CS-TR-82-04

REGENERATIVE METHOD OF A QUEUING NETWORK SIMULATION AS A TOOL IN COMPUTER SYSTEM PERFORMANCE ANALYSIS

by

Andrzej Proskurowski

Department of Computer and Information Science

University of Oregon, Eugene, OR.

Abstract A large software system requiring massive data transfers and a substantial computational effort is to be run in a timesharing environment competing for resources with a number of other users. With a simple queuing network as a model of the computer system, the regenerative method of simulation is used for studies of the expected user response times in the system, subject to varying assumptions about the distribution of user jobs' characteristics.

This report describes work performed while visiting IBM GPD, Los Gatos.

I. Introduction

Every reasonable development activity incorporates some kind of "prediction -- correction" cycle in which the anticipated effects of design decisions are evaluated, and appropriate changes are made. To issue an informed prediction of a system behavior it is often necessary to first model the relevant aspects of the system, and then to compute parameters of the model characterizing that behavior. This computation may be performed analytically, or by means of a (computer) simulation.

In the case of our study, we are interested in the response time behavior of a timesharing computer system which, beside the normal load of interactive user jobs, serves a large application system requiring a special treatment by the operating system. The study has been motivated by some questions raised during the development of a high resolution, large screen, raster scan display unit. We have chosen to model the computer system by a simple queuing network and concentrated on applying a powerful simulation method to analyze the system behavior. This method allows computation of the reponse time estimate and its variability with the prescribed confidence and with a predictable expenditure of computational effort. This <u>regenerative method</u> of simulation is based on the notion of probabilistically identical appearance of a system every time the stochastic process describing the system's behavior enters a certain <u>regeneration state</u>. If such a state exists, then the system's state trajectory between any two consecutive entries into the regeneration state can be treated as an outcome of an independent probabilistic experiment, with probability distribution identical for every such pair of time instances. An estimate of a given parameter of the system's behavior can be computed for each such experiment, and the results of a number of these experiments could be corelated giving the confidence interval for the final estimate. From a preliminary estimate one can compute the number of such experiments (which can be performed within a single simulation run) necessary to attain a desired confidence interval of the estimate.

For an exposition of the method and a discussion of its implications, the reader is referred to monographs [Crane] and [Iglehart], or to standard simulation texts like [Kobayashi] or [Sauer].

2. Basic queuing network model

Our simplest model consists of three queuing and service centers which, together with the routing scheme, constitute a closed system with two users. One of these users corresponds to the special application, distinct from the normal interactive user jobs which are collectively represented by the second user. One of the service centers corresponds to the central processing unit, and the other two -- to the "home processing" for either of the user types. When at its home processing center, the user immediately starts being served as there is no competition for that resource. The processing time in these centers models "thinking time" during which the corresponding computer user does not request processing from the computer system. Such a request is modeled in the queuing network by the arrival of a user at the central service station. At this central serving station some kind of a non-trivial queuing behavior (depending on a particular queuing strategy) is necessitated by possible conflicting requests from the users.

To complete the description of our model we have to characterize the thinking time behavior and processing requirements of the two user types, as well as the treatment of competing requests for the use of the central processor. The choice of the service time distributions at the three service centers and of the queuing strategy at the central service station should reflect our understanding of the real system (the temporal behavior of the interactive computer users) and of the projected operation of the special application. On the other hand, this choice may influence the computational burden of simulation. As a compromise, we will at first choose memoryless (exponential) distributions for both thinking times and the interactive users' processing time, and the constant time processing for the application. We will also employ the "preemptive-- resume" queuing strategy at the central service station for the initial analysis of the system's performance.

In the continuation of our experiments we will vary both the topology of the queuing network, the queuing strategy, and the service time distribution character for the users. The topology changes will involve fragmentation of the interactive user type in the closed queuing network. The alternative service strategy will be processor sharing with the infinitesimal time slice. The distribution changes consist of both changing values of the expected service time, and exploring the performance of the model for different variability of the distributions.

