Discrete Event Dynamic Systems: Theory and Applications, , 1–25 (Draft: June 19, 1995)

Functional Estimation with Respect to a Threshold Parameter via Dynamic Split-and-Merge

PIERRE L'ECUYER lecuyer@iro.umontreal.ca
FELISA J. VÁZQUEZ-ABAD vazquez@iro.umontreal.ca
Département d'IRO, Université de Montréal,
C.P. 6128, Succ. Centre-Ville, Montréal H3C 3J7, Canada

Abstract. We consider a class of stochastic models for which the performance measure is defined as a mathematical expectation that depends on a parameter θ, say $\alpha(\theta)$, and we are interested in constructing estimators of α in functional form (i.e., entire functions of θ), which can be computed from a single simulation experiment. We focus on the case where θ is a continuous parameter, and also consider estimation of the derivative $\alpha'(\theta)$. One approach for doing that, when θ is a parameter of the probability law that governs the system, is based on the use of likelihood ratios and score functions. In this paper, we study a different approach, called *split-and-merge*, for the case where θ is a threshold parameter. This approach can be viewed as a practical way of running parallel simulations at an infinite number of values of θ, with common random numbers. We give several examples showing how different kinds of parameters such as the arrival rate in a queue, the probability that an arriving customer be of a given type, a scale parameter of a service time distribution, and so on, can be turned into threshold parameters. We also discuss implementation issues.

Keywords: Simulation, functional estimation

1. Functional Estimation

Let $\{(\Omega, \Sigma, P_\theta),\ \theta \in \Theta\}$ be a family of probability spaces defined over the same measurable space, where $\Theta = [a, b]$ is a bounded interval of the real line. In general, the probability law P_θ may depend on a parameter θ. Consider a finite-horizon discrete event model defined over that family of probability spaces and let $h(\theta, \omega)$ be some random variable of interest (e.g. total sojourn time of all the customers served during a given day in a queueing system, or the total number of rejected customers in a finite-buffer system, etc). Suppose that we are interested in the function

$$\alpha(\theta) = E_\theta[h(\theta, \omega)] = \int_\Omega h(\theta, \omega) P_\theta(d\omega).$$

Normally, a simulation performed at $\theta = \theta_0$ permits one to estimate $\alpha(\theta_0)$ and perhaps $\alpha'(\theta_0)$ or higher order derivatives. Techniques for doing that include the likelihood ratio or score function method, as well as perturbation analysis and its numerous variants (see [2], [5], [9], [12], [14], [15], [18] and the several references given there). To obtain estimations at different values of θ, one would usually perform different simulations (perhaps with common random numbers) at all of those values of interest, which may become costly.

One approach for estimating α in a functional form, i.e., for estimating $\alpha(\theta)$ for all $\theta \in \Theta$ from a single simulation run, is based on the "change of measure" idea, sometimes called *importance sampling* [10]. To summarize the idea in a simplified form, suppose that $h(\theta,\omega) = h(\omega)$ does not depend (directly) on θ and that P_θ has a corresponding density f_θ. Then, assuming that the *likelihood ratio* $L(\theta_0, \theta, \omega) = (f_\theta/f_{\theta_0})(\omega)$ exists, α can be rewritten as

$$\alpha(\theta) = \int_\Omega h(\omega) f_\theta(\omega) d\omega = \int_\Omega [h(\omega) L(\theta_0, \theta, \omega)] f_{\theta_0}(\omega) d\omega.$$

So, if the simulation is performed at θ_0, then $h(\omega)L(\theta_0, \cdot, \omega)$ provides an unbiased functional estimator of α. The random variable $h(\omega)$ is computed only once, and the likelihood ratio is calculated at any value of θ of interest to compute the estimator. Under appropriate regularity conditions, an unbiased estimator of the derivative $\alpha'(\theta)$ is $h(\omega)S(\theta,\omega)L(\theta_0,\theta,\omega)$, where $S(\theta,\omega) = \frac{\partial}{\partial\theta} \ln L(\theta_0,\theta,\omega)$ is called the *score function*.

Several examples of this approach, with numerical illustrations for some queueing systems, are given in [18], [16]. Some of these examples show that the likelihood ratio approach sometimes works fine, while others show how dramatically the variance of the functional estimators for α and α' may increase (sometimes with vertical asymptotes) as θ gets away from θ_0. The variance also increases exponentially fast (typically) as a function of the length of the simulation time-horizon (see [18], [16]). As a result, those functional estimators are often useful only in a small neighborhood of θ_0 and are better suited for short horizon models, or "steady-state" models with short regenerative cycles. Moreover, they can be defined only if θ can be interpreted as a parameter of the probability law P_θ.

In this paper, we examine an alternative *split-and-merge* method for estimating α (or α') in functional form, for the following situation. For the remainder of the paper, we shall assume that the probability law $P_\theta \equiv P$ does not depend on θ (only h does). The sample point ω can be viewed for example as a sequence of i.i.d. $U(0,1)$ random variables that drive the simulation. The parameter θ is a *threshold* parameter that determines a sequence of *binary decisions* to be made during the simulation, as we now describe. Each time one of those decisions has to be made, the value of a specific random variable $Z_i = Z_i(\theta,\omega)$ that is part of the system's state is compared with θ; if it is larger, then the decision is 1, otherwise the decision is 0. The random variable Z_i can be a function of both the sample point ω and the parameter θ. This covers a rich variety of situations, as our examples will show. In particular, the possibility that θ be a parameter of a probability distribution is not ruled out; it can be dealt with indirectly. For example, θ could be the parameter of a Bernoulli distribution from which is generated a sequence of i.i.d. random variables during the simulation (to generate a Bernoulli, one generates a uniform between 0 and 1, and compares that uniform with the "threshold" θ).

The *split-and-merge* method is based on "branching" (or splitting) the simulation every time a decision has to be made and differs, depending on θ, within the range of parameter values of interest. A tree of simulations is thus constructed. Branches of the tree could also "merge" eventually. The nodes in the tree correspond to the simulation of a particular system and keep track of the system's state variables and relevant estimators. At the time when a decision has to be made in the simulation, the value of the variable Z_i at the corresponding node determines a threshold: for all θ larger (or smaller) than that value, the current decision has value 1 (or 0). At this point, the node can split into two nodes corresponding to different intervals of θ. Up to that point, those two nodes share the same history. All nodes in the tree correspond to a simulation based on a single sample point ω, but each node has a different sequence of decisions, corresponding to a particular subinterval of Θ. If the state of the system (including the future event list, and so on) eventually becomes the same for different such (neighboring) subintervals, then we say that *coupling* has occured between the corresponding trajectories. This can be exploited to reduce the growth of the simulation tree, by merging the corresponding parallel simulations. Hence the name *split-and-merge*.

Strictly speaking, the estimators are not really computed in a "single" simulation run, because the simulation run will be "split" along the way into several parallel runs. This approach can be viewed in fact as an implementation of parallel simulations for an infinite number of values of θ, with common random numbers. From the simulations, we can recover all the information required to compute what would happen at all values of θ in $[a, b]$. We exploit the fact that with our choice of probability space, the sample path changes as a function of θ only at a finite number of points. These points correspond to the nodes that are created in the tree. In general, the growth of the tree could be exponential in the worst case. However, we show that, under some assumptions, the number of discontinuity points increases only linearly with the number of binary decisions that are made. At the end of the simulation, the tree contains as many nodes as the number of subintervals of Θ that can be distinguished for the particular realization of ω generated by the simulation. Therefore, we can reconstruct the value of the piecewise constant estimator in the whole interval of interest.

The variance of the split-and-merge approach behaves much differently than that of the likelihood ratio method. In fact, the mean and the variance of the functional estimator at any given value of θ are exactly the same as if ordinary simulation runs were performed at that value of θ only. Therefore, the variance depends only on which underlying estimator is used, and is not influenced by the split-and-merge. This also applies to derivative estimators: any estimator that satisfies our assumptions (see next section) can be used. See [2], [5], [12], [14], [15], [18], [20] for more on derivative estmation. Furthermore, most variance reduction techniques that could be used when simulating at a single parameter value (such as importance sampling, control variates, antithetic variates, and so on), could be used with the split-and-merge approach as well, and will have just the same variance reduction effect at each θ.

The split-and-merge is strongly related to the *phantom method*. From a nominal simulation, the phantom method introduced in [19] computes one perturbed sample path if a particular decision was reversed. Some of the decisions would be reversed from the nominal simulation due to a small change in θ. In [24] the method was implemented to estimate a finite difference. The phantom method is based on a thought experiment, evaluating only that part of the perturbed system that differs from the nominal system under common random variables. It is generally more efficient than simulating in parallel the two systems with the same driving noise ω. The phantom method has been used by [2], [4], [24] for derivative estimation and in [26] for parallel computation for a finite number of system parameters. In [25], which was in fact a short preliminary version of the current paper, we used the term *phantom method* for the (somewhat more general) split-and-merge approach described here.

