Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

AN ALMOST REGENERATIVE SIMULATION OF VIDEO TELECONFERENCING TRAFFIC

David M. Cohen
Daniel P. Heyman

Bellcore
Morristown, NJ 07960, U.S.A.
dmc@bellcore.com
dph@bellcore.com

ABSTRACT

This note discusses the application of an *almost* regenerative method to the simulation of a Markov chain that takes a very long time to converge. Even simulation runs of 10^{12} cells did not produce the same results for simulation scenarios with an average cell-loss-rate of 10^{-8}. The regenerative method can not be used here as the time between regenerative states is too long (10^{30} events). Instead, an almost regenerative method was tried and it gave results that agreed with predictions from regenerative theory and with some very long simulation runs. The regenerative runs were done on a massively parallel processor.

Key words: regenerative simulation, parallel simulation, variance reduction.

1 INTRODUCTION

This note discusses experiments with an almost regenerative discrete event simulation of a Markov chain that takes a very long time to converge. Figure 1, on the next page, shows a block diagram of the simulation model. Access lines carrying Asynchronous Transfer Mode (ATM) cells generated by Variable Bit Rate (VBR) encoded video sources are fed into a buffer and multiplexed onto a higher speed output line. The study observed the cell loss resulting from buffer overflow. Different simulation scenarios were obtained by varying the number of access lines, the size of the buffer, the speed of the output line, and the parameters for the video sources. The results on engineering ATM networks are discussed in greater detail in Cohen and Heyman (1993a/b) and Cohen, Cooper, Heyman and Reilly (1992). This note discusses our experiments with an almost regenerative method.

The study was concerned with low loss probabilities, ones on the order of 10^{-8}. Simulation runs of the order of 10^{11} cells are normally regarded as sufficient to find a loss rate of 10^{-8}. However, even runs of 10^{12} cells did not always see the same cell-loss-rate. A reason is that the cell losses are clustered and neither the number of lost cells per cluster nor the time between cluster of loss are uniformly distributed (see Cohen and Heyman (1993a/b)).

A few thousand runs of 10^{10} cells were done for each scenario on a massively parallel processor, the MasPar MP-1216. Twenty additional runs of 10^{11} and 10^{12} cells were done on a workstation for some of the scenarios. The average cell-loss-rates observed by the few long runs and the average rates observed by the many shorter ones were of the same order of magnitude. While these simulations generated useful results about the distribution of losses, the observed variance in cell-loss-rates was high for scenarios with low cell-loss-rates. For example, the standard deviation in cell-loss-rates observed by different processors was five times the observed mean for scenarios with an observed mean of 10^{-8}. Obtaining estimates with a lower variance would either require doing much longer runs or "patching" together several runs to generate a longer run.

The regenerative method has been suggested by several authors, most notably Crane and Iglehart (1975a) and Fishman (1974), as a way to use many independent simulations to find a steady state probability. In a regenerative simulation, a regenerative state is identified and a regenerative cycle is defined to be a simulation run that starts and ends in the regenerative state. However, the regenerative method could not be used here because the time between regenerative states is too large. For example, with 14 sources, the mean time between visits to the state with the highest steady state probability of occurring is 7.25×10^{30} frames. Even at an extraordinary fast rate of 10^{13} frames a second, it would take 20 billion years to generate this many frames. Instead, an *almost* regenerative approach was tried. It gave results that had a much lower variance and which agreed

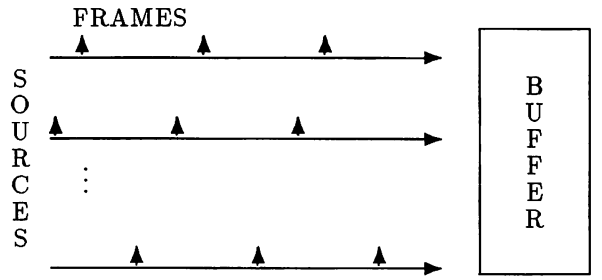

Figure 1: Block Diagram of Simulation Architecture

with the results of a few runs of 10^{13} cells.

The next three sections discuss the model used for the video sources, the basic simulation algorithm, and simulations done using a naive approach. Section 5 discusses the almost regenerative method and gives an illustrative application to a simpler problem, an M/M/1/k queue. Section 6 gives the details of the almost regenerative simulation of the ATM buffer. The appendix gives the details of the M/M/1/k example.

