engineering and Operations Research, Columbia University, Email: nm2692@columbia.edu

[2]Industrial Engineering and Operations Research, Columbia University, Email: ww2040@columbia.edu, Correspondence to: MailCode 4704, S. W. Mudd Building, 500 West 120th Street, New York NY 10027-6699, U.S.A.

processes. Recent work developing non-Poisson nonstationary arrival process models and staffing algorithms to stabilize the performance in an associated queueing model with that arrival process is contained in [8, 9]. An important part of that work was conducting simulation experiments to evaluate how successful the proposed algorithms are in stabilizing performance at the designated targets. The purpose of the present note is to communicate how these simulations can be efficiently conducted.

In this line of research, it has been accepted practice to use stylized arrival rate functions that capture essential features of arrival rate functions that can be estimated from data. In particular, it has been standard to use the sinusoidal arrival rate function

$$\lambda(t) \equiv \lambda(t; \bar{\lambda}, \beta, \gamma) \equiv \bar{\lambda}(1 + \beta \sin(\gamma t)) \quad \text{for} \quad 0 < \beta < 1 \quad \text{and} \quad \gamma > 0, \quad (1)$$

where $\bar{\lambda}$ is the average arrival rate (the spatial scale), β is the relative amplitude and γ is the time scaling factor, determining the associated cycle length $T = 2\pi/\gamma$. (We also typically assume that the mean service time is 1 in the queueing model, which just fixes the time units.)

To understand how a staffing algorithm performs, we need to consider a range of the parameters for spatial scale $\bar{\lambda}$ and temporal scale γ in (1). (These same parameters can be part of any periodic arrival rate function.) Moreover, we need to consider a range of nonstationary arrival processes with such an arrival rate function. Here we show that these requirements can be achieved with remarkable efficiency with an appropriate approach.

We first note that extending the arrival process beyond an NHPP presents a challenge for the simulation. The standard way to simulate an NHPP is to apply time-dependent thinning of a homogeneous Poisson process (PP), as in §2.4 of [10] and [11]. Thinning can be applied to a large class of renewal processes using hazard rate stochastic ordering, as in §9.3 of [10] and §2.3 of [12], but extension to non-renewal processes requires additional work.

Our main idea for simulating non-Poisson nonstationary arrival processes is to exploit the inverse method, as often used in generating non-uniform random numbers; see §II.2 and §III.2 of [13] and §3.8 of [14]. The inverse method can be used for NHPP's, but it is even more appealing here because it allows us to efficiently simulate a large class of non-Poisson nonstationary arrival processes, not just one.

The first step in this approach is to construct a large class of non-Poisson nonstationary arrival process models by using the inverse Λ^{-1} of the cumulative arrival rate function Λ, which for the sinusoidal arrival rate function

in (1) is

$$\Lambda(t) \equiv \int_0^t \lambda(s)\, ds = \bar{\lambda}[t + (\beta/\gamma)(1 - \cos(\gamma t))], \quad t \geq 0. \tag{2}$$

The associated inverse function Λ^{-1} is well defined for (2) and any arrival rate function for which

$$0 < \lambda_L \leq \lambda(t) \leq \lambda_U < \infty \quad \text{for all} \quad 0 \leq t \leq T < \infty; \tag{3}$$

e.g., we could apply basic properties of inverse functions, as in §13.6 of [15].

Exploiting a well known representation, as in §7 in [16] and [8, 12, 17, 18], we define a nonstationary counting process A for any cumulative arrival rate function Λ (such as in (2)) and any rate-1 counting process N that we are able to simulate by letting

$$A(t) \equiv N(\Lambda(t)), \quad t \geq 0. \tag{4}$$

It is immediate that $E[A(t)] = \Lambda(t)$, $t \geq 0$, and the arrival times of A and N, denoted by A_k and N_k respectively, are related by

$$A_k = \Lambda^{-1}(N_k), \quad k \geq 1. \tag{5}$$

Since the inverse function Λ^{-1} is often unavailable explicitly, we construct a suitably accurate approximation of it and apply it by table lookup. In §2 we explain how the possibility of re-use provides remarkable efficiency; in §3 we develop an algorithm to efficiently construct the approximate inverse function with specified accuracy; and in §4 we discuss additional application issues.

