EFFICIENT PARALLEL ALGORITHMS FOR TANDEM QUEUEING SYSTEM SIMULATION

Sergei M. Ermakov* and Nikolai K. Krivulin
(Faculty of Mathematics and Mechanics, St.Petersburg State University
Bibliotechnaya sq.2, Petrodvorets, 198904 St.Petersburg, Russia)

1. Abstract

Parallel algorithms designed for simulation and performance evaluation of single-server tandem queueing systems with both infinite and finite buffers are presented. The algorithms exploit a simple computational procedure based on recursive equations as a representation of system dynamics. A brief analysis of the performance of the algorithms are given to show that they involve low time and memory requirements.

2. Introduction

The simulation of a queueing system is normally an iterative process which involves generation of random variables associated with current events in the system, and evaluation of the system state variables when new events occur [4, 1, 3]. In a system being simulated the random variables may represent the interarrival and service time of customers, whereas, as state variables, the arrival and departure time of customers, and the service initiation and completion time can be considered.

The usual way to represent dynamics of queueing systems as well as their performance criteria relies on recursive equations describing evolution of system state variables [1, 3, 8, 2, 9]. Since the recursive equations actually determine a global structure of changes in the system state variables consecutively, they can serve as a basis for the development of efficient simulation algorithms [1, 3, 5, 9].

In this paper, we assume as in [3, 5, 9] that appropriate realizations of the random variables involved in simulation are available when required, and we therefore concentrate only on deterministic parallel algorithms of evaluating the system state variables from these realizations. Methods and algorithms of generating random variables and their analysis can be found in [4]. A thorough investigation of parallel simulation from the viewpoint of statistics is given in [6].

We present parallel algorithms designed for simulation and performance evaluation of open single-server tandem queueing systems with both infinite and finite buffers. The algorithms are based on a simple computational procedure which exploits a particular order of evaluating the system state variables from the related recursive equations, and they are intended for implementation on either a vector processor or single instruction, multiple data (SIMD) parallel processors [10]. The analysis of their performance shows that the algorithms involve low time and memory requirements.

In Section 3, we give recursive equations which describe the dynamics of tandem systems with both infinite and finite buffers. Furthermore, tandem system performance criteria are represented in terms of state variables involved in the recursive equations. In Section 4, parallel simulation algorithms are presented and their performance is discussed. A brief conclusion is given in Section 5.

3. Models of Tandem Queues

In this section we consider recursive equation based models of tandem queues, and give related representation of system performance measures. We start with a simple model of a single-server tandem queueing system with infinite buffers, and then extend it to more complicated models of systems with finite buffers, in which servers may be blocked according to some blocking rule.

(1) Tandem Queues with Infinite Buffers

Consider a series of N single-server queues with infinite buffers, depicted in Fig. 1. An additional queue labelled with 0 is included in the model to represent the external arrival stream of customers.

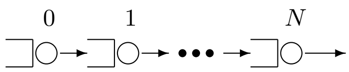

Fig. 1 Tandem queues with infinite buffers.

Each customer that arrives into the system is initially placed in the buffer at the 1st server and then has to pass

*The research described in this publication was made possible in part by Grant #NVZ000 from the International Science Foundation.

through all the queues one after the other. Upon the completion of his service at server n, the customer is instantaneously transferred to queue $n+1$, $n = 1, \ldots, N-1$, and occupies the $(n+1)$st server provided that it is free. If the customer finds this server busy, he is placed in its buffer and has to wait until the service of all his predecessors is completed.

For each queue n in the system, $n = 0, 1, \ldots, N$, we introduce the following notations:

A_n^k, the kth arrival epoch to the queue;

B_n^k, the kth service initiation time at the queue;

C_n^k, the kth service completion time at the queue;

D_n^k, the kth departure epoch from the queue.

Furthermore, let us denote the time between the arrivals of kth customer and his predecessor to the system by τ_0^k, and the service time of the kth customer at server n by τ_n^k, $n = 1, \ldots, N$, $k = 1, 2, \ldots$. We assume that $\tau_n^k \geq 0$ are given parameters, whereas A_n^k, B_n^k, C_n^k, and D_n^k present unknown state variables. Finally, for each $n = 0, \ldots, N$, we define $D_n^k \equiv 0$ for all $k \leq 0$, and $D_{-1}^k \equiv 0$ for all $k = 1, 2, \ldots$.