2 SIMULATION MODEL

The interframe period for the video sources used in this study is the PAL standard, which is 40ms. This means that once a video source starts transmitting, it sends a frame containing ATM cells every 40ms. The number of cells per frame is given by a Markov chain model for VBR video traffic that was developed in Heyman, Tabatabai, and Lakshman (1992). The Markov model is determined by three parameters: mean, variance, and the correlation factor between the number of cells in successive frames. Let f_k be the negative binomial probability:

$$f_k = \binom{k+r-1}{k} p^r q^k \qquad (1)$$

where $p = mean/variance$ and $r = mean \times p/(1-p)$. The transition matrix then has the discrete autoregression DAR(1) form:

$$M = \rho I + (1-\rho)Q \qquad (2)$$

where ρ is the correlation factor between successive frame sizes, I is the identity matrix, and each row of Q consists of the negative binomial probabilities $(f_0, \ldots, f_K, F_K)$ where $F_k = \Sigma_{k>K} f_k$ and K is the maximum number of cells per frame.

3 SIMULATION ALGORITHM

Although the study is interested in cell-loss-rate, the basic event in the simulation is the arrival of a video frame and not the arrival of an individual cell. Using frame arrivals instead of cell arrivals greatly reduces the number of events. The study assumes that cells are either front-loaded in the video frame, i.e. cells are sent at the access line speed until the frame is complete, or that cells are evenly distributed in the frame. Under either assumption, the buffer overflow in any time period can be computed from the frame arrival times, the number of cells per frame, and the buffer's initial state.

The frame arrival times are completely determined by the arrival time of the call's initial frame since each active source sends a new frame every 40ms. By comparing the number of active lines with the ratio of access to output line speeds, the simulation can identify intervals when the combined input rate is less than the output rate. If the buffer is empty at the start of such an interval, no cells will be lost during the interval and the buffer will be empty at the end of the interval. The simulation can then avoid doing several calculations. This optimization can be very helpful as the buffer is frequently empty in scenarios with low cell-loss-rates.

Two different implementations were done for the MasPar MP-1216, a SIMD parallel processor. The most expensive parts of the computation cycle are random number generation and maintenance of a priority queue of frame completion times (needed when the cells are front loaded in the video frames). The first parallelization distributes an individual simulation across a row of processors. Each processor is responsible for generating a random number and for maintaining one slot in the queue. A new item is added to the queue by broadcasting it along the row to all processors forming the queue. Each processor determines if it should replace the queue item in its slot in the queue. An item is deleted from the head of the queue by having each processor shift its value to the left. This gives a constant time for updating a priority queue of fixed maximum size. However, the number of items in the priority queue in scenarios with very low loss probabilities is frequently below the crossover point between the parallel and non-parallel implementation.

An alternative is to do many independent replications in parallel. A drawback is that if one processor in a SIMD machine is executing a statement then the other processors are either idle or are executing that statement. Thus, if one processor needs to compute the exact number of cells that arrive in a time interval, all processors are delayed until that computation is done. This defeats the optimization mentioned above and slows down the simulation considerably. Even without this optimization, the replicated

Table 1: Non-regenerative Method

run	*mean*	*std*
1	1.3040×10^{-8}	6.883×10^{-8}
2	1.3173×10^{-8}	7.169×10^{-8}
3	1.2619×10^{-8}	7.289×10^{-8}
4	1.3122×10^{-8}	7.224×10^{-8}
5	1.3968×10^{-8}	8.026×10^{-8}
6	1.3146×10^{-8}	7.251×10^{-8}
all	1.3178×10^{-8}	7.313×10^{-8}

simulation approach was found to generate more cells per computation time than parallelizing each individual simulation. This is in accord with the results of Heidelberger (1986) in analyzing the efficiency of the replicated approach for estimating steady-state quantities.

4 NAIVE SIMULATION OF THE ATM BUFFER

Several thousand independent replications were done in parallel for each simulation scenario. Each individual replication ran for 8×10^7 frame arrivals, which gives about 1.04×10^{10} cells per replication. The mean and standard deviation of the cell-loss-rates observed by the different replications of each scenario were computed. Table 1 shows the results from 6 sets of 16K replications of a scenario with 14 access lines, a DS-3 rate output line, and a video source with an average of 130 cells per video frame. The cell-loss-rates observed by the individual processors were not normally distributed; 89 % of the replications experienced no cell loss. As shown in the table, the standard deviation is 5 times the mean.

5 ALMOST REGENERATIVE METHOD

The regenerative method has been suggested as a way to estimate steady state probabilities from a single simulation run. In a regenerative simulation, a regenerative state is identified and a regenerative cycle is defined to be any portion of simulation run that starts and ends in the regenerative state. The estimator for the loss probability is

$$\hat{L} = \frac{\hat{E}(\text{number-of-losses-in-a-cycle})}{\hat{E}(\text{number-of-cells-arriving-in-a-cycle})} \qquad (3)$$

where $\hat{E}$ indicates that the expectation is estimated from the data in the simulation. See, e.g. section 6 in Rubinstein (1981) for an extensive discussion.