2. The Basis for Efficiency Through Re-Use

A main advantage of the inverse function approach is the possibility of re-use. Since the inverse function satisfies a fixed point equation, an alternative way to calculate the inverse is to solve the fixed point equation for each arrival time, perhaps by search, exploiting the monotonicity. That is done in [19]. However, that search has to be performed at each arrival time. The search has the advantage that there should usually be far fewer arrivals in a fixed interval $[0, T]$ than arguments in a tabled inverse function, but the inverse function has the advantage that it can be constructed once outside the simulation and re-used. Moreover, the calculation from the table can be very fast, because it is possible to proceed forward through the table only once.

2.1. One Cycle for Periodic Arrival Rate Functions

The algorithm can be accelerated if the arrival rate function is periodic, as for the sinusoidal function in (1), because it suffices to calculate the inverse only for a single cycle. For example, with the sinusoidal arrival rate function in (1), $\Lambda(2k\pi/\gamma) = \bar{\lambda}2k\pi/\gamma$ for all integers $k \geq 0$, so that $\Lambda^{-1}(2k\bar{\lambda}\pi/\gamma) = 2k\pi/\gamma$ for all integers $k \geq 0$. Hence, it suffices to construct the inverse for $0 \leq t < 2\pi/\gamma$. Overall, we get

$$\Lambda^{-1}((2k\bar{\lambda}\pi/\gamma) + t) \;\; = \;\; (2k\pi/\gamma) + \Lambda^{-1}(t), \quad 0 \leq t \leq 2\bar{\lambda}\pi/\gamma, \tag{6}$$

so that it suffices to calculate Λ^{-1} on the interval $[0, 2\bar{\lambda}\pi/\gamma]$.

2.2. Different Scaling of Time and Space

We also can use one constructed inverse function Λ^{-1} to obtain inverse functions for scaled versions of the original function Λ. This commonly occurs with sinusoidal arrival rate functions $\lambda(t; \bar{\lambda}, \beta, \gamma)$ in (1). We are often interested in different spatial and temporal scale parameters $\bar{\lambda}$ and γ. Since

$$\Lambda(t; \bar{\lambda}, \beta, \gamma) = \bar{\lambda}\Lambda(\gamma t; 1, \beta, 1)/\gamma, \tag{7}$$

we can apply Lemma 13.6.6 of [15] to express the inverse as

$$\Lambda^{-1}(t; \bar{\lambda}, \beta, \gamma) = \Lambda^{-1}(\gamma t/\bar{\lambda}; 1, \beta, 1)/\gamma. \tag{8}$$

Hence, we can use the constructed inverse function $\Lambda^{-1}(t; 1, \beta, 1)$ for $\Lambda(t; 1, \beta, 1)$ to construct the inverse function $\Lambda^{-1}(t; \bar{\lambda}, \beta, \gamma)$ for $\Lambda(t; \bar{\lambda}, \beta, \gamma)$; i.e., we can reduce the three parameters to just one.

2.3. Multiple Non-Poisson Nonstationary Arrival Process Models

In order to evaluate performance approximations and system controls such as staffing algorithms, we need to consider a variety of models to ensure that the methods are successful for a large class of models. It is thus significant that a constructed inverse function Λ^{-1} can be re-used with different rate-1 stochastic counting processes N. For any rate-1 counting process N that we can simulate, we can generate the corresponding nonstationary arrival process with the same arrival rate function λ simply by applying the tabled inverse function to the arrival times of that rate-1 process, as in (5). Methods for simulating stationary counting processes are well established.

2.4. Multiple Replications to Obtain Accurate Performance Estimates

The tabled inverse function can be re-used in each replication when many replications are performed to obtain accurate performance estimates. For example, we might use 10^4 or more i.i.d. replications.