3. Service time distribution considerations

We will briefly describe the motivation for decisions involving modeling of the temporal processes in the real system and the terms used to characterize them. The computer system exhibits certain "input" behavior: thinking time for a given user, requests for the central processing unit use for a given job issued at given time points, the related processing times. In our simulation procedure we would like to substitute for these input values outcomes (positive real numbers) of random experiments of the same character as the real life process. This "character" is described by the probability distribution function, the probability that the experiment's outcome takes a value not greater than the function's argument. Its derivative is called the probability density function. A parametrization of a distribution function may contain the expected value of a random variable and the variance: the integral of, respectively, the outcome values and the squares of their differences from the expected value, weighted by the probability density, over all possible outcome values. The ratio of these two parameters is called the variation coefficient and is frequently used to quantify variability of a random process. It takes value 0 for a constant process (whose all outcomes are equal), value close to 1 for processes in which the square root of variance value is commeasurable with the expected value, and is very large for a process with variance large when compared to its expected value. For example, a random variable distributed uniformly between 0 and some given value has the coefficient of variation near 0.6. Another important distribution type, very frequently used in modeling of computer systems is the exponential distribution. The coefficient of variation of this distribution is exactly 1. We will choose the exponential distribution for the first approximation of the temporal behavior of the computer system users. Sometimes we have more information about the users' time requirements: either they are a homogeneous group (yielding relatively small variations of the service time values), or can be partitioned into two distinct groups (with two different mean service time values). These situations are frequently modeled by, respectively, hypoexponential and hyperexponential distribution types. Random variables with these distributions have values of the coefficient of variation respectively below and above 1 (which characterizes under- and over-dispersed distributions). We will use these types of distribution at a later stage in our simulation experiments. The exponential distribution is often called memoryless when the process it characterizes involves the occurrence times of some events. The distribution function of such a random variable remains unchanged when subjected to positive shift of the time origin: the probability that an event will occur within a given time interval is the same at any reference point, provided that it has not occurred by that time point. This memoryless property is a great asset in a simulated system if the regenerative method is used, since it often permits more states of the model to qualify as the regeneration states.

Our choice of the constant processing time for the special application job has been motivated by the character of the application under consideration. The working representation of the picture is often in a vector form, while rewriting of its display requires its transformation into a bit raster form. Depending on the conversion technology, this process may take time only little influenced by the content of the picture. A better approximation of the processing time requirements will allow for "partial rewrites".

4. Computation of the expected response time

In this section we discuss the analytical computation of a queuing network performance parameter estimate. We formally describe the queuing network and the states of a continuous time stochastic process describing its behavior. Time elapsed between certain state transitions of this process determines another stochastic process which may be interpreted as representing response time for a user of the system modeled by the network. In order to estimate this response time, an embedded discrete time process is defined which carries enough information about state transitions. The analysis of this latter process leads to the determination of its parameters: the probabilities of one step transitions and the expected unconditional holding times. These in turn allow to analytically compute the expected value of the response time, since this discrete time stochastic process exhibits regenerative behavior.

Our queuing network consists of three queuing and service stations in a closed system for two users. Both users are served at center Ø; upon completion of this service, user 1 is routed to service station 1, and user 2 is routed to service station 2. The network is depicted in Figure 1. Upon completion of service in their respective home centers, the users may again compete for service at center Ø. The queuing strategy employed at center Ø is preemptive--resume, with user 2 having the priority over user 1 and thus never waiting for service to commence. The preempted user 1 has its service resumed when user 2 leaves the center Ø. The service times tØ1, t11, t22 for users 1 and 2 at centers Ø, 1, and 2, respectively, are distributed exponentially with rate parameters lambdaØ1, lambda1, and lambda2. The service time for user 2 at center Ø is assumed to be constant and equal tØ.