Vakili [21] describes an approach to perform simultaneous simulations at a finite number of distinct parameter values, for a continuous-time Markov chain model (all random variables that determine event times are exponential). That approach is based on a uniformization of the chain, which permits one to use the same sequence of event times for all the simulations. In contrast with our approach, it produces an estimation only for a finite number of parameter values, which must be fixed in advance. On the other hand, it allows for different kinds of parameters than the one we consider. If a model fits both frameworks: all times to events are exponential, one parameter satisfies our assumptions and others do not, then one can easily combine the two approaches. Notice also that continuous-time Markov chains can most of the time (when the performance measure is additive) be simulated without generating event times at all: just use *discrete-time conversion*, which consists in replacing times between events and transition costs by their (conditional) expectations (see [6], [7], [8] and Example 4.8 of [14]).

We define our setup more precisely and describe the split-and-merge functional estimation method in Section 2. We also give an upper bound on the computational work required by the method, as a function of the simulation time-horizon, under specific (sufficient) assumptions. That work can increase exponentially fast in general. But we show that under a condition that holds in several real-life situations, the increase is only linear. In Section 3, we illustrate the approach with some examples, most of which are variants of an admission control model for a $GI/GI/1$ queue, where each customer is accepted with probability θ. Different kinds of performance measures are considered, including some defined by derivatives. Numerical results are given for two examples. The first one involves a finite-capacity queue with two classes of customers, where customers of class 2 are admitted with probability θ and various mathematical expectations are estimated as functions of θ. In the second one, we estimate the derivative of the expected number of customers per busy cycle, with respect to some parameter ν of the service time distribution, as a function of θ. In both cases, the numerical behavior of the method agrees with the results of Section 2 and is quite encouraging.

2. Simulation Trees for Functional Estimation

We consider a finite-horizon simulation, parameterized by $\theta \in \Theta = [a, b] \subseteq \mathbb{R}$, defined over a probability space (Ω, Σ, P). For each sample point ω, the evolution of the model depends on the parameter θ as well as on ω (but the distribution of ω does not depend on θ). Let $\mathcal{F}_t(\theta)$ denote the sigma-field generated by the model's evolution up to time t (which can be viewed as representing all the information available by observing all what happens up to time t, for a fixed $\theta \in \Theta$). We shall assume that the dependence on θ takes the following form. For each θ, there is a sequence of (perhaps random) times $0 \le T_1 \le T_2 \le \cdots$ and a sequence of real-valued random variables $Z_1, Z_2, \ldots$, where $T_i \equiv T_i(\theta, \omega)$ and $Z_i \equiv Z_i(\theta, \omega)$ are $\mathcal{F}_{T_i}(\theta)$-measurable for each i and θ. At time T_i, Z_i is compared with θ and a binary random variable (or decision) η_i is defined as:

$$\eta_i \equiv \eta_i(\theta, \omega) = I[Z_i(\theta, \omega) \le \theta], \tag{1}$$

where I denotes the indicator function. Define $\tilde{\eta}_{t,i} = \eta_i \cdot I[T_i \le t]$ and let $\tilde{\mathcal{F}}_t(\theta)$ be the sigma-field generated by ω and $(\tilde{\eta}_{t,1}, \tilde{\eta}_{t,2}, \ldots)$. We shall assume that $\tilde{\mathcal{F}}_t(\theta) = \mathcal{F}_t(\theta)$; i.e., that the model's evolution up to time t depends on θ only through the binary decisions $\eta_i(\theta, \omega)$ that have been made at or before time t (i.e., such that $T_i \le t$). This implies in particular that we can write

$$T_i = T_i(\theta, \omega) = \varphi_i^t(\eta_1(\theta, \omega), \ldots, \eta_{i-1}(\theta, \omega), \omega) \tag{2}$$

and

$$Z_i = Z_i(\theta, \omega) = \varphi_i^z(\eta_1(\theta, \omega), \ldots, \eta_{i-1}(\theta, \omega), \omega), \tag{3}$$

where each φ_i^z and φ_i^t are measurable real-valued functions defined on $\{0, 1\}^{i-1} \times \Omega$. So, conditional on the values of the binary random variables $(\eta_1, \ldots, \eta_{i-1})$, T_i and Z_i are independent of θ.

We also assume that with probability one (w.p.1), the model has a finite (generally random) time-horizon $T \equiv T(\theta, \omega)$ and that the number $\tau \equiv \tau(\theta, \omega)$ of binary decisions made by time $T(\theta, \omega)$ is finite, uniformly over θ. More specifically, we have:

Assumption 1 *For each $\theta \in [a, b]$, we have that*

(i) $\tilde{\mathcal{F}}_t(\theta) = \mathcal{F}_t(\theta)$.

(ii) There is a random variable $T \equiv T(\theta, \omega) \ge 0$ such that $T(\theta, \omega)$ and $h(\theta, \omega)$ are $\mathcal{F}_{T(\theta,\omega)}(\theta)$-measurable and

$$\sup_{\theta \in [a,b]} T(\theta, \omega) < \infty \quad w.p.1. \tag{4}$$

(iii) For each $i > 0$, $T_i(\theta, \omega)$ and $Z_i(\theta, \omega)$ are $\mathcal{F}_{T_i(\theta,\omega)}(\theta)$-measurable and $\eta_i(\theta, \omega)$ satisfies (1).

(iv) Define

$$\tau(\theta,\omega) = \sum_{i=1}^{\infty} I[T_i(\theta,\omega) \le T(\theta,\omega)].$$

Then,

$$\sup_{\theta\in[a,b]} \tau(\theta,\omega) < \infty \quad w.p.1. \tag{5}$$

These assumptions imply that $h(\theta,\omega)$ can be written as

$$h(\theta,\omega) = \varphi^h_{\tau(\theta,\omega)}(\eta_1(\theta,\omega),\ldots,\eta_{\tau(\theta,\omega)}(\theta,\omega),\omega), \tag{6}$$

where $\varphi^h_i : \{0,1\}^{i-1} \times \Omega \to \mathbb{R}$ for each i, and $\tau(\theta,\omega)$ represents the number of binary decisions made when (or before) the finite horizon $T(\theta,\omega)$ is reached. The latter is a stopping time with respect to $\{\mathcal{F}_t(\theta),\ t \ge 0\}$.

From our assumptions, Z_1 and T_1 are functions of ω only, then η_1 is a function of Z_1 and θ, then Z_2 and T_2 are functions of ω and η_1, then η_2 is a function of Z_2 and θ, and so on. We have used the notation $Z_i(\theta,\omega)$ instead of expressing Z_i explicitly as a function of $(\eta_1,\ldots,\eta_{i-1},\omega)$ as in the right side of (3), and similarly for T_i, because our ultimate interest is a functional estimator which is a function of θ. However, our assumptions about the form of the dependence on θ are key assumptions for the rest of our development. For the remainder of the paper, to simplify, we will remove ω from the notation of the random variables h, T, τ, T_i, and Z_i.

Consider a fixed sample point $\omega \in \Omega$. For each $i \ge 1$, let $\Psi_i = \{z_{i,1}, z_{i,2},\ldots\}$ denote the set of values of θ in $[a,b]$ such that $i \le \tau(\theta)$ and $Z_i(\theta) = \theta$. Let

$$\Psi = \cup_{i=1}^{\infty} \Psi_i.$$

PROPOSITION 1 *Under Assumption 1, we have that $0 \le |\Psi_i| \le 2^{i-1}$ for each ω. We also have that w.p.1, Ψ_i eventually becomes empty for large enough i, so Ψ is finite.*

Proof: For each i and sample point ω, since $Z_i(\theta) = \varphi^z_i(\eta_1(\theta),\ldots,\eta_{i-1}(\theta),\omega)$ may depend on θ only via $\eta_1(\theta),\ldots,\eta_{i-1}(\theta)$, $Z_i(\theta)$ can take at most 2^{i-1} different values when θ varies in $[a,b]$. Therefore, the equation $Z_i(\theta) = \theta$ has at most 2^{i-1} roots in $[a,b]$ and the first part follows. The second part is a direct consequence of (5) in Assumption 1. ■

Let τ' be the cardinality of Ψ, and let $v_1 \le \cdots \le v_{\tau'}$ denote the values contained in Ψ, sorted by increasing order. Let also $v_0 = a$ and $v_{\tau'+1} = b$. We can now state the following.

PROPOSITION 2 *Under Assumption 1, w.p.1, $h(\cdot)$ is piecewise constant over $[a,b]$, with at most $\tau'+1$ pieces, and possible jumps only at $v_1,\ldots,v_{\tau'}$.*

Proof: Note that the v_j's are the values of θ at which $\eta_i(\theta)$ switches from 1 to 0 for some i such that $T_i(\theta) \le T(\theta)$. Therefore, for each $j \in \{0, 1, \ldots, \tau'\}$ and each i such that $T_i(\theta) \le T(\theta)$ (i.e., such that $i \le \tau(\theta)$), $\eta_i(\theta)$ must remain constant as a function of θ for $\theta \in (v_j, v_{j+1})$. From Assumption 1 (i) and (ii), $T(\theta)$ and $h(\theta)$ must then be constant over each (v_j, v_{j+1}). ∎

We call the values $v_1, \ldots, v_{\tau'}$ the *breakpoints* of $h(\cdot)$. Proposition 2 can be used to estimate $\alpha(\theta)$ simultaneously for all $\theta \in [a, b]$ as follows. Perform the simulation, obtaining the $z_{i,k}$'s along the way (further details on that will follow). Sort the $z_{i,k}$'s dynamically while they are generated (e.g., using a heap, as suggested in Problem 1.9.7 of Bratley, Fox, and Schrage [1]). At the end of all the simulations, let I_j denote the interval $I_j = [v_j, v_{j+1})$, for $j = 0, \ldots, \tau'$. The value of $h(\theta)$ is constant (and the sample path is the same) within each of those intervals, but may differ considerably between the intervals. To obtain a functional estimator, one must generate the sample path and compute h within each interval. Roughly, this is computationally equivalent to performing τ' simulation runs in parallel. Of course, a large fraction of the required computations is often common to many intervals and does not have to be repeated. Think for example of the situation where $\eta_i(\theta)$ represents the decision of accepting customer i into a queueing system. Changing $\eta_i(\theta)$ then changes the evolution of the system, but many of the other random variables generated are the same for both values of $\eta_i(\theta)$. Furthermore, the sample paths associated with two neighboring intervals I_{j-1} and I_j are exactly the same up to the time when the $Z_i(\theta)$ that corresponds to v_j is generated, and potentially differ only from that time on. So, before that time, there is not need to duplicate the simulation in order to distinguish between those intervals.