3. Constructing the Approximation of the Inverse Function

By (5), if we can simulate the arrival times N_k of the designated rate-1 process, then to simulate the desired arrival times A_k of the nonstationary point process A, it only remains to compute $\Lambda^{-1}(N_k)$ for each k. This is straightforward if the inverse function is available explicitly. If we use data to estimate the cumulative arrival rate function, then we can fit a convenient invertible function Λ. Indeed, with data there seems to be no reason not to use an invertible function. For example, it could be a piecewise-linear function as in [12, 18, 20, 21].

However, starting with an explicit non-invertible function Λ, as in (2), we want to efficiently construct an approximation of Λ^{-1} that is (i) easy to implement, (ii) fast in its implication and has (iii) suitably small specified accuracy. We could act just as if we had data, and fit a convenient invertible function, but then it remains to substantiate that the three goals have been met. To achieve these three goals, we contend that a good approach is to construct a piecewise-constant approximation. Of course, this construction can yield multiple points when that is not possible in the counting process A, but that is easily eliminated if it is deemed important; see §4.4. At some extra work, we could convert the piecewise-constant approximation to a piecewise-linear approximation, paralleling [20]. For all these modifications, our error bound still applies. For the queueing applications, this last refinement step should usually not be necessary.

We assume that a cumulative arrival rate function Λ associated with an arrival rate function λ satisfying (3) is given over a finite interval $[0, T]$. By (3), there exists a function r such that $\Lambda^{-1}(t) = \int_0^t r(s)\, ds$, $0 \le t \le \Lambda(T)$, and

$$0 < 1/\lambda_U \le r(t) \le 1/\lambda_L < \infty, \quad 0 \le t \le \Lambda(T). \tag{9}$$

Our goal is to efficiently construct an approximation J to the inverse function Λ^{-1} mapping the interval $[0, \Lambda(T)]$ into $[0, T]$ with specified accuracy

$$\|J - \Lambda^{-1}\| \equiv \sup_{0 \le t \le \Lambda(T)} \{|J(t) - \Lambda^{-1}(t)|\} \le \epsilon \tag{10}$$

for some suitably small target $\epsilon > 0$. This is a natural way to quantify the error, because ϵ specified the maximum error in the arrival times.

Our general strategy is to partition the two intervals $[0, T]$ and $[0, \Lambda(T)]$ into n_x and n_y evenly spaced subintervals of width η and δ, respectively, and then define J at $i\delta$ to be an appropriate $j\eta$, for each i, $0 \le i \le n_y$. We extend J to $[0, \Lambda(T)]$ by making J a right-continuous step function, assuming these constant values specified at $i\delta$.

Key parameters for our algorithm are

$$\rho \equiv \rho_\Lambda \equiv \frac{\lambda_U}{\lambda_L}, \quad \eta = \frac{\epsilon}{1+\rho} \quad \text{and} \quad \delta = \lambda_U \eta = \frac{\lambda_U \epsilon}{1+\rho}, \tag{11}$$

where λ_L and λ_U are the lower and upper bounds on the arrival rate function λ given in (3) and ϵ the desired error bound in (10). Thus ρ is the slope ratio with $1 \le \rho < \infty$, while δ and η are spacings used to achieve the target error bound ϵ in (10).

To construct J, we first calculate $\Lambda(x)$ for each of the $n_x + 1$ points x in $[0, T]$ by letting

$$a(j) \equiv \Lambda(k\eta),\ 0 \le j \le n_x. \tag{12}$$

Then we approximate the $\Lambda^{-1}(y)$ value of each of the $n_y + 1$ points y in $[0, \Lambda(T)]$ by a suitable point within the n_x points in $[0, T]$, i.e.,

$$b(i) \equiv \inf\{j \ge 0 : a(j) \ge i\delta\}, \quad 0 \le i \le n_y. \tag{13}$$

Then $J(i\delta) = b(i)\eta$ for all i, $0 \le i \le n_y$. The simple vector representations in (12) and (13) are the basis for the implementation efficiency.