The regenerative method could not be used here because the time between regenerative states is too large. For example, the state in the video source model that has the largest probability of occurring has a steady-state probability of 0.006247. For a simulation scenario with 14 access lines, the mean time between visits to the situation where every access line is in that state is $.006247^{-14} = 7.25 \times 10^{30}$ frames. Even at a rate of 10^{13} frames a second, it would take 20 billion years to simulate this many frame arrivals.

The probability of a source generating less than 100 cells/frame is .404. Thus, the mean time between visits to the situation where the state of every access line is less than 100 is $.404^{-14} = 3.22 \times 10^5$ video frames. This is only 920 sec or 5.85×10^8 cells of real time. It takes 2 minutes for each of the processors on the MasPar to simulate this many frames. While the regenerative method is not feasible, an *almost* regenerative approach is.

Because of the difficulty in finding regenerative events that occur frequently enough, the almost regenerative method was proposed by Crane and Iglehart (1975b) and followed up by Gunther and Wolff (1980). The examples in these papers are the M/M/1 queue in the former and a pair of M/M/c queues in tandem in the latter. The processes considered were continuous-time, so an almost regenerative event is defined in the following way. Partition the state space into disjoint sets, U and V say. An almost regenerative event occurs when a transition from some state $u \in U$ to some other state $v \in V$ occurs.

Our process is discrete-time and our almost regenerative event is the entrance to some set of states, W say. This raises the following issue. In discrete time, when a regenerative event occurs, the next occurrence is the very next epoch when the regenerative event occurs. This means that a transition from the regenerative state to itself can occur. Shall we require an almost regenerative event to leave W, as the continuous-time definition requires, or shall we allow transitions from state $w \in W$ to state $w\prime \in W$ to form a cycle?

Denote these alternatives by method I and method II respectively. We will seek guidance by analytically analyzing the effects of choosing method I or method II on simulations of the M/M/1/k queue. The purpose of the simulation is to estimate the loss probability. In the Appendix it is shown that when $W = \{0, 1, \ldots, b-1\}$ and entrance into W is treated as a regenerative event, method II provides the exact solution and method I does not. This means that method II produces an asymptotically-unbiased estimate and method I does not.

6 ALMOST REGENERATIVE SIMULATION OF THE ATM BUFFER

The almost regenerative set $S(W)$ in our experiments is defined to be the set of states where the buffer is empty and the current state of each access line is less than a watermark W, i.e. each access line sent W or fewer cells in its last input frame. The first condition is satisfied by most states in a simulation since the buffer is almost always empty for low loss rates. To be a truly regenerative state, all of the states with input less than W should have the same outgoing paths. This is not the case here. However, the transition matrix M in equation 2 has the discrete autoregression DAR(1) form $\rho I + (1-\rho)Q$ where every row of Q is identical. Since every row of Q is identical, when a line changes state, the new state is independent of the original state. Since the buffer overflow events are rare and have long periods between them, a plausible assumption is that every line will change before the next overflow condition. Thus, the almost regenerative approach is a plausible approximation.

The simulation model is the same as shown in Table 1, namely 14 access lines, a 436 cell buffer, a DS-3 rate output line, and a video source with mean of about 130 cells per frame. The watermark W was chosen to be 105, so the regenerative set consisted of all states where the buffer is empty and each access line is sending less than 105 cells per frame. In the experiments reported here, a transition from one state in the regenerative set to another state also in the regenerative set is considered to be a complete regenerative cycle. This is consistent with the analysis of the M/M/1/k queue in the previous section. Bias can arise when doing multiple independent simulations in parallel if the completion time of the simulation on one processor effects the completion time of the other simulations. Heidelberger (1988) and Heidelberger and Glynn (1991) have given some estimators that are unbiased as the number of processors or the simulation length goes to infinity. We tried two of them.

