Improving Adaptive Importance Sampling Simulation of Markovian Queueing Models using Non-parametric Smoothing

Edwin Woudt, Pieter-Tjerk de Boer and Jan-Kees van Ommeren

SIMULATION 2007; 83; 811
DOI: 10.1177/0037549707087223

The online version of this article can be found at:
http://sim.sagepub.com/cgi/content/abstract/83/12/811
Improving Adaptive Importance Sampling
Simulation of Markovian Queueing Models
using Non-parametric Smoothing

Edwin Woudt
Pieter-Tjerk de Boer
Jan-Kees van Ommeren
Department of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede, The Netherlands
ptdeboer@cs.utwente.nl

Previous work on state-dependent adaptive importance sampling techniques for the simulation of rare events in Markovian queueing models used either no smoothing or a parametric smoothing technique, which was known to be non-optimal. In this paper, we introduce the use of kernel smoothing in this context. We derive expressions for the smoothed transition probabilities, compare several variations of the technique, and explore the choice of kernel width. We provide some examples, demonstrating that the technique significantly improves convergence and estimator variance.

Keywords: rare-event simulation, importance sampling, queueing networks

1. Introduction

In performance evaluation, events which occur only with a small probability are often of interest; for example, the probability of a complete system failure in reliability models or the probability of a buffer overflow in a data communication network. When such probabilities cannot be determined analytically or numerically, one may resort to simulation. However, the simulation of rare events typically needs a speed-up technique in order to obtain sufficiently accurate results in a reasonable time.

Several simulation speed-up methods have been proposed for the estimation of small overflow probabilities in queueing networks, often based on importance sampling. The challenge in importance sampling is choosing an appropriate 'change of measure' (see Section 2 for details). At first, methods were proposed in which this change of measure does not depend on the current state of the model. A specific state-independent heuristic was given by Parekh and Walrand [1], proven to be asymptotically efficient [2] for a single queue, but not asymptotically efficient for some simple networks [3]. De Boer [4] proved that no state-independent change of measure can be asymptotically efficient for some simple two-node tandem networks, thus formally establishing the need for a state-dependent change of measure. However, due to the many degrees of freedom, it is not easy to find a good state-dependent change of measure. Some heuristics for specific models are provided in [5, 6, 7], in some cases with asymptotic efficiency proofs [6, 7]. For more general models, however, these heuristics may not work well (if applicable at all). In cases where no heuristics are available, adaptive methods may be used to choose the change of measure (e.g. [8, 9, 10]). In the current paper, we will build on the method of de Boer and Nicola [9] which iteratively improves the change of measure using a cross-entropy minimization. For models up to about five queues, this method has been found experimentally to converge and typically result in asymptotical efficiency.

One key aspect of this adaptive method when dealing with large models is the need to apply smoothing. This is needed because a change of measure needs to be estimated separately for every state. However, not enough observations may be available from a limited simulation run to do so. Smoothing then exploits the fact that a good change of measure will typically be a smooth function of the model’s state variables (such as queue lengths). A spline-based smoothing technique was described by de Boer [11]. However, this spline-based method was far from ideal: it assumed that a cubic spline was a good fit, did not use all available information optimally and required some parameters to be set by the simulation user. Furthermore, the simulation user needed to (manually) try switching off the spline smoothing after convergence.
since after convergence this smoothing technique in many cases actually increased the estimator variance.

In the present paper, we present a smoothing method which does not have these disadvantages. It is based on local kernel regression with a kernel width that is tuned adaptively. Local kernel regression is a non-parametric smoothing technique and therefore does not assume that, for example, a polynomial is a good fit. Furthermore, the adaptive tuning of the kernel width means the simulation user does not need to choose any parameters, and after convergence, the kernel width is automatically reduced so far that the smoothing does not negatively affect the estimator variance.