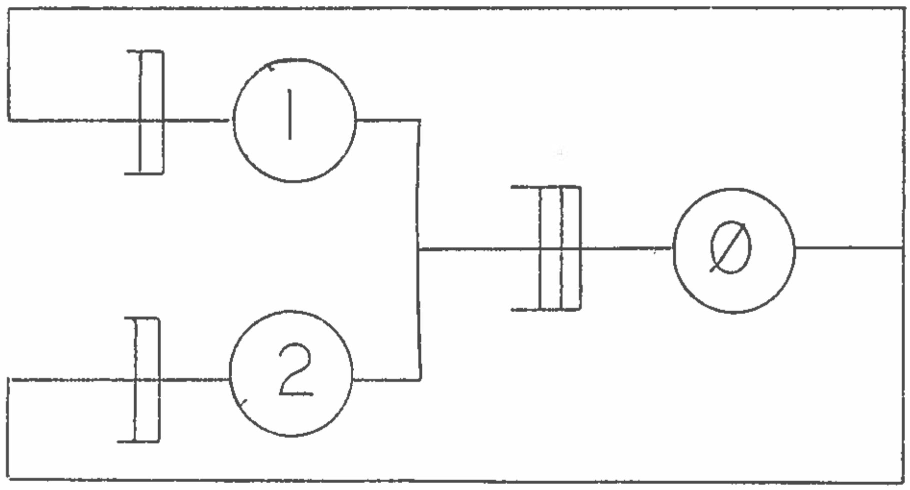


Figure 1. The queuing network model

The state of the network is given by the "states" of the two users. Let "h" denote processing by the home center (center 1 for user 1 and center 2 for user 2), let 'w' denote user waiting for service at center Ø, and let 'c' denote user being served at center Ø. There is clearly no waiting involved at the home centers. Because of the preemptive strategy, user 2 never waits for service at center Ø either, and thus the states of our model can be partially represented by one of the following ordered pairs of user states (the first position representing user 1, and the second representing user 2): <h,h>, <h,c>, <c,h>, <w,c>. Observing the state of the network in time we define a continuous time stochastic process $\{X(t): t \geq Ø\}$, with the four states in its state space. The state transition diagram of process X(t) is given in Figure 2a.

We define the system response time for a user as the time spent between completion of service at its home center and the subsequent completion of the requested service at center Ø. This corresponds to the time elapsed between the model leaving state 'h' on the appropriate position and the subsequent entrance of a state with 'h' on this position. This defines a continuous time process $\{R1(t): t \geq Ø\}$ taking value Ø when user 1 is at service center 1, and 1 otherwise (when "the response time clock is ticking"). We want to compute the expected value of this process. For this purpose we define discrete time stochastic process $\{Y(tk): k \geq Ø\}$ embedded in the process X(t). Time instances tk are defined as moments of completion of service at centers Ø (for both users) and at center 2 (by user 2). The completion of service at center 1 has been excluded because of implications of the possible partial completion of service to user 2 at center Ø on the future behavior of the process Y. The choice of tk and the memoryless distribution of the remaining service times ensure that Y(tk) is a discrete time semi-Markov process with the state space E = {<h,h>, <h,c>, <w,c>, <c,h>}. The state transition diagram for Y(tk) is given in Figure 2b. Note the state transitions in X(t) not appearing in the transitions of Y(tk), and vice versa.

The termination of each response time for user 1 is reflected by a transition to state <h,h>. Every such transition, and only such a transition, marks completion of a life-cycle for user 1 which in terms of the modeled computer system can be expressed as "think, possibly wait for processing, process". Therefore, the expected value of this response time can be computed as the difference between the expected value of the life-cycle and the expected value of think time for user 1.

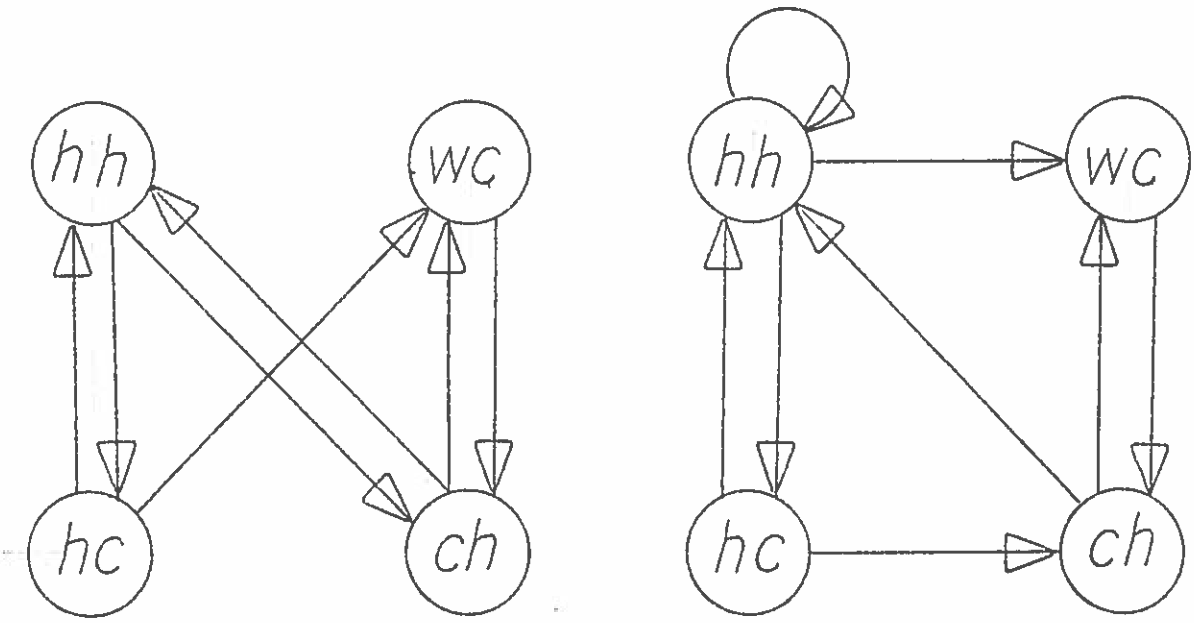


Figure 2. State transition diagrams for the processes X(t) and Y(tk)