The split-and-merge functional estimation can be implemented by constructing a tree of sample paths as follows. Run the simulation until time T_1 and observe Z_1 (up to then, things do not depend on θ). Then, we consider two possible situations: (i) the value $z_{1,1}$ of Z_1 is in $[a, b]$ and (ii) the value of Z_1 is outside $[a, b]$. For situation (i), split the sample path in two pieces: one for the case $\theta < z_{1,1}$ (with $\eta_1(\theta) = 0$) and one for the case $\theta \ge z_{1,1}$ (with $\eta_1(\theta) = 1$). Continue the simulation for each of the two pieces (with the same ω). There are now two simulations running in parallel, associated with the intervals $[a, z_{1,1})$ and $[z_{1,1}, b]$, respectively. Run each of those simulations up to time T_2 (the evolution and the value of T_2 may be different for each of those two intervals). In situation (ii), the value of Z_1 being outside $[a, b]$, no splitting occurs at time T_1: there will still be a single sample path, at least up to time T_2.

For each interval (i.e., each value of η_1), at time T_2, observe the value of Z_2 for that interval and repeat the above with Z_1 and $[a, b]$ replaced by Z_2 and the interval considered, respectively. For example, if splitting has occured at T_1 and if $Z_2 = z_{2,1} \in [a, z_{1,1})$ for $\eta_1 = 0$, then the interval $[a, z_{1,1})$ is split in two pieces: $[a, z_{2,1})$, for which $\eta_2 = 0$, and $[z_{2,1}, z_{1,1})$, for which $\eta_2 = 1$. The simulation must be continued separately for each of those two intervals, which now yields three sample paths evolving in parallel. If $Z_2 \notin [a, z_{1,1})$ for $\eta_1 = 0$, then no splitting occurs for that interval at time T_2. Similarly, if $Z_2 = z_{2,2} \in [z_{1,1}, b]$ for $\eta_1 = 1$, then $[z_{1,1}, b]$ is split into $[z_{1,1}, z_{2,2})$ and $[z_{2,2}, b]$ at time T_2, otherwise it is not. Therefore, after time T_2 (where T_2 may take a different value for each value of η_1), the number of sample paths that are maintained in parallel could be anywhere from 1 to 4.

The process is continued that way: some of the sample paths (intervals) are split in two when Z_3 is observed, and so on. We thus construct a *tree* of sample paths, also called *simulation tree*. Each branch of the tree (or sample path) *stops* when the value of $\tau(\theta)$ associated with the corresponding interval is reached. Some branches may end up sooner than others and since Ψ is finite w.p.1, the tree is finite w.p.1. When all the sample paths have ended, we recover from our parallel simulations the values of $h(\theta)$ for each interval and, from that, construct the piecewise constant function $h(\cdot)$ of Proposition 2. This entire process can be repeated, say, N times, and a functional estimator $\bar{h}(\cdot)$ of α be constructed by averaging out those N piecewise constant functions.

Remark 1 Our assumptions imply that $h(\cdot)$ is piecewise-constant with jumps only at the v_i's. Our model could of course be slightly generalized to deal with objective functions that are (known) transformations of a function which satisfies our assumptions, and that are not necessarily piecewise-constant themselves. For example, let $\tilde{h}(\theta) = g(\theta)h(\theta) + f(\theta)$, where g and f are known well-behaved (deterministic) functions of θ while $h(\theta)$ is random and satisfies our assumptions. In that case, we can use the split-and-merge approach to estimate $E[h(\cdot)]$ in functional form, then compute and unbiased functional estimator of $E[\tilde{h}(\cdot)]$. In general, the latter will *not* be piecewise-constant.

A major drawback of the split-and-merge method just presented, that most readers have certainly noticed already, is the fact that the size of the simulation tree (and the number of breakpoints) may increase exponentially fast w.r.t. $\sup_\theta \tau(\theta)$. That would make the method impractical when a large number of binary decisions are made. Indeed, it is easily seen from the preceding discussion (and from Proposition 1) that the simulation tree could have (in the worst case) 2^i nodes after generating Z_i for all θ. Of course, this pessimistic scenario is an artificial worst case that may never happen for a given application. Nevertheless, the size of the tree is likely to grow very fast.

There are interesting situations, however, where the growth rate of the simulation tree can be proved to be much slower than exponential. For instance, if there is a constant K such that no more than K splittings (in total) could occur at any level i of the tree, then the width of the tree could not grow faster than linearly: the number of parallel simulations after time T_i is bounded by iK. An important special case of this is when $K = 1$, which happens under the following assumption.

Assumption 2 *For almost all $\omega \in \Omega$ and for each i, there is at most one value of θ in $[a, b]$ for which $Z_i(\theta) = \theta$.*

This assumption holds for several interesting applications, as illustrated in the next section. Notice that the assumption automatically holds if the Z_i's are independent of θ, which will in fact be the case for most of the applications that we have in mind. This happens in particular when the decisions are Bernoulli random variables with parameter θ, in which case the Z_i are $U(0,1)$ random variables.

PROPOSITION 3 *Under Assumptions 1 and 2, one has*

$$\tau' \leq \sup_{\theta \in [a,b]} \tau(\theta).$$

Proof: Under Assumption 2, it is clear that $|\Psi_i| \leq 1$ for each i. Therefore, $|\Psi| \leq \sup_\theta \tau(\theta)$ and the result follows. ■

For typical simulation models, the expected computational effort to perform a single simulation at a fixed parameter value θ is roughly proportional to the time-horizon length $T(\theta)$, at least for large time horizons. If C_f denotes the total simulation effort for functional estimation, then, at worst, with the implementation discussed above, one has $C_f = O(\tau' \sup_\theta T(\theta))$. Let us further assume that $\tau(\theta)$ is proportional to $T(\theta)$ (which is also typical) and the constant t is an upper bound for the expression (4); that is, $\sup_\theta T(\theta) \leq t$ w.p.1. Then, $\tau' = O(t)$ and $C_f = O(t^2)$ if Assumption 2 holds, while $\tau' = O(2^t)$ and $C_f = O(2^t t)$ (in the worst case) otherwise.

If n independent replicates of the functional estimator are computed over, say, a finite horizon t, and then averaged out to obtain a better functional estimator $\bar{h}$, then we should typically have the following under Assumption 2: the total computational effort for performing the n simulations will be $O(nt^2)$, the total number of breakpoints of $\bar{h}$ will be $O(nt)$, and the total effort to compute $\bar{h}$ will be $O(nt^2 + nt\log(nt)) = O(nt(t + \log(nt))$, where $O(nt\log(nt))$ is the time required for sorting all the breakpoints in increasing order.

When coupling occurs between the trajectories of two or more neighboring intervals, there is no longer need to maintain distinct parallel simulations for those intervals. We will discuss how this can be exploited to reduce the computations, in the context of the $G/G/1$ example of the next section. Note, however, that when neighboring intervals are merged due to coupling, the fact that the system evolution has been different for a while over those intervals could result in different values of $h(\theta)$, depending on how this function is defined. So, coupling reduces the number of simulations that must be performed in parallel, but does not necessarily reduce the number of breakpoints τ', in the sense that the information relative to the computation of $h(\theta)$ must still be maintained (in general) over each of the intervals merged by coupling. In Example 9, we give an illustration where the total computational effort for functional estimation is in $O(t \log t)$ where t is the total number of customers simulated in a single queue.

If θ is a d-dimensional vector with $d \geq 2$ (several parameters), functional estimation can still be performed in principle. However, in that case, the number of pieces where $h(\cdot)$ is constant (i.e., the number of nodes in the simulation tree) could increase as $O((\tau')^d)$. This quickly becomes impractical in general if d exceeds 2 or 3.