The rest of this paper is organized as follows. In Section 2, the simulation method of de Boer and Nicola [9] is briefly introduced. In Section 3, local kernel regression is discussed expressions are derived for our specific situation, and appropriate choices for parameters are made and motivated. In Section 4, it is demonstrated experimentally that the method works well for a five-node network and in fact performs significantly better than the old spline-based method. Finally, Section 5 provides conclusions.

2. Preliminaries

In this paper, we consider the simulation of (discrete-time) Markovian queueing networks, and are particularly interested in transient overflow probabilities of the form: what is the probability of reaching some target state (e.g. states in which the total network population exceeds some threshold), starting from some specific state, before reaching some specific absorbing state (e.g. state in which the network is empty). In this section, we introduce some notation, followed by an overview of the adaptive importance sampling method of which the improved smoothing method will become part.

2.1 Notation

The (discrete-time) Markov chain representing the queueing network is specified completely by the transition probabilities from each state of the Markov chain to each other state. The transition probabilities in Markov chains describing queueing models have a lot of structure. For example, in a two-node tandem queue, there are only three possible transitions from most states, corresponding to: an external arrival to the first queue, a service completion at the first queue and a service completion at the second queue. The exceptions to this are the states at the boundary of the state space, i.e. where one of the queues is empty. For example, the transition corresponding to a service completion at an empty queue is impossible.

The highly regular structure of the Markov chain can be exploited to make the notation more convenient by mapping each of the possible transitions onto a ‘transition type’. In the two-node tandem example, there will be three transition types corresponding to arrival and to service completions at both queues. Symbolically, we will denote the original transition probabilities as \( q_{ik} \), for a transition of type \( k \) from state \( i \) to state \( k(x) \), where the function \( k(x) \) specifies the next state for transition type \( k \) out of state \( x \). Considering again the two-node tandem example, if \( k = 2 \) is defined to be the transition type for a service completion at the first queue, then \( \phi_2(x_1, x_2) = (x_1 - 1, x_2 + 1) \), with \( x_i \) denoting the content of the \( i \)th queue. Note that the mapping of integers \( k \) to transitions in the model is arbitrary. We will also denote the entire set of transition probabilities \( q_{ik} \) as a matrix (bold) \( q \).

Next, consider the estimation of overflow probabilities by simulation. Without importance sampling, this is simple: generate random sample paths from the starting state until they reach either the target or the absorbing state, and count how many reach the target state. Formally, this can be described as follows. Let \( Z \) denote one (random) sample path from the starting state to the target or the absorbing state; i.e. \( Z \) is a sequence of states \( x_1, \ldots, x_r \) that are visited subsequently, where \( r \) is the length of this sample path. This \( r \) is a stopping time: \( r \) is such that \( x_r \) is a target or absorbing state and for all \( i < r, x_i \) is not a target or absorbing state. Let \( I(Z) \) be the indicator function which is 1 if \( Z \) ends in the target state. Then the standard simulation estimator of the target probability \( \gamma \) is

\[
\hat{\gamma} = \frac{\sum_{i=1}^{N} I(Z_i)L(Z_i, q')}{N},
\]

where \( Z_1, \ldots, Z_N \) are \( N \) independent random sample paths under the original probability distributions.

If importance sampling is used, sample paths are generated under some other set of transition probabilities, denoted by \( \tilde{q} \). In order to obtain an unbiased estimate, all samples must be weighted by the likelihood ratio, resulting in the following expression for the estimator:

\[
\hat{\gamma} = \frac{\sum_{i=1}^{N} I(Z_i)L(Z_i, \tilde{q})}{N},
\]

where \( Z_1, \ldots, Z_N \) are \( N \) independent random sample paths under some modified probability distribution \( \tilde{q} \), and the likelihood ratio \( L(Z, \tilde{q}) \) is the probability of sample path \( Z \) under the original distribution \( q \) divided by its probability under the modified distribution \( q' \):

\[
L(Z, q') = \prod_{i=1}^{r-1} q_{x_i k_i} / \prod_{i=1}^{r-1} q'_{x_i k_i}
\]

where \( x_i \) are the states on \( Z \), and \( k_i \) is such that \( x_{i+1} = \phi_{k_i} x_i \). For more details on importance sampling in general, the reader is referred to Heidelberger [12].