We will determine the expected life span of a computer job of type 1 by analysis of the process Y(tk) represented by its one step transition matrix P and the unconditional holding time rate parameter vector Q . The analysis involves considering all possible multiple step transitions to the state <h,h> which mark completion of service at center Ø to user 1. The probability of these transitions are indirectly given by P and the amount of time involved is given by Q .

Let us first determine the entries in P, which are the probabilities of one step state transitions in Y(tk). Let s1, s2 denote instances of the service time at centers 1 and 2, while sØ1 and sØ2 denote service time at center Ø for users 1 and 2, respectively. We recall that s1, s2, and sØ1 are distributed exponentially, while sØ2 always takes the value tØ.

<h,h> -> <h,c> occurs when s2≤s1, with probability lambda2/(lambda1 + lambda2).

<h,h> -> <h,h> occurs in Y(tk) when user 1 completes service at center 1 and then at center Ø before user 2 completes service at center 2 (cf. Figure 2a). This happens when s1+sØ1≤s2 with probability lambda1/(lambda1 + lambda2) * lambdaØ1/(lambdaØ1 + lambda2)

<h,h> -> <w,c> occurs in Y(tk) when the user 1 completes service at center 1 before user 2 completes service at center 2 but then the latter event interrupts service to user 1 at center Ø. This happens with probability lambda1/(lambda1 + lambda2) * lambda2/ (lambdaØ1 + lambda2) = Pr[s1≤s2 and s1 + sØ1≥s2]

<h,c> -> <c,h> occurs when user 1 completes service at center 1 before

user 2 completes service at center 0 (the occurence time of the former event is not one of tk's in Y(tk); the transition in Y(tk) occurs after user 2 eventually computes the service at which time user 1 starts being served by center 0). The total probability of this transition is Pr[s1≤s02] = 1 - exp(-lambda1*t0)

<h,c> -> <h,h> occurs when s02≤s1, which happens with probability exp(-lambda1*t0);

<c,h> -> <h,h> occurs when s01≤s2, with probability lambda01/ (lambda01 + lambda2)

<c,h> -> <w,c> occurs when s2≤s0, with probability lambda2/ (lambda01 + lambda2)

<w,c> -> <c,h> occurs always.

For each of these four states of Y(tk) we will now analyze the average time that elapses between two consecutive transitions into and from that state in Y(tk). This information will be represented by the unconditional holding time rate parameter vector Q .

For the states <h,c> and <w,c>, the time between the entrance and the exit is equal to the service time of user 2 at center 0. Thus the corresponding entries of Q are both 1/t0. Time spent in state <c,h> is the smaller of the thinking time for user 2, s2, and the processing time for user 1, s01. Because both these times are distributed exponentially, the expected value is given by the rate parameter of lambda01 + lambda2. In the continuous time process X(t), the transition from state <h,h> may occur after the smaller of the times s1 and s2 elapses since the entry into this state; the expected value of this time is 1/(lambda1 + lambda2). If s2 is this smaller time, then the process Y(tk) undergoes the state transition into <h,c>. If s1 happens to be the smaller time (which occurs with probability lambda1 / (lambda1 + lambda2)), then no state transfer would occur in Y(tk) until, additionally, the smaller of s2 and s01 elapses (compare with the above analysis for <c,h>). Averaging over these two possibilities, we get the expected holding time in <h,h> equal to 1/(lambda1 + lambda2) + lambda1 / ((lambda1 + lambda2) * (lambda01 + lambda2)).