For the first estimator, after a processor has run for a fixed time T, it is allowed to complete its current regenerative cycle and then stops. This corresponds to Heidelberger's μ_2 (1988). Let $loss_i$ be the number of cells lost in the simulation on processor i and $total_i$ be the total number of cells on the processor. Then the μ_2 estimator for the simulations run on a set of processors P is given by:

$$\mu_2(\mathsf{P}) = \sum_{i \in \mathsf{P}} (loss_i) \Big/ \sum_{i \in \mathsf{P}} (total_i) \qquad (4)$$

For the other estimator, only the processors that have not completed a cycle by time T are allowed to continue. The other simulations stop and the data from their unfinished cycles is discarded. This corresponds to the μ_3 estimator from Heidelberger (1988). If n_i is the number of cycles done on processor i, then μ_3 for the set P is given by:

$$\mu_3(\mathsf{P}) = \sum_{i \in \mathsf{P}} (loss_i/n_i) \Big/ \sum_{i \in \mathsf{P}} (total_i/n_i) \qquad (5)$$

Table 2: μ_2 Estimator using Almost Regenerative Simulation with T = 8×10^7 frames

P	*exp 1*		*all*	
	mean $\times 10^8$	*std* $\times 10^8$	*mean* $\times 10^8$	*std* $\times 10^8$
128	1.3476	0.585	1.3195	0.590
1024	1.3482	0.179	1.3196	0.205
2048	1.3483	0.168	1.3197	0.158
4096	1.3485	0.138	1.3197	0.123
8192	1.3485	0.547	1.3197	0.735
16384	1.3484	-	1.3197	0.541

Table 3: μ_3 Estimator using Almost Regenerative Method with $T = 8 \times 10^7$ frames

P	*exp 1*		*all*	
	mean $\times 10^8$	*std* $\times 10^8$	*mean* $\times 10^8$	*std* $\times 10^8$
128	1.215	0.800	1.271	0.909
1024	1.203	0.276	1.259	0.337
2048	1.201	0.229	1.256	0.255
16384	1.197	-	1.253	0.120

Tables 2 and 3 show results obtained using T equal to the time used in the Table 1, namely 8×10^7 frame arrivals. The tables show the values of the μ_2 and μ_3 estimator obtained from 6 sets of 16K processors. Each set of 16384 simulations can be partitioned into N sets of $P = 16394/N$ processors each. Partitioning the processors into sets of 128 processors each gives 128 different estimates of the cell-loss-rate. Each group of 128 processors has a total of 10^{12} cells. Partitioning into sets of 1024 processors each gives 16 estimates each with 10^{13} cells. The column in Table 2 labeled *exp1* shows the results of the first set of 16K simulations. The column labeled *all* shows the combined results of the six sets. The Central Limit Theorem says that as the number of processors in the grouping P increases to infinity, the distribution of the estimates $\mu_2(P)$ converge to a normal distribution. Figure 2 shows the QQ-plot (see Becker, Cham-

bers and Wilks (1988)) for the estimates of μ_2 given by dividing the processors into groups of 2048. It is normally distributed. Figure 3 shows the corresponding plot for the μ_3 estimator and again it is close to normal.

Comparing Tables 2 and 3 with Table 1, we observe that, no matter how the processors are grouped, the variance given by even only one run of 16K simulations done using the almost regenerative method is an order of magnitude lower than the variance given by $6 \times 16K$ simulations using the naive method. Also, the mean given by the almost regenerative method is stable. Doing additional sets of 16K runs reduced the variance but did not change the mean substantially. The variance given by both the μ_2 and μ_3 estimators decreases as the number of processors per group increases. The mean given by the μ_2 estimator is stable while the mean given by the μ_3 estimator changes slightly.

Results for the μ_2 estimator obtained with different values for the stopping time T are shown in Table 4. Reducing T from 8×10^7 to 8×10^6 events had little effect on the mean but doubled the standard deviation. Reducing T from 8×10^6 to 1000 events changed the mean and variance only slightly. Reducing T further to 10 events changed the mean slightly but increased the variance by an order of magnitude. The standard deviation when $T = 10$ is only slightly less than the mean for $P = 2048$ and is about 2/3 of the mean for $P = 4096$. This is still much less than the standard deviation for the naive method. The variance jumps when $T = 10$ as most processors stop their simulations after 10 events without ever leaving the regenerative set. They contribute nothing to the numerator and little to the denominator of the μ_2 estimate. This effectively decreases the number of processors producing the estimate. Table 5 shows the μ_2 estimates given by $T = 1000$ and $T = 10$ for varying values of P. As P increases the variance declines which is in accord with regenerative theory.

Table 6 shows the effect that varying the stopping time has on total simulation time. It also shows the time for 97% and 99.7 % of the simulations to complete. After most of the processors have finished their simulations, it is easy and economical to transfer the remaining simulations and finish them on a workstation. Reducing the time T from 8×10^7 to 8×10^6 events gave only a 25% reduction in the time until the last processor was finished, but almost a 50 % reduction in the time until 99.7 % of the processors were finished. The reduction from 8×10^6 events to 1000 events had only a 5 % to 15 % effect on the simulation time. The further reduction to 10 events had a much greater effect on running time.