2.2 Adaptive Importance Sampling and the Need for Smoothing

The challenge in importance sampling simulation is to find new transition probabilities \( \tilde{q} \) such that the variance of the estimator is significantly reduced. A practical
method for doing this is the cross-entropy method [13]. In this method, some (typically non-optimal) \( q_0 \) is initially chosen. Many sample paths are then generated under this measure. From the results, a new measure \( q_1 \) is calculated which minimizes a quantity known as the cross-entropy. Sample paths are then generated using \( q_1 \), from which \( q_2 \) is calculated, and so on. In many practical problems, the sequence \( q_j \) converges to a set of transition probabilities that provide a good estimator. The cross-entropy calculation, details of which are in [9], leads to the following expression for updating the transition probabilities:

\[
(q_{j+1})_{ik} = \frac{Q_{ik}}{\sum_l Q_{il}}
\]

(1)

with

\[
Q_{ik} = \frac{\sum_{Z_1 \ldots Z_N} L(Z) \sum_{x_{i+1} = x} L(Z, q_j) \delta_{x_{i+1}, x_i}}{\sum_{Z_1, \ldots, Z_N} L(Z) \sum_{x_{i+1} = x} 1},
\]

(2)

where \( Z_1, \ldots, Z_N \) denote \( N \) random sample paths obtained under the transition probabilities \( q_j \), \( x_i \) are the states on sample path \( Z \) and \( L(Z, q) \) denotes the likelihood ratio of sample path \( Z \) under transition probabilities \( q \).

Equation (2) means that \( Q_{ik} \) is the average of \( L(Z, q_j) \delta_{x_{i+1}, x_i} \) over all visits to state \( x \) on all sample paths which eventually reached the target. Then equation (1) simply ensures that the new transition probabilities are proportional to these averages. If the simulation was run under the original measure, the likelihood ratio \( L(\cdot, \cdot) \) is 1 and the interpretation is very straightforward: The new transition probabilities are simply proportional to the number of times each transition was observed on sample paths leading to the target. If the simulation was run under a modified measure \( q_j \), the likelihood ratio ensures a proper weighting to estimate these same probabilities. In principle, the division by the denominator in equation (2) could have been omitted since it cancels in equation (1). However, including the division has the advantage that \( Q_{ik} \) are sample averages, so it is easy to estimate their accuracy (variance).

Unfortunately, the values for the components of \( q_{j+1} \) as calculated above are usually not very accurate, because many states may only be visited rarely. De Boer [11] and de Boer and Nicola [9] discuss several techniques to improve these estimates, all exploiting the fact that the optimal transition probabilities out of neighbouring states tend to be close. These techniques are referred to as boundary layers, local averaging and smoothing, respectively. The former two improve the estimates of the transition probabilities out of a given state by also counting visits to neighbouring states. Effectively, this means that in equation (2), the sums over \( i : x_i = x \) would be extended to also include \( i \) for which \( x_i \) is in some sense ‘near’ \( x \).

In the case of boundary layers, this ‘nearness’ is in terms of distance from the state space boundary as illustrated in Figure 1; states that are far away from a boundary

Figure 1. Illustration showing which states are taken together when three boundary layers are used in a two-queue system, with \( x \) and \( y \) denoting the queue lengths

are combined. This is based on the empirical observation that transition probabilities tend to vary only close to the state space boundaries. Note that the number of boundary layers is typically chosen before the simulation is started, thus also reducing the number of distinct states for which information needs to be stored.

For local averaging, states are combined up to a specific distance from the state under consideration. This distance is chosen adaptively and separately for every state by increasing it until a variance estimate for equation (2) is sufficiently small. For more details, the reader is referred to de Boer [11] and de Boer and Nicola [9].