With the condition that the system starts operating at time zero, and its servers are free of customers at the initial time, the state variables in the model can be related by the equations [1, 3, 2, 9]

$$\begin{aligned} A_n^k &= D_{n-1}^k, \\ B_n^k &= A_n^k \vee D_n^{k-1}, \\ C_n^k &= B_n^k + \tau_n^k, \\ D_n^k &= C_n^k, \end{aligned}$$

where the symbol $\vee$ stands for the maximum operator, $n = 0, 1, \ldots, N$, $k = 1, 2, \ldots$. Clearly, the above set of recursive equations may be reduced to two equations

$$B_n^k = D_{n-1}^k \vee D_n^{k-1}, \quad (1)$$
$$D_n^k = B_n^k + \tau_n^k, \quad (2)$$

and even to the equation

$$D_n^k = (D_{n-1}^k \vee D_n^{k-1}) + \tau_n^k, \quad (3)$$

which will provide the basic representations for simulation algorithms in the next sections.

(2) Tandem Queues with Finite Buffers

Suppose now that the buffers of servers in the open tandem system have finite capacity. Furthermore, we assume that the servers may be blocked according to some blocking rule. In this paper, we restrict our consideration to *manufacturing* blocking and *communication* blocking which are most commonly encountered in practice [1, 3, 2].

Let us consider an open tandem system of N queues (Fig. 2), and assume the buffer at the nth server, $n = 1, \ldots, N$, to be of the capacity m_n, $0 < m_n < \infty$.

Manufacturing Blocking. First we suppose that the dynamics of the system follows the manufacturing blocking rule. Under this type of blocking, if upon completion

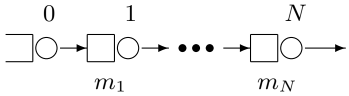

Fig. 2 Tandem queues with finite buffers.

of a service, the nth server sees the buffer of the $(n+1)$st server full, it cannot be unoccupied and has to be busy until the $(n+1)$st server completes its current service to provide a free space in its buffer. Clearly, since the customers leave the system upon their service completion at the Nth server, this server cannot be blocked.

With the additional condition that $D_n^k \equiv 0$ if $n > N$, one can describe the dynamics of the system by the equations [1, 3, 2, 9]

$$B_n^k = D_{n-1}^k \vee D_n^{k-1}, \quad (4)$$
$$C_n^k = B_n^k + \tau_n^k, \quad (5)$$
$$D_n^k = C_n^k \vee D_{n+1}^{k-m_{n+1}-1}. \quad (6)$$

Communication Blocking. This rule does not permit a server to initiate service of a customer if the buffer of the next server is full. In that case, the server remains unavailable until the current service at the next server is completed.

Let us assume that the system depicted in Fig. 2 follows communication blocking, and introduce the notation H_n^k to denote the time instant at which the nth server becomes ready to check whether there is empty space at the buffer of the next server, and to initiate service of customer k if it is possible. Now the system dynamics may be represented by the equations [1, 2, 9]

$$H_n^k = D_{n-1}^k \vee D_n^{k-1}, \quad (7)$$
$$B_n^k = H_n^k \vee D_{n+1}^{k-m_{n+1}-1}, \quad (8)$$
$$D_n^k = B_n^k + \tau_n^k. \quad (9)$$

(3) Representation of System Performance

Suppose that we observe the system until the Kth service completion at server n, $1 \leq n \leq N$. As is customary in queueing system simulation, we assume that $K > N$. The following average quantities are normally considered as performance criteria for server n in the observation period [1, 7, 8, 9]:

system time of one customer: $S_n = \sum_{k=1}^K (D_n^k - A_n^k)/K$,

waiting time of one customer: $W_n = \sum_{k=1}^K (B_n^k - A_n^k)/K$,

throughput rate of the server: $T_n = K/D_n^K$,

utilization of the server: $U_n = \sum_{k=1}^K \tau_n^k / D_n^K$, number of customers: $J_n = \sum_{k=1}^{K}(D_n^k - A_n^k)/D_n^K,$

queue length at the server: $Q_n = \sum_{k=1}^{K}(B_n^k - A_n^k)/D_n^K.$

Clearly, the above criteria are suited to the systems with both infinite and finite buffers. Furthermore, one can consider the average idle time of server n, which presents a criterion inherent only in the systems with finite buffers. It is defined for the manufacturing and communication blocking rules respectively as [1, 9]

$$\begin{aligned} IM_n &= \sum_{k=1}^{K}(D_n^k - C_n^k)/K, \\ IC_n &= \sum_{k=1}^{K}(B_n^k - H_n^k)/K. \end{aligned}$$