Table 4: μ_2 Estimator with Variable Stopping Time T

T	*P = 2048*		*P = 4096*	
	mean $\times 10^8$	*std* $\times 10^8$	*mean* $\times 10^8$	*std* $\times 10^8$
8.0×10^7	1.320	0.1583	1.320	0.1233
4.8×10^7	1.295	0.1554	1.295	0.1192
1.6×10^7	1.291	0.2702	1.285	0.1689
0.8×10^7	1.293	0.3519	1.293	0.2482
1000	1.2436	0.313	1.2438	0.232
10	1.2044	1.086	1.2032	0.801

Table 5: μ_2 Estimator using T = 1000 and T = 10 Events

P	*T = 1000*		*T = 10*	
	mean $\times 10^8$	*std* $\times 10^8$	*mean* $\times 10^8$	*std* $\times 10^8$
1024	1.2438	0.466	1.2047	1.510
2048	1.2436	0.313	1.2044	1.086
4096	1.2438	0.232	1.2032	0.801
8192	1.2961	0.143	1.2041	0.685
16384	1.2433	0.122	1.2078	0.352

Table 6: Simulation Run Length for μ_2 with Variable Stopping Time T

T	*simulation run length number of events* $\times 10^{-7}$		
	100%	99.7%	97%
8.0×10^7	22.32	16.32	12.96
4.8×10^7	21.28	13.20	9.84
1.6×10^7	17.20	9.92	6.56
0.8×10^7	16.48	9.12	5.76
1000	15.76	8.32	4.96
10	10.16	4.88	1.52

Table 7: Cells and Cycles per 2048 Processors with Varying Stopping Time

T	*cells*		*cycles*	
	mean $\times 10^{12}$	*var* $\times 10^{12}$	*mean* $\times 10^{5}$	*var* $\times 10^{5}$
8.0×10^7	25.15	0.075	16.56	0.180
4.8×10^7	16.62	0.078	10.97	0.164
1.6×10^7	8.04	0.085	5.33	0.102
0.8×10^7	5.92	0.091	3.92	0.081
1000	3.79	0.097	2.50	0.059
10	0.33	0.042	0.216	.00076

Table 7 shows the number of cells and cycles per group of 2048 processors.

We did runs of 2.2×10^{13} cells, 4.5×10^{13} cells, and 7.4×10^{13} cells. The observed cell-loss-rates were 1.49×10^{-8}, 1.02×10^{-8} and 1.31×10^{-8} respectively. These agree with the results from the regenerative experiments. We divided the long workstation runs into 135 segments of 1.04×10^{12} cells and plotted the distribution of cell-loss-rates for the individual segments. This distribution was not normal which indicates that using normal sampling theory with the method of batch means is inappropriate here.

7 CONCLUSIONS

The Markov chain simulated in these experiments takes a very long time to converge to steady-state. For a scenario with an average cell-loss-rate of 10^{-8}, even runs of 10^{12} cells did not always produce the same cell-loss-rate. The regenerative approach was not applicable because the time between regenerative states is too long. Instead, we tried an almost regenerative method and showed that it gave results that agreed with predictions from regenerative theory and with naive runs that are 1000 times longer.

APPENDIX: ALMOST REGENERATIVE ANALYSIS OF AN M/M/1/k Queue

Let π_i be the steady-state probability that i customers are present. It is well-known that

$$\pi_i(k) = \pi_i = \frac{(1-\rho)\rho^i}{1-\rho^{k+1}}, i = 0, 1, \ldots k. \qquad \text{(A-1)}$$

Since the arrivals are Poisson, π_i is also the probability that an arriving customer finds i customers present, and so π_k is the loss probability.

Let $W = \{0, 1, \ldots b-1\}$, $(b < k)$ be the almost regenerative set of states, $\bar{L}_b$ be the expected number of losses in a cycle, and $\bar{C}_b$ be the expected number of arrivals in a cycle. The almost regenerative estimate of π_k, $\hat{\pi_k}$ say, is

$$\hat{\pi_k} = \frac{\bar{L}_b}{\bar{C}_b} \qquad \text{(A-2)}$$

Method II

Towards obtaining formulas for the quantities on the right-side of (A-1), let $L_i(j)$ be the expected number of losses in an interval that starts when an arrival finds i in the system and ends when the number in the system is no larger than j, $i >= j$, and $c_i(j)$ be the mean length of that interval. By examining sample paths it is easy to see that $L_0(k) = L_1(k+1) = L_2(k+2) = \ldots$, and so