The idea of smoothing is to fit a nice smooth function to the results obtained using the above methods, thus reducing noise even further. Thus in practice, one iteration of the method consists of: (1) running the simulation under \( q_j \); (2) calculating \( Q_{ik} \), already taking into account boundary layers; (3) performing local averaging where needed; and (4) optionally applying smoothing. The result from the last step is the transition probability matrix \( q_{j+1} \) for the next iteration. For more details, the reader is referred to [9, 11, 14].

In the rest of this paper, we focus on the smoothing problem. Formally, this can be expressed as follows. We are given some numbers \( \gamma_{ik} \) (namely, the output of the local averaging step applied to \( Q_{ik} \)) and need to find numbers \( \tilde{q}_{ik} \) which are smooth functions of the system state \( x \), such that \( \sum_k \tilde{q}_{ik} = 1 \) for any \( x \).

### 3. The Proposed Smoothing Method

In this section, we will describe the proposed smoothing method in detail. We start by discussing the basics, namely local kernel regression, and illustrating this with an M/M/1 example. Within this class of smoothing methods, we choose to use the local linear regression and derive the equations suitable for obtaining normalized sets of transition probabilities. We then investigate how the kernel width (a parameter of the regression technique) can be

Volume 83, Number 12 | SIMULATION 813

---

*IMPROVING ADAPTIVE IMPORTANCE SAMPLING SIMULATION OF MARKOVIAN QUEUING MODELS*

---

**© 2007 Simulation Councils Inc.. All rights reserved. Not for commercial use or unauthorized distribution.**
chosen adaptively, and conclude with some remarks about practical issues.

3.1 Basics of Local Kernel Regression

A popular class of non-parametric regression models are local regression models based on kernels. In these methods, for every point of interest a polynomial is fitted to the data. This fit is chosen by minimizing a weighted average of the errors (i.e. deviations between data points and the polynomial), where the weights are given by a so-called kernel function. See Wand and Jones [15] for a detailed overview of such methods.

For each point (or state, in our application) \( \zeta_i \), the kernel function \( K(x - \zeta_i) \) attaches weights to all points (states) \( x \) based on their distance from \( \zeta_i \). (In actual fact, what we use here is not a true Kernel function, as it does not satisfy the requirement that \( \int K(z)dz = 1 \). However, due to the way we use the function here, this requirement is not necessary and would only result in extra unneeded computations.) A practical kernel function attaches greater weights to states that are close to \( \zeta_i \), of course. According to literature [15], the precise shape of the kernel function is not very important. We choose the very popular Gaussian kernel, which in the one-dimensional case is

\[
K(z) = e^{-z^2}.
\]

In order to adapt the kernel function to the problem at hand, it needs to have an extra parameter \( h \) that defines the bandwidth of the kernel by scaling the argument. Starting from our Gaussian kernel defined above, the Gaussian kernel function with width parameter \( h \) is

\[
K_h(z) = K(z/h) = e^{-((z/h))^2}.
\]

Increasing \( h \) makes the function ‘wider’, meaning that the weight assigned to states farther away increases.

After the kernel function, the parameter \( h \) and the degree \( r \) have been chosen, a polynomial weighted least-squares regression is executed. This means finding values of the parameters \( \beta_i \) which minimize the expression:

\[
\sum_{i=1}^{n} w_i \left( \beta_0 + \beta_1 (x_i - \zeta_i) + \beta_2 (x_i - \zeta_i)^2 + \ldots + \beta_r (x_i - \zeta_i)^r - y_i \right)^2,
\]

where \( y_i \) are the unsmoothed values (transition probability estimates in our application) at points (states, in our application) \( x_i \), and \( w_i = K_h(x_i - \zeta_i) \) are the weights. The smoothed value at point \( \zeta_i \) is \( \beta_0 \). In order to find the smoothed value at multiple points \( \zeta \) (namely, for every state), the procedure is repeated for each \( \zeta \) of interest.

Note that the above formula only applies to one-dimensional univariate problems, which we can use only for illustration in the M/M/1 queue. The expressions for more general situations are developed in Section 3.3.