Note finally that these expressions may be also written in terms of departure epochs and service times in the same form as

$$I_n = \sum_{k=1}^{K}\left(D_n^k - (D_{n-1}^k \vee D_n^{k-1}) - \tau_n^k\right)/K.$$

4. Tandem Queues Simulation Algorithms

We start with the description of a simple simulation procedure designed for the tandem system with infinite buffers, and then extend the procedure to algorithms for systems with finite buffers and blocking. It is shown how the algorithms can be refined so as to evaluate system performance. In addition, time and memory requirements associated with the algorithms are briefly discussed.

(1) The Basic Simulation Procedure

We use the procedure proposed in [5], which was designed for the simulation of the tandem queueing system described by equations (1-2). It actually performs computations of successive state variables B_n^k and D_n^k with indices being varied in a particular order. According to this order, at each iteration i, the variables with $n+k=i$, $i=1,2,\ldots$, have to be evaluated. The next algorithm shows how to implement this procedure to the simulation of the first K customers in a tandem queueing system with infinite buffers and N servers, $K > N$.

Algorithm 1.

Set $d_i = 0, \quad i = -1, 0, \ldots, N;$
for $i = 1, \ldots, K+N,$ **do**
  $j_0 \longleftarrow \max(1, i-N);$
  $J \longleftarrow \min(i, K);$
  for $j = j_0, j_0+1, \ldots, J,$ **do**
    $b_{i-j} \longleftarrow d_{i-j-1} \vee d_{i-j};$
    $d_{i-j} \longleftarrow b_{i-j} + \tau_{i-j}^j.$

In Algorithm 1, the variables b_n and d_n serve all the iterations to store current values of B_n^k and D_n^k respectively, for $k = 1, \ldots, K$. Upon the completion of the algorithm, we have for server n the Kth departure time saved in d_n, $n = 0, 1, \ldots, N$.

Since one maximization and one addition have to be performed so as to get new variables B_n^k and D_n^k, one can conclude that the entire algorithm requires $O(2(N+1)K)$ arithmetic operations without considering index manipulations. Moreover, the order in which the variables are evaluated within each iteration is essential for reducing memory used for computations. It is easy to see that only $O(N+1)$ memory locations are actually required with this order, provided only the departure epochs D_n^k are to be calculated. To illuminate the memory requirements, let us represent Algorithm 1 in another form as

Algorithm 2.

Set $d_i = 0, \quad i = -1, 0, \ldots, N;$
for $i = 1, \ldots, K+N,$ **do**
  $j_0 \longleftarrow \max(1, i-N);$
  $J \longleftarrow \min(i, K);$
  for $j = j_0, j_0+1, \ldots, J,$ **do**
    $d_{i-j} \longleftarrow d_{i-j-1} \vee d_{i-j} + \tau_{i-j}^j.$

Finally, we suppose that there is a computer system with either a vector processor or SIMD parallel processors available for tandem queueing system simulation. In that case, we can use the following algorithm, which is actually a simple modification of Algorithm 1.

Algorithm 3.

Set $d_i = 0, \quad i = -1, 0, \ldots, N;$
for $i = 1, \ldots, K+N,$ **do**
  $j_0 \longleftarrow \max(1, i-N);$
  $J \longleftarrow \min(i, K);$
  in parallel, for $j = j_0, j_0+1, \ldots, J,$ **do**
    $b_{i-j} \longleftarrow d_{i-j-1} \vee d_{i-j};$
  in parallel, for $j = j_0, j_0+1, \ldots, J,$ **do**
    $d_{i-j} \longleftarrow b_{i-j} + \tau_{i-j}^j.$

Let P denote the length of vector registers of the vector processor or the number of parallel processors, depending on whether a vector or parallel computer system appears to be available. It is not difficult to see that Algorithm 3 requires the condition $P \geq N+1$ to be satisfied. Otherwise, if $P < N+1$, one simply has to rearrange computations so as to execute each iteration in several parallel steps. In other words, all operations within an iteration should be sequentially separated into groups of P operations, assigned to the sequential steps.