$$L_0(k) = L_b(k+b), \quad b >= 0. \qquad \text{(A-3)}$$

The regenerative theorem asserts that

$$\pi_k = \frac{L_0(k)}{c_0(k)}$$

and

$$\pi_0 = \frac{1}{c_0(k)},$$

so $L_0(k) = \pi_k/\pi_0 = \rho^k$ and (A-2) yields

$$L_{b-1}(k) = \rho^{k-b+1}. \qquad \text{(A-4)}$$

Under method I, a cycle that starts in state $i < b-1$ will have no losses. The probability that a cycle starts in state $b-1$, s_{b-1} say, is

$$s_{b-1} = \frac{\pi_{b-1}}{\sum_0^{b-1} \pi_j}.$$

Thus, (5) and (A-3) yield

$$\bar{L}_b = s_{b-1} L_{b-1}(k) = \frac{(1-\rho)\rho^k}{1-\rho^b}. \qquad \text{(A-5)}$$

The same sample-path argument shows that $c_0(k) = c_1(k+1) = c_2(k+2) = \ldots$, so

$$c_{b-1}(k) = c_0(k-b-1) = \frac{1}{\pi_0} = \frac{1-\rho^{k-b}}{1-\rho}. \qquad \text{(A-6)}$$

Now

$$\bar{C}_b = (1 - s_{b-1}) + s_{b-1} c_{b-1}(k) = \frac{1-\rho^{k+1}}{1-\rho^b}; \qquad \text{(A-7)}$$

substituting (A-4) and (A-6) into (A-1) and comparing with (5) yields $\hat{\pi_k} = \pi_k$ for all $b < k$.

Method I

The mean number of losses in a cycle is $L_{b-1}(k) = \rho^{k-b-1}$ because every cycle starts when a arrival finds $b-1$ customers present. The mean length of a cycle is obtained as follows. Let m_i be the mean first-passage-time from state i to state $b-1$. The probability that a cycle ends in state i is given by

$$s_i = \frac{\pi_i}{\sum_0^{b-1} \pi_j}.$$

The mean length of a cycle is

$$\bar{C}_b = \sum_{i=0}^{b-2} s_i m_i + c_{b-1}(k).$$

The m_i satisfy a difference equation; the solution is very messy and will not be given here. It suffices to note that when $b = 2$, m_0 is the reciprocal of the probability that an interarrival time is less than a service time, which is $(1+\rho)/\rho$. Hence

$$\bar{C}_b = 1 + \frac{1-\rho_k}{1-\rho}$$

and so (A-1) gives

$$\hat{\pi_k} = \frac{\rho^{k-1}(1-\rho)}{2-\rho-\rho^k}$$

which does not equal π_k.

REFERENCES

Becker, R. A., J. M. Chambers, and A. R. Wilks. 1988. *The New S Language: A Programming Environment for Data Analysis and Graphics*, Wadsworth & Brooks, Pacific Grove, Ca, 1988.

Cohen, D. M. and D. P. Heyman. 1993a. "A Simulation Study of Video Teleconferencing Traffic in ATM Networks," *Proceedings IEEE Infocom 93*, M. G. Hluchyj and K. S. Vastola, Eds, IEEE Press, Los Alamitos CA, pp 894 - 901.

Cohen, D. M., and D. P. Heyman. 1993b. "Performance Modeling of Video Teleconferencing in ATM Networks," to appear in *IEEE Transactions on Circuits and Systems for Video Technology.*

Cohen, D. M, C. A. Cooper, D. P. Heyman, and A. K. Reilly. (1992). "A Study of Cell Loss Characteristics for Selected ATM Traffic Types," ANSI T1 standards contribution, T1A1.3/92-083

Crane, M. A. and D. L. Iglehart. 1975a. "Simulating Stable Stochastic Systems: III. Regenerative Processes and Discrete-Event Simulations," *Operations Research*, 23:33 - 45.

Crane, M. A., and D. L. Iglehart. 1975b. "Simulating Stable Stochastic Systems, IV: Approximation Techniques," *Management Science*, 21:1215-1224.

Fishman, G. S. 1974. "Estimation in Multiserver Queuing Simulation," *Operations Research*, 22:72 - 78.

Glynn, P. W. and P. Heidelberger. 1991. "Analysis of Parallel Replicated Simulation Under a Completion Time Constraint," *ACM Transactions of Modeling and Computer Simulation*, 1:2 - 24.

Gunther, F. L, and R. W. Wolff. 1980. "The Almost Regenerative Method for Stochastic System Simulations," *Operations Research*, 28:375 - 387

Heidelberger, P. 1986. "Statistical Analysis of Parallel Simulations," *1986 Winter Simulation Conference Proceedings*, J. Wilson and J. Henriksen, Eds, IEEE Press, New York, 290-295.