3.2 Example: The M/M/1 Queue

Let us illustrate the above, and in particular the importance of choosing \( h \), with the simplest possible Jackson network: a single M/M/1 queue with arrival rate \( \lambda \) and departure rate \( \mu \). The probability of an arrival (in a non-empty state) is \( p = \lambda/(\lambda + \mu) \) and the probability of a departure is \( 1 - p \). We study the probability of the queue contents reaching 20 during a single busy cycle.

After simulating 1000 replications (i.e. entire busy cycles) of such a network (with \( p = 0.49 \)), we can apply equation (1) to calculate for each state the new arrival and departure probabilities. These arrival probabilities are plotted as dots in Figure 2. For the M/M/1 queue, the optimal state-dependent transition probabilities (i.e. those that would result in a zero-variance estimator) can be calculated according to de Boer [11, Appendix 8.A] and are shown in the figure as the dotted line.

As can be seen, the transition probability estimates resulting from the simulation are quite noisy. The smoothing problem is to find some sort of function that removes the noise. Ideally, this function should resemble the optimal function (the dotted line).

In our example, we performed a local linear (\( r = 1 \)) weighted least-squares regression for several values of \( h \). The results are also shown in Figure 2. Note that the regression was also computed at points between states, to obtain an accurate impression of what the resulting function looks like.

The choice of \( h \) influences the resulting function. For a small value such as \( h = 1.5 \), the function is still a bit noisy. For a large value such as \( h = 15 \), the resulting function is almost a straight line, as if a simple non-local non-weighted linear least-squares had been executed (which would indeed happen as \( h \) goes to infinity).

While \( h = 15 \) produces too much smoothing and \( h = 1.5 \) too little, \( h = 4 \) looks much better for this example: the result is a smooth function that still resembles the optimal function. It is somewhat positively biased, but that is to be expected by looking at the simulation results: most of them happen to be above the optimal function.

3.3 Generalization to Multiple Dimensions and Multiple Transition Types

The kernel function is easily extended to multiple dimensions by using the Euclidean distance between points. For two vectors \( x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \) and \( x_j = (x_{j1}, x_{j2}, \ldots, x_{jD}) \), the kernel function in \( D \) dimensions \( K_D \) is defined in terms of a one-dimensional kernel function \( K_h \):

\[
K_D(x_i - x_j) = \frac{K_h}{\sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{iD} - x_{jD})^2}}.
\]
A problem arises when we take different transition types into account. This problem was ignored in the previous example; for the two transition rates $p$ and $q$ we just investigated smoothing $p$ and set $q = 1 - p$ without actually using the simulation data for $q$. The desired behavior for multiple transition types is that the probabilities for all transition types have smoothing applied, but when smoothing is applied to all transition types individually, the smoothed transition probability estimates in a state may not sum to 1. This can be solved by introducing an extra constraint to the minimization.

For the local linear estimator (i.e. $r = 1$), the objective is to minimize the following sum w.r.t. $\beta$:

$$S = \sum_{i=1}^{n} \sum_{k=1}^{T} \left( y_{ik} - \beta_{0k} - \sum_{d=1}^{D} \beta_{dk} (x_{id} - \xi_d) \right)^2 w_i.$$ 

The smoothed probability estimates we look for here are $\beta_{0k}$ for $k = 1, \ldots, T$, where $T$ is the number of transition types. The constraint for ensuring that the smoothed probabilities of the different transition types sum to 1, is

$$\sum_{k=1}^{T} \beta_{0k} = 1. \quad (3)$$

The partial derivatives of $S$ with respect to $\beta_{dk}$ for $\delta = 0, \ldots, D$ can be calculated straightforwardly [14] and turn out to be linear in $\beta_{dk}$. According to the method of Lagrange multipliers, minimizing $S$ under the constraint then implies the following set of linear equations:

$$\frac{\partial S}{\partial \beta_{0k}} = \lambda \quad \text{for } k = 1, \ldots, T \quad \text{and}$$

$$\frac{\partial S}{\partial \beta_{dk}} = 0 \quad \text{for } \delta = 1, \ldots, D, \quad k = 1, \ldots, T.$$

Combined with the constraint (3) itself, this gives a set of $T(D + 1) + 1$ linear equations with the same number of unknowns, which can easily be solved. As in Section 3.1, this minimization needs to be repeated for every state $\xi$ for which one wishes to obtain the smoothed transition probabilities.