It has been shown in [5] that for any integer $P > 0$, Algorithm 3 requires $O\left(2N + 2K + 2\lfloor N/P \rfloor (K-P)\right)$ parallel (vector) operation, where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. Moreover, provided that $P = N+1$, the algorithm achieves linear speedup in relation to Algorithm 1 as the number of customers $K \to \infty$. Finally, it is easy to understand that Algorithm 3 entails $O(2(N+1))$ memory locations.

(2) Simulation of Queues with Finite Buffers

Taking equations (4-6) as the starting point, we can readily rewrite Algorithm 3 so as to make it possible to simulate tandem queueing systems with manufacturing blocking. Let us first introduce the variables b_n and c_n to represent current values of B_n^k and C_n^k. Since calculation of D_n^k involves taking account of the value of $D_{n+1}^{k-m_{n+1}-1}$, one has to keep in memory all values D_{n+1}^j with $j = k - m_{n+1} - 1, k - m_{n+1}, \dots, k$. Therefore, we further introduce the variables d_n^j as memory locations of these values, $n = 1, \dots, N$, $j = 0, 1, \dots, m_n$. The locations $d_n^0, d_n^1, \dots, d_n^{m_n}$ are intended to be occupied using cyclic overwriting so that the value D_n^k is put into the location d_n^j with $j = k \bmod (m_n + 1)$, where mod indicates the modulo operation.

In order to simplify further formulas, we define the index function

$$\rho(k,n) = \begin{cases} k \bmod (m_n+1), & \text{if } 1 \leq n \leq N \\ 0, & \text{otherwise} \end{cases}$$

for all $k = 1, 2, \dots$. Finally, with the variables d_{-1}^0, d_0^0, and d_{N+1}^0 reserved respectively for D_{-1}^0, D_0^k, and D_{N+1}^k, we have the next parallel algorithm.

Algorithm 4.

Set $d_{-1}^0, d_0^0, d_{N+1}^0 = 0$;
set $d_i^j = 0,\ i = 1, \dots, N,\ j = 0, 1, \dots, m_i$;
for $i = 1, \dots, K+N$, **do**
  $j_0 \longleftarrow \max(1, i-N)$;
  $J \longleftarrow \min(i, K)$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $b_{i-j} \longleftarrow d_{i-j-1}^{\rho(j,i-j-1)} \vee d_{i-j}^{\rho(j-1,i-j)}$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $c_{i-j} \longleftarrow b_{i-j} + \tau_{i-j}^j$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $d_{i-j}^{\rho(j,i-j)} \longleftarrow c_{i-j} \vee d_{i-j+1}^{\rho(j,i-j+1)}$.

Consider now equations (7-9) which describe the dynamics of the tandem system operating under the communication blocking rule. With the variables h_n, $n = 0, 1, \dots, N$, used as storage for the values of H_n^k, it is easy to arrive at

Algorithm 5.

Set $d_{-1}^0, d_0^0, d_{N+1}^0 = 0$;
set $d_i^j = 0,\ i = 1, \dots, N,\ j = 0, 1, \dots, m_i$;
for $i = 1, \dots, K+N$, **do**
  $j_0 \longleftarrow \max(1, i-N)$;
  $J \longleftarrow \min(i, K)$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $h_{i-j} \longleftarrow d_{i-j-1}^{\rho(j,i-j-1)} \vee d_{i-j}^{\rho(j-1,i-j)}$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $b_{i-j} \longleftarrow h_{i-j} \vee d_{i-j+1}^{\rho(j,i-j+1)}$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $d_{i-j}^{\rho(j,i-j)} \longleftarrow b_{i-j} + \tau_{i-j}^j$.

In fact, both algorithms differ from Algorithm 3 in that at every iteration, they involve three parallel operations each, whereas the latter does two operations. Therefore, we may extend the above estimate of time requirements for Algorithm 3 to Algorithm 4 and Algorithm 5, which then becomes $O(3N + 3K + 3\lfloor N/P \rfloor (K-P))$. The number of memory locations now involved in computations can be evaluated as $O(3(N+1) + M + 1)$, where $M = \sum_{i=1}^N m_i$.