We can finally get the value of J at any time y in $[0, \Lambda(T)]$ by

$$J(y) = J(\lfloor y/\delta \rfloor \delta),\ 0 \le y \le \Lambda(T), \tag{14}$$

where $\lfloor y \rfloor$ is the floor function, yielding the greatest integer less than or equal to y. However, this extension is not used directly because we start by changing N_k to $\lfloor N_k/\delta \rfloor \delta$, so we only use J defined on the finite subset $\{i\delta : 0 \le i \le n_y\}$. The function J is constructed to be one-to-one on the finite subset $\{i\delta : 0 \le i \le n_y\}$.

Theorem 3.1. *(error bound and computational complexity) Algorithm 1 above constructs a nondecreasing function J on $[0, \Lambda(T)]$ approximating Λ^{-1} with the error upper bound ϵ prescribed in* (10) *using of order $O(n_x + n_y) = O(2T(1+\rho)/\epsilon)$ storage (two vectors each of size n_x and n_y) with computational complexity of order $O(n_x + n_y) = O(2T(1+\rho)/\epsilon)$.*

Algorithm 1 Constructing the approximation J of the inverse function Λ^{-1} for given time T, function $\Lambda : [0, T] \to [0, \Lambda(T)]$ and error bound ϵ

Set $\rho \leftarrow \lambda_U/\lambda_L$, $\eta \leftarrow \epsilon/(1+\rho)$, $\delta \leftarrow \lambda_U\epsilon/(1+\rho)$, $n_x \leftarrow T/\epsilon$, $n_y \leftarrow \Lambda(T)/\delta$ // *(five constant parameters)*
Set $x \leftarrow (0 : \eta : T), y \leftarrow (0 : \delta : \Lambda(T))$ *//(two equally spaced vectors of length* $n_x + 1$ *and* $n_y + 1$*)*
Set $a \leftarrow \Lambda(x), b \leftarrow []$ *//(two new vectors of length* $n_x + 1$ *and* $n_y + 1$ *with b zero vector)*
Set $i \leftarrow 1, j \leftarrow 1$ *//(initialize for* $n_x + n_y$ *operations)*
While $j < n_x + 1 \quad \&\& \quad i < n_y + 1$ do
    If $y(i) > a(j)$ Then
        $j \leftarrow j + 1$
    Else
        $b(i) \leftarrow j, \quad i \leftarrow i + 1$
    End if
End While
//(For $0 \le i \le n_y$*,* $J(i\delta) = b(i)\eta$*; J extended to* $[0, \Lambda(T)]$ *by right-continuity.)*

Proof. For any $\delta > 0$ and $\eta > 0$, a bound on the error in J is

$$\begin{aligned}
\|J - \Lambda^{-1}\| &\equiv \sup_{0\le t\le\Lambda(T)} |J(t) - \Lambda^{-1}(t)| = \sup_{0\le i\le n_y} \sup_{t\in[i\delta,(i+1)\delta)} |J(i\delta) - \Lambda^{-1}(t)| \\
&= \sup_{0\le i\le n_y} \sup_{t\in[i\delta,(i+1)\delta)} |b(i)\eta - \Lambda^{-1}(i\delta) + \Lambda^{-1}(i\delta) - \Lambda^{-1}(t)| \\
&\le \sup_{0\le i\le n_y} (|b(i)\eta - \Lambda^{-1}(i\delta)| + |\Lambda^{-1}(i\delta) - \Lambda^{-1}((i+1)\delta)|) \\
&\le \eta + \delta/\lambda_L, \qquad (15)
\end{aligned}$$

where the fourth line follows because the point $\Lambda^{-1}(i\delta)$ lies in the interval $(b(i)\eta, b(i+1)\eta]$.

Next observe that the function J will be one-to-one (have distinct values) on the set $\{i\delta : 0 \le i \le n_y\}$ if $\delta \ge \lambda_U\eta$. Now we choose δ such that

$$\delta = \lambda_U\eta. \qquad (16)$$

Then J is one-to-one on $\{i\delta : 0 \le i \le n_y\}$ and, by (15) and (16),

$$\|J - \Lambda^{-1}\| \le \eta + \delta/\lambda_L \le \frac{\epsilon}{1+\rho} + \frac{\rho\epsilon}{1+\rho} = \epsilon. \qquad (17)$$