The above derivation is for local linear regression, i.e. $r = 1$. Repeating this derivation for higher-degree regression polynomials is in principle straightforward. However, the resulting number of linear equations is of the order $TD^r$ which quickly becomes unwieldy for larger dimensions $D$ and $r \geq 2$ (especially since solving them needs to be repeated for every state).

### 3.4 Choices

There are many possibilities of integrating the kernel smoothing method into the iterative cross-entropy framework. In particular, the following decisions need to be made:

- Should the local average calculation (see Section 2) be retained?
- What degree polynomial ($r$) should be used?
Should the weights of the states as calculated by the kernel function be multiplied by the number of times that particular state was visited? This would account for the fact that the transition probability estimate is more reliable in a state that has been visited frequently.

Limiting ourselves to the case of local constant \((r = 0)\) and local linear \((r = 1)\) regression, this results in eight possible combinations.

In order to judge them, an example model was simulated using all eight possible combinations. This example model consists of four queues in tandem, where the quantity of interest is the probability that the total population reaches 50 during a busy cycle. The arrival rate was 0.09 and the service rates 0.23, 0.227, 0.227 and 0.226. We used 5000 replications per iteration and 7 boundary layers.

Figure 3 shows how the relative error of the estimator changes in the course of the iterations for all eight combinations. It is clear from the figure that without local averaging the convergence is almost non-existent, so we should not omit this step. The four remaining combinations all converge, but the version with a degree-1 regression polynomial and no extra multiplication of the weights performs notably better than the other three versions. This version will therefore be used in the rest of this paper.

As well as the above comparison with local constant regression \((r = 0)\), there are other reasons to prefer linear regression, i.e. \(r = 1\), and not to try higher values of \(r\). Wand and Jones \[15, Section 5.4\] advise using either \(r = 1\) or \(r = 3\), as odd degree polynomials have better bias and boundary properties, and high degrees will rarely perform better, especially with the relatively small number of datapoints which we use. Furthermore, as we saw at the end of Section 3.3, the set of linear equations to be solved for degree \(r \geq 2\) becomes very large and thus computationally prohibitive.

### 3.5 Choosing the Bandwidth

Now that we have determined that a local linear regression method combined with the local average method works best, the question remains how to choose the bandwidth \(h\). Clearly, the optimal choice for \(h\) is related to the variance of the unsmoothed probability estimates equation (1): the larger this variance, the larger \(h\) should be. We investigate numerically for a few example cases how \(h\) should vary with that variance, with the aim of finding a formula expressing the optimal \(h\) in terms of that variance. We will indeed find such a formula, but also see that this is only a rough estimate; fortunately, it turns out that the algorithm is relatively insensitive to the precise choice of \(h\).

In the current article, we choose a single value for \(h\) for all states, based on the average variance of the unsmoothed transition probability estimates. Experiments with a per-state choice of \(h\) were not succesful; presumably, the per-state estimates of the variance of the unsmoothed values are not a sufficiently reliable basis for choosing \(h\).

In order to find the optimal choice of \(h\), we run the entire algorithm for several iterations. We do this once with a high number of replications \((10^7)\) without regression; this gives us a good estimate, \(q^*\), of the truly optimal transition probabilities. Next, we do this with a low number of replications with regression, and optimize \(h\) (using binary search) such that the smoothed transition probabilities are as close as possible to \(q^*\) in the sense of minimum sum of squared differences. (Actually, this method was also used to select \(h\) for the experiments in Section 3.4.) Thus, we find one datapoint: a value for the average variance of the
unsmoothed transition probability estimates and a corresponding optimal value for $h$. We repeat this experiment many times for four different models and with many different seeds for the random number generator. The result is the set of scatterplots in Figure 4, from which we may try to derive a relationship between the average variance and the optimal $h$.