(3) Evaluation of Performance Criteria

In order to present a modification of Algorithm 3 suitable for the evaluation of the tandem system performance criteria introduced in the previous section, first define the additional variables x_n, y_n, and z_n, $n = 0, 1, \dots, N$, to represent the memory locations which are to store current values of the sums

$$\sum_{i=1}^k (A_n^i - D_n^i), \qquad \sum_{i=1}^k (B_n^i - A_n^i), \qquad \sum_{i=1}^k \tau_n^i,$$

respectively. Taking into account that in the tandem systems with both infinite and finite buffers, we have $A_n^k = D_{n-1}^k$ for all $n = 0, 1, \dots, N$, and $k = 1, 2, \dots$, we may write the following parallel algorithm.

Algorithm 6.

Set $d_i = 0,\ i = -1, 0, \dots, N$;
set $x_i, y_i, z_i = 0,\ i = 0, 1, \dots, N$;
for $i = 1, \dots, K+N$, **do**
  $j_0 \longleftarrow \max(1, i-N)$;
  $J \longleftarrow \min(i, K)$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $x_{i-j} \longleftarrow x_{i-j} - d_{i-j-1}$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $y_{i-j} \longleftarrow y_{i-j} - d_{i-j-1}$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $b_{i-j} \longleftarrow d_{i-j-1} \vee d_{i-j}$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $d_{i-j} \longleftarrow b_{i-j} + \tau_{i-j}^j$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $x_{i-j} \longleftarrow x_{i-j} + d_{i-j}$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $y_{i-j} \longleftarrow y_{i-j} + b_{i-j}$;
  in parallel, for $j = j_0, j_0+1, \dots, J$, **do**
    $z_{i-j} \longleftarrow z_{i-j} + \tau_{i-j}^j$.

As it is easy to understand, Algorithm 6 requires $O(7N + 7K + 7\lfloor N/P \rfloor (K-P))$ parallel operations, and involves $O(5(N+1))$ memory locations. Upon the completion of the algorithm, the performance criteria associated with each queue n, $n = 1, \dots, N$, can be calculated as

$$\begin{aligned} S_n &= x_n/K, & W_n &= y_n/K, \\ T_n &= K/d_n, & U_n &= x_n/d_n, \\ J_n &= x_n/d_n, & Q_n &= y_n/d_n. \end{aligned}$$

One can modify both Algorithm 4 and Algorithm 5 to provide performance evaluation in tandem queueing systems with finite buffers in an analogous way. Specifically, the next two algorithms intended to compute the average idle time of each server in the system. The first one based on Algorithm 4 is designed for the system operating under the manufacturing blocking rule.

ALGORITHM 7.

Set $d_{-1}^0, d_0^0, d_{N+1}^0 = 0$;
set $d_i^j = 0, \; i = 1, \ldots, N, \; j = 0, 1, \ldots, m_i$;
set $x_i = 0, \; i = 0, 1, \ldots, N$;
for $i = 1, \ldots, K + N$, **do**
  $j_0 \longleftarrow \max(1, i - N)$;
  $J \longleftarrow \min(i, K)$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $b_{i-j} \longleftarrow d_{i-j-1}^{\rho(j,i-j-1)} \vee d_{i-j}^{\rho(j-1,i-j)}$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $c_{i-j} \longleftarrow b_{i-j} + \tau_{i-j}^j$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $x_{i-j} \longleftarrow x_{i-j} - c_{i-j}$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $d_{i-j}^{\rho(j,i-j)} \longleftarrow c_{i-j} \vee d_{i-j+1}^{\rho(j,i-j+1)}$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $x_{i-j} \longleftarrow x_{i-j} + d_{i-j}^{\rho(j,i-j)}$.

The variable x_n inserted in Algorithm 7 serves for each n, $n = 0, 1, \ldots, N$ to represent current values of the sums $\sum_{i=1}^k (D_n^i - C_n^i)$. Upon the completion of the algorithm, one can calculate x_n/K which gives the value of IM_n. The time and memory costs can be estimated respectively as $O\,(5N + 5K + 5\lfloor N/P \rfloor (K - P))$ and $O(4(N + 1) + M + 1)$.