Turning to the computational complexity, we see that four vectors need to be stored: x, y, a and b, which is of total length $2(n_x + n_y + 2)$. To construct the table of J, the while loop in algorithm 1 searches for $b(i)$ for each $0 \le i \le n_y$, which checks each of the $(n_x + n_y)$ points only once and takes time $O(n_x + n_y)$. Finally, by (11) again,

$$n_x + n_y = \frac{T}{\delta} + \frac{\Lambda(T)}{\eta} = \frac{T(1+\rho)}{\epsilon} + \frac{\Lambda(T)(1+\rho)}{\lambda_U \epsilon} \le \frac{2T(1+\rho)}{\epsilon}. \quad \blacksquare \qquad (18)$$

4. Application Issues

4.1. Generating the Arrival Times

Given Algorithm 1, the algorithm to construct the actual arrival times $A_k = \Lambda^{-1}(N_k)$ given all the rate-1 arrival times N_k can be very simple. If we apply the floor function and the inverse function in Algorithm 1 in a single vector operation to all components of the vector of rate-1 arrival times, then the code can be expressed in a single line.

Algorithm 2 constructing the vector $A \equiv \{A_k\}$ of arrival times in $[0, T]$ given Algorithm 1 specified in terms of the triple (δ, η, b) depending on the error bound ϵ in (10) and the associated nondecreasing vector of nonnegative rate-1 arrival times $N \equiv \{N_k : 1 \le k \le n\}$ with $N_n \le \Lambda(T)$

1: Set $A \leftarrow b(\lfloor N/\delta \rfloor)\eta$ // *(vector application of the floor function and Algorithm 1 term by term)*

In the single line of Algorithm 2 we have used (14) and line 12 of Algorithm 1, i.e.,

$$J(\lfloor t/\delta \rfloor \delta) = b(\lfloor t/\delta \rfloor)\eta \quad \text{or} \quad J(i\delta) = b(i)\eta, \quad 0 \le i \le n_y. \qquad (19)$$

This is important for implementation efficiency, because we make only one pass through the table to generate all the arrival times A_k.

4.2. Partitioning Into Subintervals

For difficult arrival rate functions, it might be preferable to modify the representation of the inverse function, e.g., moving closer to a piecewise-linear approximation. In particular, if the slope ratio ρ in (11) is large, then it may be easy to accelerate the algorithm by dividing the original interval $[0, T]$

into subintervals. A simple example is a piecewise linear function with two pieces, one having a flat slope and the other having a steep slope, so that the ratio ρ might be very large. If we divide the interval into the two parts where Λ is linear, then ρ is reduced to 1 on each subinterval. Given that we divide $[0, T]$ into the two intervals $[0, T_1]$ and $[T_1, T]$, we can calculate Λ^{-1} separately on the two intervals $[0, \Lambda(T_1)]$ and $[\Lambda(T_1), \Lambda(T)]$.

4.3. Choosing the Error Bound

It is natural to ask how the error bound ϵ should be chosen in practice. We think it should usually be possible to choose ϵ relatively small compared to an expected interarrival time of A, which has a time-varying value exceeding $1/\lambda_U$ for λ_U in (3). However, for queueing applications that might be smaller than necessary, because the relevant time scale in a queueing system is typically of order equal to a mean service time, which depends on the units used to measure time. Suppose, without loss of generality, we choose the time units so that the mean service time is 1. Then we think it usually should suffice to let ϵ be small compared to the maximum of these, e.g., $\epsilon \approx \max\{1, 1/\lambda_U\}/100$.

To illustrate, consider an example of a moderately large call center in which the mean service time is about 5 minutes, while the arrival rate is 600 per hour or 1/6 per second, as in §3.1 of [5], which makes $\lambda_U = 600/12 = 50$ in units of mean service times. The rough guideline above yields $\epsilon \approx \max\{1, 0.02\}/100 = 0.01$ mean service times or $300/100 = 3$ seconds, which seems reasonable.