To be more precise, the cases used for Figure 4 are as follows. Figure 4(a) concerns a two-node tandem network with $\lambda = 0.04$, $\mu_1 = \mu_2 = 0.48$. Each experiment consists of 6 iterations with 1000 replications followed by 6 iterations with 2000 replications; 40 different seeds were used. Figure 4(b) concerns a three-node tandem network with $\lambda = 1$, $\mu_1 = \mu_2 = \mu_3 = 20$, 30 iterations with 10 times 2000 replications, 10 times 4000 replications and finally 10 times 8000 replications, for 20 different seeds. Figure 4(d) concerns a four-node tandem network with $\lambda = 0.09$, $\mu_1 = 0.23$, $\mu_2 = \mu_3 = 0.227$ and $\mu_4 = 0.226$, for which 5 iterations with 10 000 replications were simulated for 18 different seeds. In all of these cases, 7 boundary layers were used. Additionally, the three-node experiment was also executed with 5 boundary layers, resulting in Figure 4(c). For all experiments, the target rare event was the total network population reaching 50 within a busy cycle.

From the graphs it is clear that with a larger average variance, $h$ should also be larger. However, there is a lot of spread in this optimum for a given variance, so a precise formula cannot be given. The solid line drawn in all four plots is given by

$$h_{opt}(x) = 10^{0.4D+0.15B-1} \times (\sigma^2(x))^{1/3}.$$ (4)

We do not claim that this line is in any sense an optimal fit to the scattered points; in fact, it was mostly chosen to fit to the rather dense and narrow cloud of points that is visible in the left half of all four plots.

Yang and Tschernig [16] show that for a $D$-dimensional system, the optimal $h$ in a point $x$ is of order

$$h_{opt}(x) \propto (\sigma^2(x))^{1/\nu+\epsilon}.$$ (5)
where \( \sigma^2(x) \) is the variance in point \( x \) and \( x \) denotes linear proportionality. This is clearly different from our results. This theoretical result suggests that the regression lines in the figure should be even less steep. We have no explanation where this difference comes from. It may have something to do with the application of boundary layers, which adds its own bias and changes the variance of the input values for the local regression method.

We choose \( h \) according to equation (4). Although it is not a very accurate result, it is the best we have. It was noted during experimentation that the sum of the squared differences of the optimal transition probabilities and the transition probability estimates after local regression was not very sensitive to the choice of \( h \). Using a \( h \) that is a factor 2 smaller or larger than the optimum results in only a few percent increase of the sum of squared errors. We therefore expect that even a relatively inaccurate formula such as equation (4) is sufficient for our purpose.

### 3.6 Practical Considerations

In this section, we briefly discuss a couple of implementation issues.

When a state is never visited during simulation, no input probability estimates \( y_{ik} \) can be calculated for the point corresponding to that particular state. Fortunately, there is an easy solution to this problem: leave out that particular point completely and the smoothing algorithm will calculate probability estimates from the surrounding points.

Another problem that can arise is that some of the transition probability estimates resulting from smoothing are very small, or even negative. To solve this, it was seen during experimentation that for an enabled transition type the transition probability is always larger than the minimum of the original transition probabilities. While we have not found a theoretical explanation for this, we use this property as a safeguard. Any transition probability estimate that is smaller than the minimum of the original transition probabilities is set equal to that minimum. The resulting transition probability estimates in that state are again normalized, so that the sum still equals 1.