3. Applications and Examples

3.1. A G/G/1 Queue with Mixed Service Time Distribution

Consider a single queue with arbitrary interarrival and service time distributions. Assume that the interarrival times are i.i.d. and that each arrival is of type I (or II) with probability θ (respectively, $1-\theta$), independently of all other randomness in the model, where $\theta \in \Theta = [0,1]$. For $i \geq 1$, let $W_i(\theta)$, $S_i(\theta)$, and $X_i(\theta) = W_i(\theta) + S_i(\theta)$ denote the *waiting* time, *service* time, and *sojourn* time of customer i, respectively, and let A_i be the time between arrivals of customers i and $i+1$. Define $\eta_i(\theta) = 1$ if arrival i is of type I, $\eta_i(\theta) = 0$ otherwise. Here, the Z_i's can be made as a sequence of i.i.d. uniform random variables over the interval (0,1), independent of everything else, therefore Assumption 2 is satisfied. We suppose that the two types of customers have different service time distributions, say B_{I} for type I and B_{II} for type II. For each i, let $S_{i,1}$ and $S_{i,2}$ be two random variables with respective distributions B_{I} and B_{II} (independent of the interarrival and other service times, and also independent of the Z_i's) and let

$$S_i(\theta) = \eta_i(\theta) S_{i,1} + (1 - \eta_i(\theta)) S_{i,2}.$$

Assuming that the queue has infinite capacity, the evolution of this system can be described as usual by Lindley's equation:

$$W_{i+1}(\theta) = (W_i(\theta) + S_i(\theta) - A_i)^+, \tag{7}$$

for $i \geq 1$, where x^+ means $\max(x, 0)$, and $W_1(\theta) = s \geq 0$ is the initial state of the queue ($W_1(\theta) = 0$ corresponds to an initially empty system). Note that $W_i(\theta)$ depends on θ only through $\eta_1(\theta), \ldots, \eta_{i-1}(\theta)$.

A sample point ω can be identified with $\{(A_i, S_{i,1}, S_{i,2}, Z_i),\ i \geq 1\}$. Let τ be a (possibly random) stopping time for this system and suppose that the performance measure of interest is additive:

$$h(\theta) = \sum_{i=1}^{\tau} f_i(W_i(\theta), S_i(\theta), \eta_i(\theta))$$

for some measurable functions $f_1, f_2, \ldots$. Then, Propositions 1–3 apply and the v_j's are just the values of the Z_i's, sorted by increasing order. For example, τ can be the number of customers entering the system during a given fixed time interval when $\theta = 1$ (in that case, τ would be independent of θ). The random variable $h(\theta)$ could represent for example the total waiting time for a given type of customer, or the number of customers whose waiting time exceeds a given constant, and so on. In some cases, what we really want to estimate is a nonlinear function of several mathematical expectations (such as a ratio of two expectations; see Examples 7 and 9), each being the expectation of an additive function as above. In that case, each such expectation is estimated separately as described here, from a single simulation experiment, by maintaining the required number of statistical accumulators in parallel. This could be replicated several times and each expectation estimated by the corresponding sample average.

Let us look more closely at how the simulation tree will evolve in this case. Observe that for a fixed ω, when θ goes down from, say, I_j to I_{j-1}, the only changes in the system's evolution are due to the fact that $\eta_i(\theta)$ goes from 1 to 0, i.e., $S_i(\theta)$ changes from $S_{i,1}$ to $S_{i,2}$, where i is the index such that v_j is the value of Z_i. In other words, the Lindley equations associated with adjacent intervals evolve in almost exactly the same way. The generation of the random variables A_i, $S_{i,1}$, $S_{i,2}$ and Z_i need not be duplicated. During the simulation, the value of $W_i(\theta)$ must be maintained, using equation (7), for each interval, i.e., for each node of the simulation tree. When $W_{i+1}(\theta)$ is to be computed, i customers have arrived, and so the interval $[0, 1]$ is divided at that moment into $i + 1$ subintervals (the simulation tree has $i + 1$ nodes).

Coupling would occur when the values of $W_i(\theta)$ become the same for two or more neighboring intervals I_j. The corresponding intervals can then be merged for the computation of $W_j(\theta)$ for $j > i$ by Lindley's equation, at least until further splitting occurs within those merged intervals. On the other hand, the different values of $\sum_{j=1}^{i} f_j(W_j(\theta), S_j(\theta), \eta_j(\theta))$, over the different intervals which have been merged, must be memorized for the evaluation of $h(\theta)$ later on. Of course, if we are estimating M mathematical expectations in parallel, then there are M values to memorize over each interval.

Remark 2 We have assumed in this section that $[a, b] = [0, 1]$. The decision epochs $T_i(\theta)$ correspond to the arrivals epochs, which are shared by all the nodes in the simulation, since these epochs are independent of θ. At each arrival, due to Assumption 2, only one node splits and a single breakpoint $z_{i,1}$ is added to Ω. This means that the total number of breakpoints (and also the size of the simulation tree, if we assume that no merging is performed) is proportional to the number of arrivals so far, which is $\tau(\theta)$ at the end of the simulation. The computational cost for a simulation is then proportional to $\sum_{i=1}^{\tau(\theta)} i \approx \tau^2(\theta)/2$ if no merging is performed. On the other hand, if the interval $[a, b]$ for which we are interested in having a functional estimation is only a strict subinterval of $[0, 1]$, then we only need to take care of the uniform variates Z_i that fall in that subinterval for the construction of the simulation tree. The number of such variates will be τ' and one has $E[\tau']/E[\tau] = b - a < 1$. More generally, the expected number of breakpoints after the ith arrival should be proportional to $(b - a)i$, which implies that for a fixed simulation length, the expected computational cost for functional estimation is roughly proportional to the size of the interval $[a, b]$ of interest.

Remark 3 This setup is straigthforward to generalize to the case where only *some* of the customers are of type I or II. For example, as a slight generalization of [13], we may assume that there are several classes of customers, each class having its own arrival stream (with possible correlations between the streams or within each stream), and that only the customers of a specific class (say, class c) can be of type I or II as above. The A_i's in (7) are then the interarrivals for the superposition of all arrival streams. The setup is also easily generalized to the case where the queue has more than one server; however, the Lindley equations (7) must then be replaced by slightly more complicated equations.

Remark 4 This single queue could be part of a large open queueing network. Call it queue q. If we assume that the customers leaving queue q cannot influence any longer the arrival process to queue q (e.g., no feedback), then all the development of this subsection still holds. Note that feedback is still allowed within other parts of the network.

We will now examine more specific variants of that single-queue example and see how our approach works for each of them. The first example is an admission control problem, where each arriving customer is accepted with probability θ. Examples 2–6 discuss different problems which fit the framework of Example 1.

Example 1 Suppose that $S_{i,2} = 0$ w.p.1 for all i. This represents a system with only one type of customers, with service time distribution B_{I}, but where each customer is rejected with probability $1-\theta$, independently of everything else. Here, $\eta_i(\theta)$ represents the decision of accepting the i-th arrival into the system (admission control). For the rejected customers, $W_i(\theta)$ and $X_i(\theta)$ are "virtual" waiting times and sojourn times, but are nevertheless well defined, and $S_i(\theta) = 0$.

In this case, our method is strongly linked with the so-called *phantom method* [19], [24], [2], [4]. In the last three references, the phantom RPA method is used to estimate derivatives. With that method, a *nominal* simulation is run at the parameter value $\theta = 1$ (no rejection); the performance h is also evaluated for the case where customer i is "phantomized" (rejected), for each i, and a derivative estimator is constructed from that. In [26], the phantom method was implemented for estimating the expected response of a system at a finite number of prespecified values of two control parameters. In the present example, the rejected customers can be viewed as "phantomized" and we could also call our approach "phantom method".

Here, whenever $W_i(\theta_0)$ becomes zero for some θ_0, we also have $W_i(\theta) = 0$ for all $\theta \le \theta_0$ and coupling is achieved. In other words, the systems with $\theta < \theta_0$ are *dominated* by the system with $\theta = \theta_0$; that is, the sample paths are *monotone* in θ. A particular case of such coupling is when the system empties out in the nominal sample path, i.e., when $W_i(1)$ becomes zero. Then, the simulation tree shrinks back to a single node. If the cost function $h(\theta)$ is additive between busy cycles, which is typically the case, then one only needs to memorize the piecewise constant (sample) cost functions associated with the different busy cycles. The number of pieces in each such piecewise constant function will be equal to the number of customers in the cycle, plus one. Functional estimation can then be performed very efficiently if the number of customers in a busy cycle has a relatively small second moment (recall that the work per busy cycle is roughly proportional to that second moment). In view of Remark 3, this also covers (with trivial adaptations) all the examples given in [13].

Example 2 The domination discussed in the previous example occurs more generally. For example, suppose that the service distribution is of the form $S_i(\theta) = \nu_1 I[\eta_i(\theta) = 1] + \nu_2 I[\eta_i(\theta) = 0]$, where $\nu_1 > \nu_2$. Here the two types of customers have different service requirements and each bifurcation in the simulation tree corresponds to changing the attribute of a customer from ν_1 to ν_2. Since we are using common random variables Z_i across the nodes of the tree, then clearly $S_i(\theta_1) \leq S_i(\theta_2)$ whenever $\theta_1 \leq \theta_2$, which implies domination.

Example 3 Consider an $M/G/1$ queue with arrival rate λ, and suppose that we want to estimate some (expected) performance measure as a function of λ, for $0 \leq \lambda \leq \lambda_0$, where λ_0 is a positive constant smaller than the inverse of the mean service time (so that the system is stable). Define $\theta = \lambda/\lambda_0$. Then, simulating that system at λ is equivalent to simulating it at λ_0 and rejecting each arrival with probability $1 - \theta$. This "thinning" idea is well-known in the area of stochastic processes and is often used for simulating non-homogeneous Poisson processes; see [1], [2], [3], [17]. Estimating a performance measure as a function of λ is equivalent to estimating it as a function of θ, which was the subject of the previous example.