ALGORITHM 8.

Set $d_{-1}^0, d_0^0, d_{N+1}^0 = 0$;
set $d_i^j = 0, \; i = 1, \ldots, N, \; j = 0, 1, \ldots, m_i$;
set $x_i = 0, \; i = 0, 1, \ldots, N$;
for $i = 1, \ldots, K + N$, **do**
  $j_0 \longleftarrow \max(1, i - N)$;
  $J \longleftarrow \min(i, K)$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $h_{i-j} \longleftarrow d_{i-j-1}^{\rho(j,i-j-1)} \vee d_{i-j}^{\rho(j-1,i-j)}$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $x_{i-j} \longleftarrow x_{i-j} - h_{i-j}$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $b_{i-j} \longleftarrow h_{i-j} \vee d_{i-j+1}^{\rho(j,i-j+1)}$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $x_{i-j} \longleftarrow x_{i-j} + b_{i-j}$;
  in parallel, for $j = j_0, j_0 + 1, \ldots, J$, **do**
    $d_{i-j}^{\rho(j,i-j)} \longleftarrow b_{i-j} + \tau_{i-j}^j$.

With the same time and memory requirements as for the previous algorithm, Algorithm 8 allows one to evaluate the average idle time of each servers in tandem queues with communication blocking. It produces the sums $\sum_{i=1}^K (B_n^i - H_n^i)$ stored in x_n, $n = 1, \ldots, N$, which can be used in calculation of the criteria IC_n with the expression x_n/K.

5. Conclusions

Parallel algorithms which offer a quite simple and efficient way of simulating tandem queueing system have been proposed. It has been shown that the algorithms involve low time and memory requirements. Specifically, one can conclude that the parallel simulation of the first K customers in a system with N queues requires the time of order $O\,(L(N + K + \lfloor N/P \rfloor (K - P)))$, where P is the number of processors, L is a small constant comparable with the number of the performance criteria being evaluated. Note, however, that this estimate ignores the time required for computing indices, and allocating and moving data, which can have an appreciable effect on the performance of parallel algorithms in practice.

References

[1] Chen, L. and C.-L. Chen, A Fast Simulation Approach for Tandem Queueing Systems *Proceedings of the 1990 Winter Simulation Conference, New Orleans, LA, Dec. 9-12* (O. Balci, R.P. Sadowski and R.E. Nance, eds), 1990, 539–546.

[2] Cheng, D.W., On the Design of a Tandem Queue with Blocking: Modeling, Analysis, and Gradient Estimation, *Naval Research Logistics*, **41**, 1994, 759–770.

[3] Greenberg, A.G., B.D. Lubachevsky and I. Mitrani, Algorithms for Unboundedly Parallel Simulations, *ACM Transactions on Computer Systems*, **9**(3), 1991, 201–221.

[4] Ermakov, S.M., *Die Monte-Carlo-Methode und verwandte Fragen*, VEB Deutscher Verlag der Wisseschaften, Berlin, 1975.

[5] Ermakov, S.M. and N.K. Krivulin, Efficient Algorithms for Tandem Queueing System Simulation, *Applied Mathematics Letters*, **7**(6), 1994, 45–49.

[6] Heidelberger, P., Discrete Event Simulation and Parallel Processing: Statistical Properties, *SIAM Journal on Statistical Computing*, **9**(6), 1988, 1114–1132.

[7] Krivulin, N.K., Optimization of Complex Systems for Simulation Modeling, *Vestnik Leningrad University: Mathematics*, **23**(2), 1990, 64–67.

[8] Krivulin, N.K., Unbiased Estimates for Gradients of Stochastic Network Performance Measures, *Acta Applicandae Mathematicae*, **33**, 1993, 21–43.

[9] Krivulin, N.K., Recursive Equations Based Models of Queueing Systems. In *Proceedings of the European Simulation Symposium, İstanbul, Turkey, Oct. 9-12, 1994* (Ali R. Kaylan, Axel Lehmann, Tuncer İ. Ören, Eds.), 1994, 252–256.

[10] Ortega, J.M., *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, New York, 1988.