Finally, when using boundary layers (see Section 2), a problem occurs with most interior states. For example, if we would use 5 boundary layers in the \( M/M/1 \) queue in Figure 2, there would be separate transition probability values for states \( 0 \) through \( 4 \), and a single transition probability for use in all of the states \( 5 \) through \( 19 \). It is unclear where (between \( 5 \) and \( 19 \)) on the horizontal axis this single point should be situated to calculate its distance to the other points and thus its weight. In order to avoid having to estimate the location of such points, we treat them separately, i.e. do not include them in the smoothing calculations. Smoothing in the \( M/M/1 \) queue with 5 boundary layers would only be done for states \( 1 \) through \( 4 \); recall that state \( 0 \) is also treated separately since a different set of transitions is enabled there. The extension of this principle to a two-queue system with again five boundary layers is illustrated in Figure 5, which indicates the sets of states to which smoothing is applied.

### 4. Experimental Results

To show the effectiveness of the proposed method, it was applied to four relatively large queueing networks. These four models and the simulation results are presented in Figure 6. The four networks are the following: five queues in tandem, a network with five queues and some random routing, another network with five queues and routing, and finally six queues in tandem. In all of these examples, the arrival and service rates (which are indicated in the figure) and routing probabilities (0.5 unless indicated otherwise in the figure) were chosen such that all queues are equally loaded, since it is well known that this is typically the most difficult case for importance sampling. For each of the four networks, we estimated the probability of the total population of the network exceeding some level (indicated in the figure) within a busy cycle.

The figures show for each model between 15 and 21 iterations of the method proposed in this paper. As a starting point for the first iteration, a state-independent change of measure was calculated using the method of Frater et al. [17]. All subsequent iterations used the change of measure calculated by our algorithms (including the smoothing technique) from the simulation results from the previous iteration. At one or more stages in the course of the iterations, a change to the system was made, e.g. a higher target overflow level or a larger number of replications per iteration. This is typically an efficient way of using the method: first obtain convergence for a lower overflow level, and then run an independent simulation at the target level.
Figure 6. Results for four different large queueing networks
level, and then use that as a starting point for a few final iterations to estimate the overflow probability of a higher level. Such changes are indicated using dotted lines and text in the figures.

The two curves in each figure show the overflow probability estimate (dashed line, right axis) and the corresponding relative error (solid line, left axis). The estimates themselves are normalized by dividing by the estimate obtained in the last iteration for that set of parameters, indicated as the ‘final estimate’ in the figures. Note that the samples taken at each iteration were used for two purposes: computing transition probabilities for the next iteration (using local average and smoothing) and for giving a (preliminary) estimate of the overflow probability and its relative error. The latter is also useful for judging the convergence from the graph.

It is clear from the figures that in all examples the method converged nicely to an estimate with a low relative error (typically around 1%) at a reasonable number of replications per iteration (usually $10^5$ up to level 50, and $10^6$ at level 100 but with a lower relative error).

The second example is exactly the same network that has previously been used as a benchmark [9, 10, 11]. By comparing the present results with those of de Boer [11], we can see the effectiveness of the new smoothing technique. The system now converges directly with level 50, whereas with the original smoothing technique intermediate iterations at level 20 were needed to help the system converge. Furthermore, the resulting relative error at level 50 and $10^5$ replications is about 4 times smaller.

5. Conclusions

A new smoothing method has been introduced for use in the adaptive rare-event simulation method described in de Boer [11] and de Boer and Nicola [9]. It has been shown experimentally that this new method improves the convergence and reduces the estimator variance. It does this by improving the estimates of the individual transition probabilities, by executing a locally weighted linear regression for every state.

A disadvantage of the local regression method compared to the older spline-fitting method is the execution time: the number of steps required for the local regression method is quadratic in the state space, and for larger networks with a high number of boundary layers this becomes noticeable. It may be possible to reduce the calculation time of the smoothing algorithm by using a bounded kernel, instead of the Gaussian kernel we used in this paper. Note however that another step in the method, namely the ‘local average’ step, has a worst-case behavior which is also quadratic in the state space, so adding the smoothing step does not change the order of the method’s complexity.

Although the method as developed in this paper works well, it does not exactly work according to the theory on local regression methods, specifically in the area of choosing the kernel width $h$. More research into this may result in better choices for the parameters, although due to the insensitivity to $h$ that has been observed it may be that not much improvement can be obtained.

6. References