For a specific example where this can be applied, consider the model (taken from [4]) of a central machine which allocates service slots to Q different servers. The total arrival rate is λ and each arrival is routed to server q with probability p_q, independently of everything else, for $1 \leq q \leq Q$. From the point of view of queue q, this is equivalent to accepting any arrival with probability p_q, so an expected performance measure that is a function of p_q alone can be estimated via our approach. Also, if the performance measure of interest can be expressed as $\sum_{q=1}^{Q} h_q(p_q)$ for some functions h_q that satisfy the assumptions of our framework, then we readily obtain a functional estimator of the expected value of each h_q (and so, of the expected performance measure as a function of $(p_1, \ldots, p_Q)$) by Q applications (perhaps in parallel) of our functional estimation approach.

Example 4 The same approach as in Example 3 can also be used if the queue has finite capacity K, although $S_i(\theta)$ would have to be defined differently than in the previous setup: let $S_i(\theta) = S_{i,1}$ if $\eta_i(\theta) = 1$ *and the queue is not full when customer i arrives* (which also depends on θ through $\eta_1(\theta), \ldots, \eta_{i-1}(\theta)$); $S_i(\theta) = 0$ otherwise. In that case, the sample paths are no longer monotone in θ, i.e., we loose the domination property of the systems with $\theta < \theta_0$ by the system with $\theta = \theta_0$. Indeed, when lowering θ slightly, we might cut out an arrival that has an extremely long service time, and as a result, accept more customers later on. Nevertheless, coupling will occur and can be exploited as well. See Example 7 for a numerical illustration. It is possible to recover domination by associating the service times with the server instead of with the arriving customers (i.e., by using $S_{i,1}$ as the service time of the ith customer that is *served* for any given value of θ, as suggested for example in Problems 2.1.1 et 2.1.8 of [1]), but implementing functional estimation using that approach seems messy and not so much useful.

Example 5 Consider again an $M/G/1$ queue with arrival rate λ, and let ν be a scale parameter of the service time distribution B_ν. In other words, each service time S_i can be generated by generating a random variable V_i from distribution B_1, and defining $S_i = \nu V_i$. Alternatively, one can use surrogate estimation via time rescaling (see [22]): generate the service times $S_i = V_i$ using distribution B_1, the arrivals at rate $\lambda\nu_0$ (where ν_0 is the largest parameter value of interest), and accept any given arrival with probability $\theta = \nu/\nu_0$. Here, estimating a performance measure as a function of ν is clearly equivalent to estimating it as a function of θ. The time units have been rescaled by the factor $\theta\nu_0$, which should be taken into account in the evaluation of $h(\theta)$.

Example 6 In the admission control setup, let $\tau = t$, a fixed constant, and

$$h(\theta) = \sum_{i=1}^{t} \eta_i(\theta) X_i(\theta), \tag{8}$$

the sum of sojourn times of all the customers that are accepted in the system. We have seen how to obtain a functional estimator of $\alpha(\theta)$, $0 \le \theta \le 1$, but now, suppose that we want to estimate the derivative of $\alpha(\nu)$ with respect to some parameter ν of the service time distribution $B_{\mathrm{I}} \equiv B_\nu$, and we want to estimate it everywhere as a function of θ. Let $\alpha'(\nu, \theta)$ denote that derivative. To estimate it, one can use under appropriate conditions (see [5], [14]) the following infinitesimal perturbation analysis (IPA) estimator:

$$h'(\nu, \theta) = \sum_{i=1}^{t} \eta_i(\theta) \sum_{j=\phi_i(\theta)}^{i} \eta_j(\theta) S'_j, \tag{9}$$

where $S'_j = (\partial B_\nu^{-1}/\partial\nu)(B_\nu(S_{j,1}))$, and $\phi_i(\theta) = \max\{j \mid 1 \le j \le i \text{ and } W_j(\theta) = 0\}$ if that set is non-empty, $\phi_i(\theta) = 1$ otherwise. In other words, $\phi_i(\theta)$ is the first customer with index ≥ 1 in the busy period to which customer i belongs, for that value of θ. To estimate $\alpha'(\nu, \cdot)$ in functional form, just apply the same technique as discussed previously, with h replaced by h'. Note that this can also be accomplished with other kinds of derivative estimators [16].

Example 7 For a more specific non-trivial illustration, we consider a queue with two classes of customers (as in Remark 3), and with finite capacity K. As in Example 4, K is the maximum number of customers in the queue or in service, and all the customers arriving when the buffer is full are lost. Each class of customers has its own arrival stream. Within each class, the interarrival times and service times are i.i.d., with respective distributions F_1 and G_1 for class 1, and F_2 and G_2 for class 2. Suppose that F_1 is the $U(a_1, a_2)$ distribution (uniform between a_1 and a_2), F_2 is exponential with mean $1/\lambda$, G_1 is degenerate: all class 1 customers have their service times equal to ν_1, and G_2 is exponential with mean ν_2. Each class 1 customer is admitted unless the buffer is full, but each class 2 customer is "considered" only with probability θ, and then admitted if there is room.

We want to estimate some performance measures as functions of θ. More specifically, we are interested in the expectations of the following statistics, for the first

t arriving customers (accepted or rejected), starting with an empty system, where t is a fixed constant:

(1) The average waiting time in the queue per customer, for the customers that are accepted;

(2) The fraction of admitted customers whose waiting time exceeds a given constant (threshold) L;

(3) The fraction of customers that are rejected because of a full buffer (for class 2 customers, only those that are "considered" are counted).

We shall denote these statistics by $h_w(\theta)$, $h_\ell(\theta)$, and $h_k(\theta)$, and their expectations by $\alpha_w(\theta)$, $\alpha_\ell(\theta)$, and $\alpha_k(\theta)$, respectively.

Let $M_a(\theta)$ denote the number of admitted customers (among the first t), $M_\ell(\theta)$ the number af admitted customers whose waiting time exceeds L, and $M_k(\theta)$ the number of customers rejected because of a full buffer. For each i, let $R_i(\theta) = 1$ if customer i is admitted; $R_i(\theta) = 0$ otherwise. To define $\eta_i(\theta)$, let Z_i be a $U(0,1)$ random variable if customer i is of class 2, $Z_i = 0$ otherwise, and let $\eta_i(\theta) = I[Z_i \leq \theta]$. Then, $R_i(\theta) = 1$ if and only if $\eta_i(\theta) = 1$ and the buffer is not full when customer i arrives. With that notation, we have

$$\begin{aligned}
h_w(\theta) &= \frac{1}{M_a(\theta)} \sum_{i=1}^{t} R_i(\theta) W_i(\theta); \\
h_\ell(\theta) &= \frac{M_\ell(\theta)}{M_a(\theta)}; \\
h_k(\theta) &= \frac{M_k(\theta)}{M_a(\theta) + M_k(\theta)}.
\end{aligned}$$

We now report simulation experiments with the following values: $K = 10$, $L = 3$, $\theta \in [a,b] = [0.8, 1.0]$, $\nu_1 = 1.5$, $\nu_2 = 0.5$, $a_1 = 2$, $a_2 = 3$, and $1/\lambda = 2$. With these parameter values, the system would be utilized (in the long run) at 85% if all the customers were accepted. We made $N = 1000$ replications of the simulation, for an horizon of $t = 100$ customers, obtaining thus a sample of N i.i.d. observations at each value of θ, for each quantity of interest. We then computed the average and standard deviation of those observations. We do not know the exact values of the quantities to be estimated.

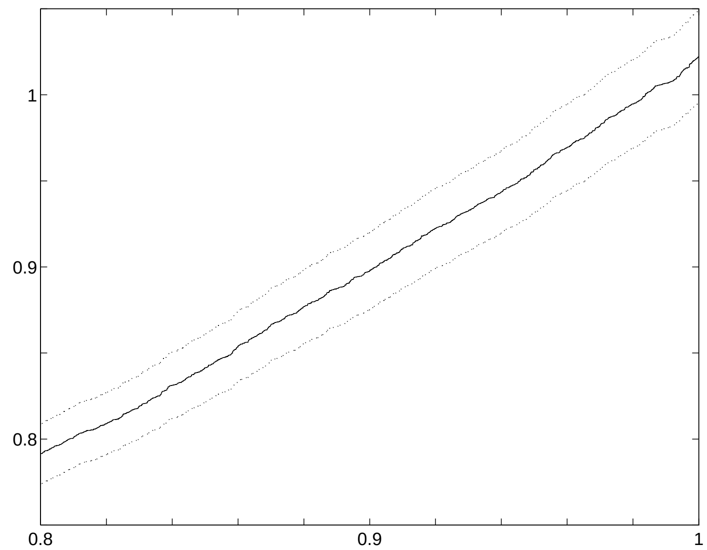

Figure 1. $\bar{h}_w(\theta)$ for $0.8 \leq \theta \leq 1$, $t = 100$, $N = 1000$