Assuming that time is measured in mean service times and $\lambda_U \geq 1$ in that scale, the computational complexity from Theorem 3.1 becomes $2T(1+\rho) \times 10^2$. In the call center example, if we let $T = 24 \times 12 = 288$ corresponding to one 24-hour day measured in units of 5 minute-calls, then the computational complexity of the algorithm to calculate the inverse function is $57,600(1+\rho)$.

4.4. Breaking Ties: Ensuring an Orderly Point Process

We have constructed the approximate inverse function J to be one-to-one in the finite subset $\{i\delta : 0 \leq i \leq n_y\}$. However, that does not prevent multiple points in A, because all points from the rate-1 process N in the interval $[i\delta, (i+1)\delta)$ are mapped into the same point $b(i)\eta$, for each i, $0 \leq i \leq n_y - 1$.

First, we can easily identify multiple points by looking for the zeros in the vector B, where $B_k \equiv A_k - A_{k-1}$. Then we can easily remove them if we want. Suppose that $A_{k-1} < A_k = A_{k+j} < A_{k+j+1}$ for some $k \geq 1$ and $j \geq 1$. Then

replace A_{k+i} by $A_k + i\epsilon/(j+1)$, $1 \le i \le j$. We could further randomize by using $A_k + (i + U_{k+i})\epsilon/(j+1)+$, $1 \le i \le j$, where $\{U_k : k \ge 1\}$ is a sequence of i.i.d. uniform random variables on $[0, 1]$. However, these adjustments should not be required for queueing applications if we are satisfied with the "measurement error" of ϵ, as discussed in §4.3.

4.5. Selecting the Rate-One Stochastic Process N

In applications, a key remaining problem is actually identifying an appropriate non-Poisson nonstationary arrival process. Assuming that ample data are available to estimate the cumulative arrival rate function, the question about choosing A is roughly equivalent to the question about choosing the rate-1 process N for given cumulative arrival rate function Λ.

As discussed in [8, 16], it is natural to specify the functional central limit theorem behavior of N, by the asymptotic index of dispersion for the arrival process A, i.e., we use measurements of A to estimate

$$c_A^2 \equiv \lim_{t\to\infty} \frac{Var(A(t))}{E[A(t)]} = \lim_{t\to\infty} \frac{Var(N(t))}{E[N(t)]}. \tag{20}$$

It is then easy to choose stationary renewal processes N with this c_A^2 [22]. However, while this should yield an appropriate c_A^2, this does not nearly specify the processes N and A fully. However, heavy-traffic limit theorems indicate that this may be sufficient; see §4 of [8].

4.6. Random-Rate Arrival Processes

As discussed in [23], [6] and references therein, it may be desirable to represent the arrival rate over each day as random. For example, the model of the arrival process on one day of length T might be

$$A(t) = N(X\Lambda(t)), \quad 0 \le t \le T, \tag{21}$$

where N is a rate-1 stochastic processes, perhaps Poisson, while Λ is a deterministic cumulative arrival rate function and X is a positive random variable. The overall cumulative arrival rate of A is

$$E[A(t)] = E[N(X\Lambda(t))] = E[X]\Lambda(t), \quad 0 \le t \le T. \tag{22}$$

With this structure, we can exploit the scaling properties in §2.2 to accelerate simulations. In particular, the representation (22) can be viewed as a

variant of our model in which the cumulative arrival rate function is the random function $\tilde{\Lambda}(t) \equiv X\Lambda(t)$. Fortunately, the inverse of $\tilde{\Lambda}$ can be expressed directly in terms of the inverse Λ^{-1} and the random variable X by

$$\tilde{\Lambda}^{-1}(t) = \Lambda^{-1}(t/X), \quad 0 \le t \le X\Lambda(T) \tag{23}$$

For any single realization of the random variable X above, we can simulate the stochastic process A in the manner described in previous sections. However, to assess the system performance, we would need to consider the values of X over successive days, but these random variables X_k over successive days k are likely to be dependent with distributions depending on the day of the week and the week of the year. Nevertheless, the inverse in (23) can be efficiently calculated for each of these these days using the single inverse function Λ^{-1}. However, by sampling sufficiently many days, we may capture the impact of this random variable X.

Acknowledgement. Support was received from NSF grant CMMI 1265070.