Heidelberger, P. 1988. "Discrete Event Simulations and parallel processing: Statistical properties," *SIAM J. Sci. Stat. Computing*, 9:1114 - 1132.

Heyman, D. P., A. Tabatabai, and T. V. Lakshman. 1992. "Statistical Analysis and Simulation Study of of Video Teleconferencing Traffic in ATM Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, 2:49 - 59.

Rubinstein, R. Y. 1981. *Simulation and the Monte Carlo Method*, Wiley, Chichester.

AUTHOR BIOGRAPHIES

DANIEL P. HEYMAN did his undergraduate work in industrial and electrical engineering at Rensselaer Polytechnic Institute (1960). He received an M.I.E. degree from Syracuse University (1962) and the Ph.D. in operations research from the University of California- Berkeley in 1966, and then joined Bell Laboratories. He transferred to Bellcore at the epoch of the AT&T divestiture. His research areas are numerical analysis of stochastic processes, queueing theory, and performance models of data communications systems.

DAVID M. COHEN did his undergraduate work in mathematics at Harvard University (1972). He received a Ph.D. in mathematics from the Massachusetts Institute of Technology in 1976 and joined Bell Laboratories in 1981. He transferred to Bellcore at divestiture. His current research interests are in simulation and software engineering.

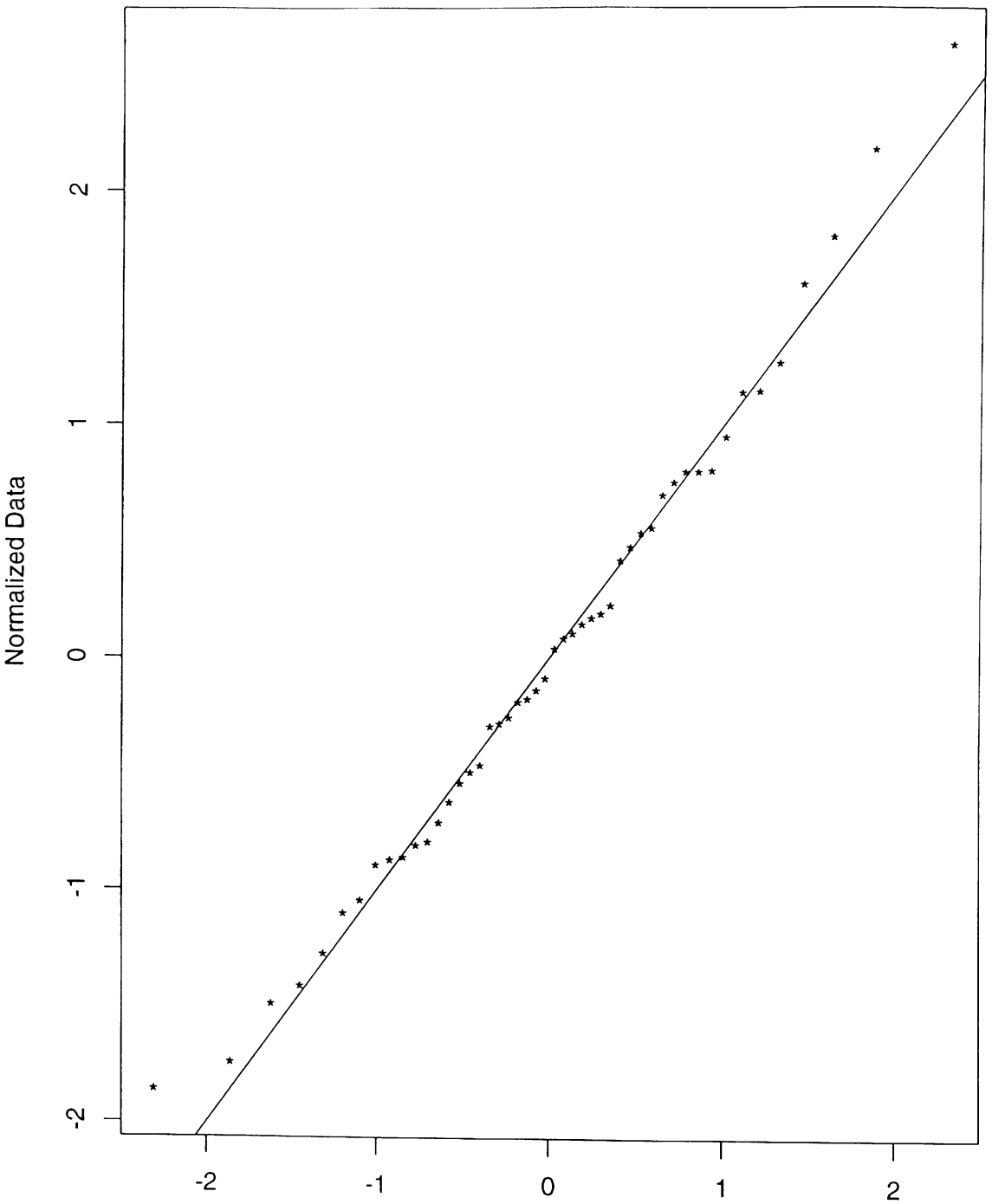

Figure 2: QQ-plot of mu2 Estimator

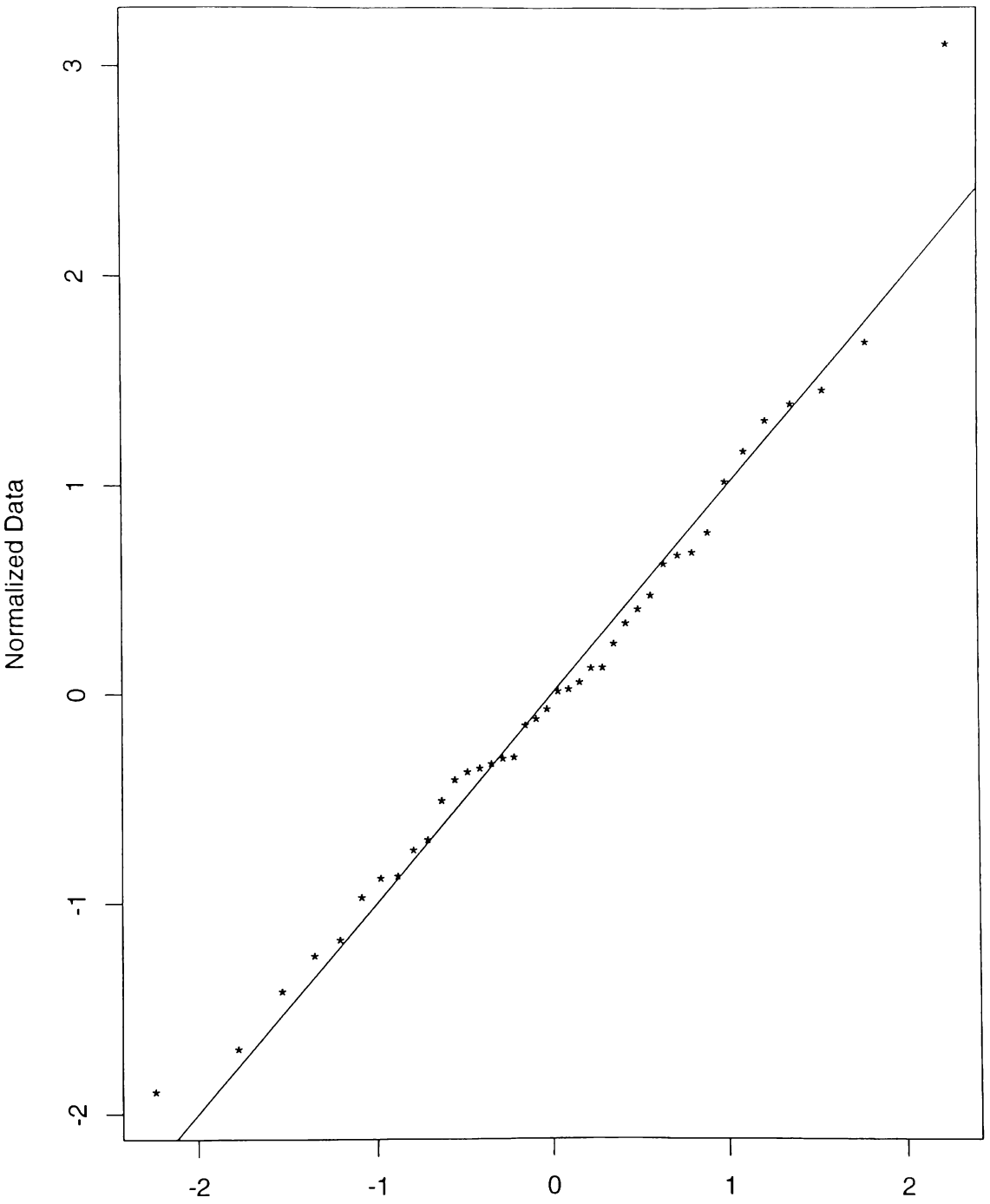

Figure 3: QQ-plot of mu3 Estimator