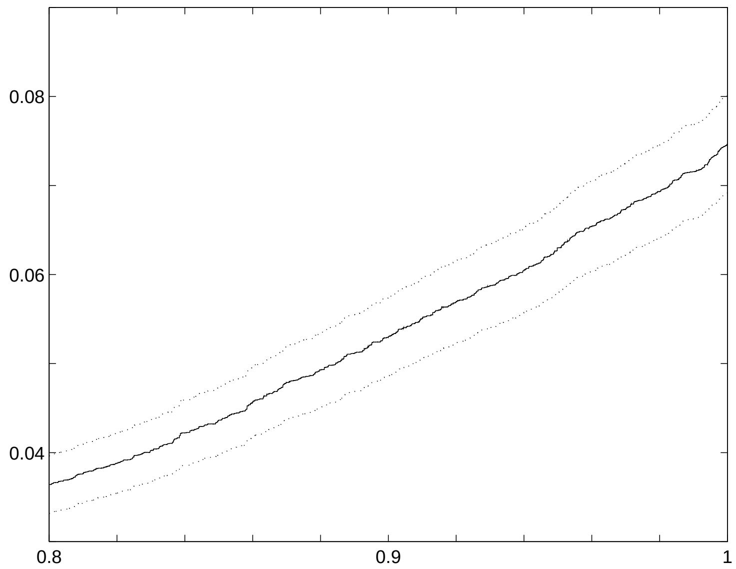

Figure 2. $\bar{h}_\ell(\theta)$ for $0.8 \leq \theta \leq 1$, $t = 100$, $N = 1000$

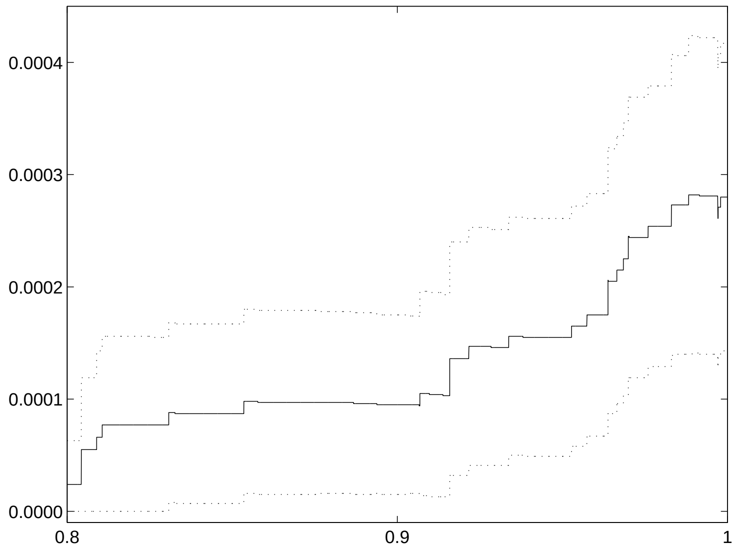

Figure 3. $\bar{h}_k(\theta)$ for $0.8 \le \theta \le 1$, $t = 100$, $N = 1000$

Figures 1–3 show the average (solid lines), for the three quantities of interest. The dotted lines are *two* (empirical) standard deviations above and below the empirical mean. Recall that both the empirical and theoretical means and variances of each estimator at any given value of θ are exactly the same as if the simulation was performed only at that value of θ. We verified that in our simulations. As expected, the variance is pretty much the same all over the interval considered for θ, and the functional estimator is nicely behaved. The three functions are piecewise constant. This is apparent for $\bar{h}_k$, which turns out to have few jumps because $M_k(\theta)$ tends to take very small values (it is equal to zero over $[0.8, 1.0]$ for most replications). On the other hand, $\bar{h}_w$ and $\bar{h}_\ell$ have so many breakpoints that one cannot visually distinguish the constant pieces on the graphics. Note that while $M_a(\theta) + M_k(\theta)$ is monotone increasing in θ, it is not necessarily so for $M_k(\theta)$, because by increasing θ, one may get rid of a customer having an extremely long service time, and loose less customers as a result. Even if both were monotone increasing, their ratio $h_k(\theta)$ would not necessarily be monotone, as illustrated by Figure 3: By increasing θ, $M_a(\theta)$ may increase while $M_k(\theta)$ remains unchanged, in which case the ratio $h_k(\theta)$ decreases.

We also ran simulations and produced graphics (not reported here) for several other parameter sets, including different values of t and different system utilizations, and the results were similar, except, of course, that the variances were higher for higher utilizations and lower for larger values of t.

We now discuss the empirically observed performance of the method. We may want to compare the total cpu time required to perform the functional estimations with that required to obtain an estimation only at $\theta = 1.0$. Our implementation of the functional estimation exploited the fact that coupling occurs whenever the system empties out for two or more neighboring intervals. When coupling occurs for several intervals during a simulation, the statistics cumulated up to that point for those intervals must be stored (e.g., in a list, to be processed later on), and the statistical counters reset to zero. At the end of the simulation, all the statistics are properly merged, yielding the piecewise-constant functions h_w, h_ℓ, and h_k. Then, when all the simulations are done, the N replicates of each function are averaged out, yielding a piecewise-constant function with much more pieces: its breakpoints are the union of all the breakpoints of all the replicates. Therefore, the second-level merge involves sorting all those breakpoints. To achieve that, our implementation uses a heap that contains the N lists obtained after the first-level merges. Each list is positioned into the heap according to the value of its smallest breakpoint not yet considered by the second level merge. At each step, one breakpoint from the list at the root of the heap is "considered" (added to the sorted list of all breakpoints) and the heap is then updated.

When counting cpu times, we distinguish (a) the time for performing the N simulations; (b) the time for doing all the first-level merges (this is relevant only when coupling is exploited); and (c) the time of the second-level merge using the heap. Note that if only an estimation at a fixed value of θ is desired, then only (a) is relevant. Table 1 gives certain statistics on the cpu times, list and heap sizes, number of breakpoints, coupling frequency, and so on, for the numerical example considered above, with different values of the horizon t. We also ran (separate) simulations without performing any coupling, and simulations for estimating the performance measures only at $\theta = 1$; the corresponding cpu times appear in the table under the headings "no coupling" and "$\theta = 1$". All the cpu times are in seconds and represent the total time for the N replications (they exclude the time required to output the results to computer files). The numbers of couplings and of breakpoints are also the totals for the N replications (so, 11214 is the number of breakpoints in the graphics of Figures 1–3, with the proviso that the functions do not necessarily have jumps at each breakpoint, since adjacent intervals may have the same value of the estimator; this is what happens in Figure 3 for instance). The average maximum size of the tree is the maximum size of the simulation tree during a simulation, averaged over the N replications. Similarly, the average size of the merge list is the average (over the replications) number of intervals at the end of a simulation. The number of intervals is the number of breakpoints, plus one. The maximum size of the tree can be smaller than the number of intervals only when coupling occurs.

From these statistics, one can see that when coupling is performed, the size of the simulation tree does not grow significantly and the cpu time increases (roughly) linearly with the horizon length t. Without coupling, the increase looks more like quadratic, which agrees with the fact that the simulation tree increases linearly with t. For large t, for this example, the functional estimation with coupling requires

Table 1. Statistics on the performance of the split-and-merge for Example 7.

t	50	100	200	400	800	1600
Simul. with coupling						
Total cpu time (sec)	20.2	39.7	95.9	174.8	376.9	837.0
simulation	10.4	20.8	51.1	89.0	194.4	433.5
level 1 merge	1.2	2.5	5.8	12.1	35.9	123.2
level 2 merge	7.2	15.1	36.3	72.0	144.1	277.1
total nb. couplings	3086	6420	13250	26704	53459	107139
total nb. breakpoints	5648	11214	22389	44695	89154	177864
Aver. max. size of tree	3.7	4.7	5.8	7.0	8.2	9.4
Aver. size of merge list	6.6	12.2	23.4	45.7	90.2	178.9
No coupling						
Total cpu time	22.3	68.1	143.0	480.6	1396.2	4928.3
simulation	12.5	44.1	104.7	393.1	1259.2	4635.0
level 2 merge	7.7	19.6	33.8	79.9	127.7	277.7
Aver. max. size of tree	6.6	12.2	23.4	45.7	90.2	178.9
Aver. size of merge list	6.6	12.2	23.4	45.7	90.2	178.9
Estim. only for $\theta = 1$						
cpu time (simulation)	6.4	12.1	24.4	53.3	115.0	218.4

between 3 and 4 times that required by an estimation at $\theta = 1$ alone, if we include the overhead for the two levels of merging. Without including that overhead, the simulation time for the functional estimation is less than twice that for $\theta = 1$ alone. Therefore, repeating the simulation over a fine grid of values of θ without splitting and merging, to obtain a functional estimator, would not be competitive with the *split-and-merge* approach in this case.

Our simulation programs were written in the Modula-2 language and run on a SUN SPARCstation 2. We made no special effort to optimize the code, but tried to make it reasonable. The cpu times that we report are just giving a rough indication of what goes on. They may certainly change somewhat by refining the implementation or changing the compiler; however, we believe that the general conclusions will remain the same.