References

[1] L. V. Green, P. J. Kolesar, W. Whitt, Coping with time-varying demand when setting staffing requirements for a service system, Production Oper. Management 16 (2007) 13–29.

[2] A. N. Avramidis, A. Deslauriers, P. L'Ecuyer, Modeling daily arrivals to a telephone call center, Management Sci. 50 (2004) 896–908.

[3] R. Ibrahim, P. L'Ecuyer, N. Regnard, H. Shen, On the modeling and forecasting of call center arrivals, Proceedings of the 2012 Winter Simulation Conference 2012 (2012) 256–267.

[4] G. Jongbloed, G. Koole, Managing uncertainty in call centers using Poisson mixtures, Applied Stochastic Models in Business and Industry 17 (2001) 307–318.

[5] S.-H. Kim, W. Whitt, Are call center and hospital arrivals well modeled by nonhomogeneous Poisson processes?, Manufacturing and Service Oper. Management 16 (3) (2014) 464–480.

[6] S.-H. Kim, P. Vel, W. Whitt, W. C. Cha, Poisson and non-Poisson properties of appointment-generated arrival processes: The case of an endocrinology clinic, Operations Research Letters 43 (2015) 247–253.

[7] X. Zhang, L. J. Hong, P. W. Glynn, Timescales in modeling call center arrivals, working paper, Department of Industrial Engineering and Logistics Management, The Hong Kong University of Science and Technology (2014).

[8] B. He, Y. Liu, W. Whitt, Staffing a service system with non-Poisson nonstationary arrivals, working paper, Department of Industrial and Systems Engineering, North Carolina State University (2015).

[9] W. Whitt, Stabilizing performance in a single-server queue with time-varying arrival rate, Columbia University, http://www.columbia.edu/∼ww2040/allpapers.html (2014).

[10] S. M. Ross, Stochastic Processes, 2nd Edition, Wiley, New York, 1996.

[11] P. A. W. Lewis, G. S. Shedler, Simulation of nonhomogeneous Poisson processes by thinning, Naval Research Logistics Quarterly 26 (1) (1979) 403–413.

[12] I. Gerhardt, B. L. Nelson, Transforming renewal processes for simulation of nonstationary arrival processes, INFORMS Journal on Computing 21 (2009) 630–640.

[13] L. Devroye, Non-Uniform Random Variate Generation, Sprnger, New York, 1986.

[14] P. L'Ecuyer, Random number generation, in: J. E. Gentle, J. K. Hardle, Y. Mori (Eds.), Handbook of Computational Statistics, Springer, New York, 2012, Ch. 3, pp. 35–71.

[15] W. Whitt, Stochastic-Process Limits, Springer, New York, 2002.

[16] W. A. Massey, W. Whitt, Unstable asymptotics for nonstationary queues., Math. Oper. Res. 19 (1994) 267–291.

[17] R. Liu, M. E. Kuhl, Y. Liu, J. R. Wilson, Modeling and simulation of nonstationary non-Poisson arrival processes, working paper, North Carolina State University, Raleigh, NC (2014).

[18] B. L. Nelson, I. Gerhardt, Modeling and simulating renewal nonstationary arrival processes to facilitate analysis, Journal of Simulation 5 (2011) 3–8.

[19] H. Chen, B. W. Schmeiser, Simulation of Poisson processes with trigonometric rate, in: Proceedings of the 1992 Winter Simulation Conference, ACM, 1992, pp. 609–617.

[20] L. M. Leemis, Nonparametric estimation of the cumulative intensity function for a nonhomogeneous poisson process, Management Science 37 (1991) 886–900.

[21] W. A. Massey, G. A. Parker, W. Whitt, Estimating the parameters of a nonhomogeneous Poisson process with linear rate, Telecommunication Systems 5 (1996) 361–388.

[22] W. Whitt, Approximating a point process by a renewal process: two basic methods, Oper. Res. 30 (1982) 125–147.

[23] W. Whitt, Dynamic staffing in a telephone call center aiming to immediately answer all calls, Operations Research Letters 24 (1999) 205–212.