3.2. A $GI/GI/1$ queue over one regenerative cycle

Consider the same queueing model as in the previous subsection, with independent interarrival and service times. This time, however, we assume that $h(\theta)$ is a function of the system's behavior over one regenerative cycle, where the regeneration points are defined as the instants at which a customer enters an empty system. For example, $h(\theta)$ could be the number of customers of type I served in a cycle, or the total waiting time of type I customers in a cycle, and so on. Note that by definition, a regenerative cycle must contain at least one customer, whatever the value of θ. This must be taken into account, as shown by the next example.

Example 8 Let us return to the admission control model of Example 1. To make sure that we are dealing with one regenerative cycle for any θ, we can use the same trick as in [2], [24]: always accept the first customer, independently of $\eta_1(\theta)$, and start admission control only from the second customer on. Then, $\tau(\theta)$ is equal to the number of customers in the cycle; it is a nondecreasing step function of θ, with $\tau(0) = 1$, and $\tau(1)$ equal to the number of customers in the cycle when no customer is rejected.

Example 9 In the preceding example, suppose that the queue is $GI/GI/1$ with interarrival and service time distributions F and B_ν, respectively, where F is continuous with density f, and B_ν depends on a parameter ν as in Example 6. Let $\alpha(\nu, \theta) = E_\nu[\tau(\theta)]$ and $\alpha'(\nu, \theta)$ its derivative with respect to ν. To estimate these quantities in functional form, as functions of θ, we can use the following SPA derivative estimator, adapted from [24].

Observe that $1/\alpha(\nu, \theta)$ is equal to the fraction of customers that are first in their busy cycles, in steady-state, among those that are accepted. So,

$$\alpha(\nu, \theta) = 1/P_{\nu,\theta}[W = 0],$$

where $P_{\nu,\theta}[W = 0]$ is the probability that a randomly chosen customer, in steady-state, has a zero waiting time, given ν and θ. Differentiating that, we obtain

$$\alpha'(\nu, \theta) = -\frac{1}{(P_{\nu,\theta}[W = 0])^2} \frac{\partial}{\partial \nu} P_{\nu,\theta}[W = 0],$$

which can be estimated indirectly by estimating $P_{\nu,\theta}[W = 0]$ and its derivative. Now, since each customer is accepted with probability θ independently of the past, we have that $P_{\nu,\theta}[W = 0]$ is also equal to the probability that a random arriving customer, accepted or rejected, in steady-state, sees the system empty when it arrives. Assuming that the system starts empty, let $W_{i+1}(\theta)$ denote the waiting time that customer $i + 1$ would have if it was accepted, for $i \geq 1$. An unbiased estimator of $P[W_{i+1}(\theta) = 0]$ is given by the conditional probability:

$$\begin{aligned} P_i(\theta) &= P[W_{i+1}(\theta) = 0 \mid X_i(\theta)] \\ &= P[A_i \geq X_i(\theta) \mid X_i(\theta)] \\ &= 1 - F(X_i(\theta)), \end{aligned}$$

where $X_i(\theta) = W_i(\theta) + S_i\eta_i(\theta)$ and $W_{i+1}(\theta) = (X_i(\theta) - A_i)^+$. When we perform our simulations, the first customer is always accepted independently of $\eta_1(\theta)$, but the value of $\eta_1(\theta)$ (a Bernoulli random variable with parameter θ) is nevertheless used in the expression for $X_1(\theta)$. Here, we take as a regenerative cycle (for all θ) the busy period that corresponds to $\theta = 1$. It is okay to always accept the first customer for the same reason as in Example 8. However, the expression for $P_1(\theta)$

must take into account the possibility that $W_2(\theta) = 0$ for a smaller θ due to a rejection of customer 1; this is why $\eta_1(\theta)$ should not be always 1. From standard renewal-reward theory, one has

$$P_{\nu,\theta}[W=0] \;=\; \frac{E\left[\sum_{i=1}^{\tau(1)} P_i(\theta)\right]}{E\left[\tau(1)\right]} \;\overset{\text{a.s.}}{=}\; \lim_{t\to\infty} \frac{1}{t}\sum_{i=1}^{t} P_i(\theta),$$

where $\tau(1)$ is the number of customers in the first busy cycle at $\theta = 1$ (the smallest $i \geq 1$ such that $W_{i+1}(1) = 0$). Likewise, under appropriate uniform integrability conditions (see [5], [14]), one has

$$\frac{\partial}{\partial\nu} P_{\nu,\theta}[W=0] \;=\; \frac{E\left[\sum_{i=1}^{\tau(1)} P_i'(\theta)\right]}{E\left[\tau(1)\right]}$$

where $P_i'(\theta)$ is the sample derivative of $P_i(\theta)$ w.r.t. ν:

$$P_i'(\theta) = \frac{\partial}{\partial\nu}(1 - F(X_i(\theta))) = -f(X_i(\theta)) \cdot X_i'(\theta)$$

and

$$X_i'(\theta) = \sum_{j=\phi_i(\theta)}^{i} \eta_j(\theta) S_j'$$

similar to Example 6.

To estimate $\alpha'(\nu,\theta)$, one can simulate the system over n regenerative cycles at $\theta = 1$ and compute the following quantities as functions of θ. Let $Y_j(\theta)$, $Y_j'(\theta)$, and $\tau_j(1)$ denote the values of $\sum_{i=1}^{\tau(1)} P_i(\theta)$, $\sum_{i=1}^{\tau(1)} P_i'(\theta)$, and $\tau(1)$, respectively, for the jth regenerative cycle. Assuming that the uniform integrability conditions hold and that the system is stable at $\theta = 1$, the following is a strongly consistent estimator of $\alpha'(\nu,\theta)$:

$$\left(\sum_{j=1}^{n} \tau_j(1)\right)\left(\sum_{j=1}^{n} Y_j'(\theta)\right)\left(\sum_{j=1}^{n} Y_j(\theta)\right)^{-2} \;=\; -T\sum_{i=1}^{T} P_i'(\theta)\left(\sum_{i=1}^{T} P_i(\theta)\right)^{-2}, \tag{10}$$

where $T = \sum_{j=1}^{n} \tau_j(1)$ is the total number of arrivals (accepted or discarded) during the n cycles. A strongly consistent estimator of $\alpha(\nu,\theta)$ is given by:

$$\left(\sum_{j=1}^{n} \tau_j(1)\right)\left(\sum_{j=1}^{n} Y_j(\theta)\right)^{-1} \;=\; \left(\frac{1}{T}\sum_{i=1}^{T} P_i(\theta)\right)^{-1}. \tag{11}$$

Here, all the regenerative cycles (for $\theta = 1$) can be simulated independently of each other and the simulation tree rarely gets large unless the system utilization is very close to one.

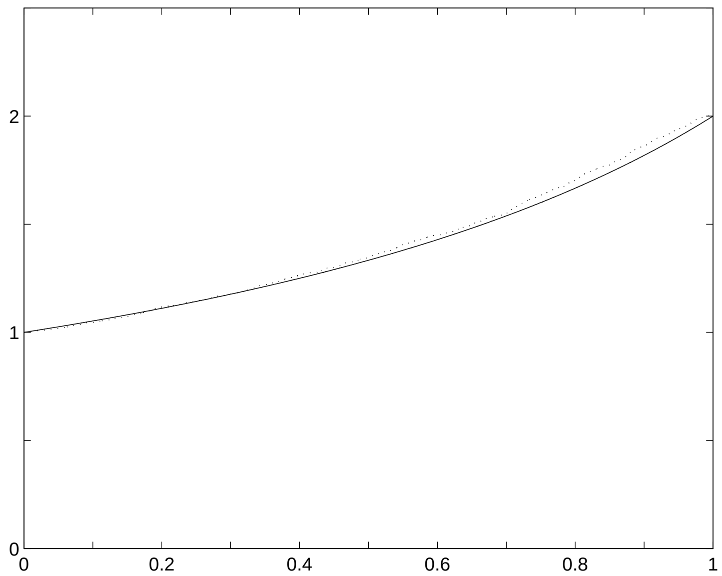

Figure 4. $\alpha(\nu, \theta)$ (solid line) and its estimator (dotted line) for $\nu = 0.5$ and $n = 1000$.

Table 2. Statistics on the performance of the split-and-merge for Example 9.

n	2500	5000	10000	20000	40000	80000
Functional estimation						
Total cpu time (sec)	5.8	12.4	25.2	58.0	114.1	241.3
simulation	3.2	6.3	12.1	27.3	52.2	107.3
heap sort and merge	2.6	6.1	13.1	30.7	61.9	134.0
total nb. breakpoints	5042	9990	19775	39651	79364	158818
total nb. customers	5042	9990	19775	39651	79364	158818
Aver. max. size of tree	3.02	3.00	2.98	2.98	2.98	2.99
Aver. size of merge list	3.02	3.00	2.98	2.98	2.98	2.99
Estim. only for $\theta = 1$						
cpu time (simulation)	1.9	4.5	8.4	17.6	35.9	72.1

We performed numerical experiments with this example. In order to compare our results with the exact (theoretical) values, we took an $M/M/1$ queue as in [24], with arrival rate of 1 and mean service time ν. In that case, one has $S_j' = S_j/\nu$. Figures 4 and 5 show the values of the functional estimators (11) and (10) for $\nu = 0.5$, $0 \le \theta \le 1$, and $n = 1000$. Those values are given by the dotted lines, whereas the solid lines indicate the theoretical values. Note that the expected number of customers per cycle at $\theta = 1$ is only 2. We also performed experiments with larger

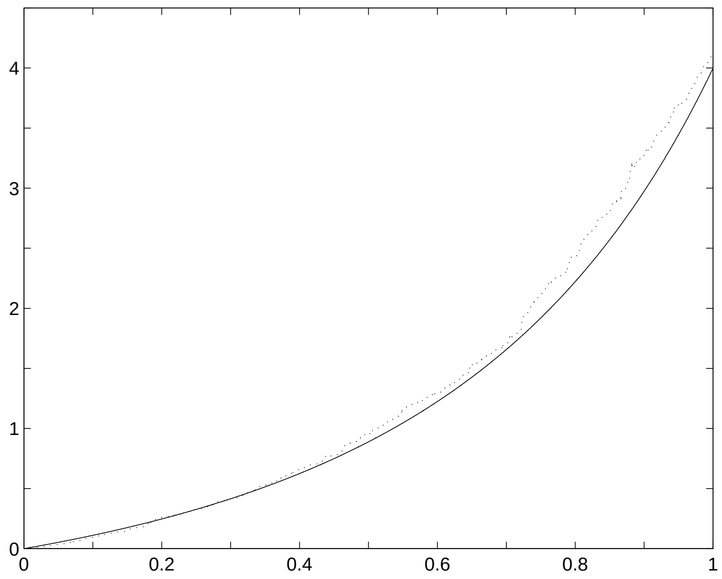

Figure 5. $\alpha'(\nu, \theta)$ (solid line) and its estimator (dotted line) for $\nu = 0.5$ and $n = 1000$.

values of n and the two curves quickly became very close to each other. We tried different values of ν and the results were similar.

Table 2 gives some statistics on the performance of the algorithm, similar to those given in Table 1 for Example 7, for different values of n, with $\nu = 0.5$. Similar results were obtained for other values of ν, with the proviso that the average number of customers per cycle (and therefore the work) increases rapidly with ν. Here, the average size of the simulation tree at the end of a cycle is approximately 3 (the number of customers in the cycle, plus one), and the total number of breakpoints is the same as the total number of customers simulated, which is approximately $2n$ (for all values of θ). The simulation time is approximately proportional to n; that is, proportional to the total number of customers. On the other hand, the merging times are in $O(n \log n)$, so the total computational effort increases slightly faster than linearly.

3.3. Age replacement policies in a multicomponent system

Example 10 This example is taken from [11], [14]. A system is comprised of N components, evolving independently, each with the same lifetime distribution, with increasing failure rate. We assume that the maintenance policy is defined by two thresholds $\theta_1 > \theta_2 > 0$ as follows. Whenever a failure occurs or a component reaches age θ_1, then all the components older than θ_2 are replaced (instantaneously) by new ones. The replacement cost at each intervention is c_i, plus c_r times the number of components that are replaced. There is also an additional cost c_f incurred each time a component fails. Here, the randomness ω may be viewed as corresponding to the sequence of component lifetimes. Let T be a fixed time horizon and $h(\theta)$ be the total cost incurred up to time T, where $\theta = (\theta_1, \theta_2)$, and $\alpha(\theta)$ its expectation.

In [11], it is shown how to compute an optimal policy for this problem by dynamic programming, for either the finite horizon or the infinite-horizon case with discounting. It turns out that the class of two-threshold policies defined above is suboptimal; the optimal policy is generally much more complicated than that. Nevertheless, the two-threshold policies are interesting because they are much simpler to implement and the best one is typically very close to the optimum. Moreover, the numerical methods used in [11] work well for small N (say, $N \leq 3$ or 4), but become impractical for large N due to the curse of dimensionality.

Suppose that for a fixed θ_1, we want to estimate $\alpha(\theta_1, \theta_2)$ as a function of θ_2. In the framework of Section 2, the binary decisions here are whether to replace or not each of the remaining components whenever an intervention occurs. In general, at each intervention, there are (up to) $N - 1$ such binary decisions. Unfortunately, this example does not satisfy Assumption 2, so it is conceivable that the expected computational effort grows much faster than linearly with T. However, our preliminary numerical experiments indicate that it is not necessarily so, and we intend to pursue further investigation (both numerical and theoretical) on this application.

Acknowledgments

This work has been supported by NSERC-Canada grants # WFA-0139015 and # OGP-0110050, as well as FCAR-Québec grant # 94-ER-1654. Benoît Martin performed the numerical experiments reported in the paper.

References

1. Bratley, P., Fox, B. L., and Schrage, L. E. 1987. *A Guide to Simulation*, Springer-Verlag, New York, second edition.
2. Brémaud, P. and Vázquez-Abad, F. J. 1992. On the Pathwise Computation of Derivatives with Respect to the Rate of a Point Process: The Phantom RPA Method, *Queueing Systems*, **10**, pp. 249–270.
3. Çinlar, E. 1975. *Introduction to Stochastic Processes*, Prentice-Hall.
4. Cassandras, C. G. and Julka, V. 1992. Marked/Phantom Slot Algorithms for a Class of Scheduling Problems, *Proceedings of the 31th IEEE Conf. on Decision and Control*, pp. 3215–3220.
5. Glasserman, P. 1991. *Gradient Estimation via Perturbation Analysis*, Kluwer Academic.
6. Fox, B. L. 1990. Generating Markov Chain Transitions Quickly: I. *ORSA J. Computing* **2**, pp. 126–135.
7. Fox, B. L. and Glynn P. W. 1990. Discrete-Time Conversion for Simulating Finite-Horizon Markov Processes. *SIAM. J. Applied Math.* **50**, pp. 1457–1473.
8. Fox, B. L. and Young, A. R. 1991. Generating Markov Chain Transitions Quickly: II. *ORSA J. Computing* **3**, pp. 3–11.

9. Glynn, P. W. 1990. Likelihood Ratio Gradient Estimation for Stochastic Systems, *Communications of the ACM*, **33**, pp. 75–84.
10. Glynn, P. W. and Iglehart, D. L. 1989. Importance Sampling for Stochastic Simulations. *Management Science* **35**, pp. 1367–1392.
11. Haurie, A. and L'Ecuyer, P. 1986. Approximation and Bounds in Discrete Event Dynamic Programming. *IEEE Transactions on Automatic Control* **AC-31**, 3 (1986), pp. 227–235.
12. Ho, Y.-C. and Cao, X.-R. 1991. Discrete-Event Dynamic Systems and Perturbation Analysis. Kluwer Academic.
13. Julka, V., Cassandras, C. G., and Gong, W.-B. 1992. Sample Path Techniques for Admission Control in Multiclass Queueing Systems with General Arrival Processes, *Proceedings of CISS'92*, pp. 227–232.
14. L'Ecuyer, P. 1990. A Unified View of the IPA, SF, and LR Gradient Estimation Techniques, *Management Science*, **36**, pp. 1364–1383.
15. L'Ecuyer, P. 1991. An Overview of Derivative Estimation. *Proceedings of the 1991 Winter Simulation Conference*, IEEE Press, pp. 207–217.
16. L'Ecuyer, P. 1993. Two Approaches for Estimating the Gradient in Functional Form. *Proceedings of the 1993 Winter Simulation Conference*, IEEE Press, pp. 338–346.
17. Lewis, P. A. W. and Shedler, G. S. 1979. Simulation of Nonhomogeneous Poisson Processes by Thinning, *Naval Research Logistics Quarterly*, **26**, pp. 403–414.
18. Rubinstein, R. Y. and Shapiro, A. 1993. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*, Wiley.
19. Suri, R. and Cao, X.-R. 1986. The Phantom and Marked Customer Methods for Optimization of Closed Queueing Networks with Blocking and General Service Times. *ACM Performance Evaluation Review*, **12**, 3, pp. 243–256.
20. Uryasev, S. 1995. Analytic Perturbation Analysis for DEDS with Discontinuous Sample-Path Functions", *Stochastic Models*, To appear.
21. Vakili, P. 1991. Using a Standard Clock Technique for Efficient Simulation. *Operations Research Letters* **10**, pp. 445–452.
22. Vázquez-Abad, F. J. 1996. Generalizations of the surrogate estimation approach for sensitivity analysis via simple examples. In preparation.
23. Vázquez-Abad, F. J., Kushner, H. J. 1993. The surrogate estimation approach for sensitivity analysis in queueing networks. *Proceedings of the 1993 Winter Simulation Conference*, IEEE Press, pp. 347–355.
24. Vázquez-Abad, F. J. and L'Ecuyer, P. 1991. Comparing Alternative Methods for Derivative Estimation when IPA Does Not Apply Directly. *Proceedings of the 1991 Winter Simulation Conference*, IEEE Press, pp. 1004-1011.
25. Vázquez-Abad, F. J. and L'Ecuyer, P. 1994. Simulation Trees for Functional Estimation via the Phantom Method. *Proceedings of the 11th International Conference on Analysis and Optimization of Systems: Discrete Event Systems*, Sophia-Antipolis, June 1994, in *Lectures Notes in Control and Information Sciences*, **199**, pp. 449–455.
26. Vázquez-Abad, F. J. and Jacobson, S. H. 1994. Application of RPA and the Phantom Harmonic Gradient Estimators to a Priority Queueing System. *Proceedings of the 1994 Winter Simulation Conference*, IEEE Press, pp. 369–376.