The unconditional holding time vector Q^{-1} has entries representing the expected time and equal to the inverses of the coresponding rate parameters in Q. Figure 3 presents both Q^{-1} and the one step transition matrix P.

$$(a)\quad \begin{array}{c} hh \\ hc \\ wc \\ ch \end{array} \begin{pmatrix} \frac{\lambda_1}{\lambda_1+\lambda_2}\cdot\frac{\lambda_{01}}{\lambda_{01}+\lambda_2} & \frac{\lambda_2}{\lambda_1+\lambda_2} & \frac{\lambda_1}{\lambda_1+\lambda_2}\cdot\frac{\lambda_2}{\lambda_{01}+\lambda_2} & 0 \\ e^{-\lambda_1 t_0} & 0 & 0 & 1-e^{-\lambda_1 t_0} \\ 0 & 0 & 0 & 1 \\ \frac{\lambda_{01}}{\lambda_{01}+\lambda_2} & 0 & \frac{\lambda_2}{\lambda_{01}+\lambda_2} & 0 \end{pmatrix} \qquad (b)\quad \begin{pmatrix} \frac{1}{\lambda_1+\lambda_2} + \frac{\lambda_1}{(\lambda_1+\lambda_2)(\lambda_{01}+\lambda_2)} \\ t_0 \\ t_0 \\ \frac{1}{\lambda_{01}+\lambda_2} \end{pmatrix}$$

<u>Figure 3.</u> (a) One step transition matrix P, and (b) unconditional holding time vector Q^{-1} describig the behavior of the queuing network model.

The fact that Y(tk) is a discrete time semi-Markov process allows us to compute analytically the expected time that elapses between two consecutive entrances of the process into the state <h,h>. The property of Y(tk) crucial for the result used here ([Hamacher]) is that the random variables Y1i ($i \geq 1$) describing the duration of time between ith and i+1st entrances into <h,h> are independent and identically distributed. Let us denote by $\emptyset \underline{P}$ the matrix obtained from $\underline{P}$ through zeroing its first column (corresponding to <h,h>). The following formula employs the information given by $\emptyset \underline{P}$ and Q^{-1} in considering all transitions from state <h,h> to itself which avoid intermediate transitions into <h,h>.

$$E(Y1) = ((I-\emptyset \underline{P})^{-1} Q^{-1})$$

By the definition of the process Y(tk), E(Y1) is also the expected life span of a computer job of type 1. This life span consists of the "home processing time" (distributed exponentially with the rate parameter lambda1) and the response time, R1. Thus, the expected value of the response time for a job of type 1 is

$$E(R1) = E(Y1) - 1/\text{lambda1}.$$

5. Response time analysis

In the analysis of the previous section we made use of the fact that the regeneration state <h,h> was being entered exactly once for every processed job of type 1. Figure 4 gives a more general queuing network modeling the computer system under consideration. This model includes a number of "standard" job types (service centers 1..N-1) and the non-standard application job (center Ø).

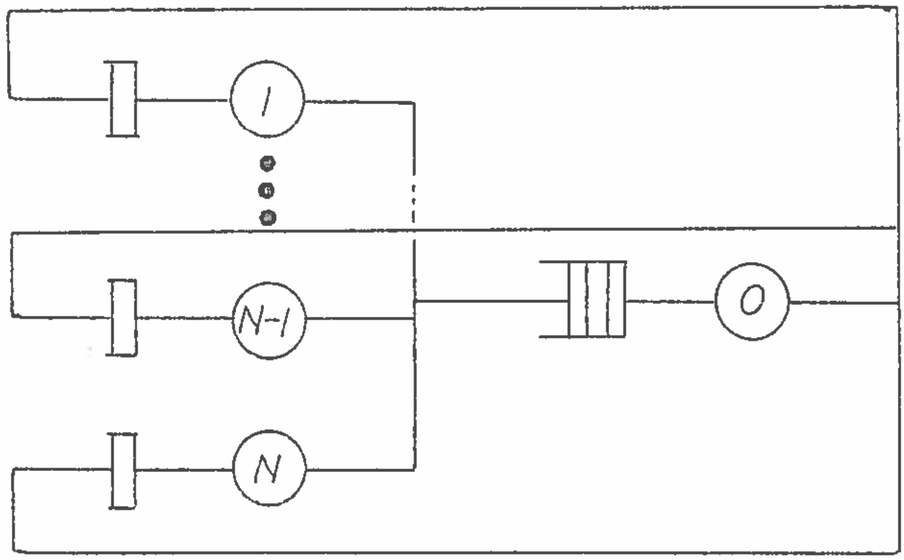


Figure 4 A generalized queuing network model

As before, the service times will be assumed to follow exponential distributions with rate parameters lambda1 .. lambdaN (for home centers) and lambdaØ1..lambdaØN-1 for center Ø (processing of standard jobs). These assumptions and a careful choice of state transition instances, tk, allow us to define in a manner analogous to that of the previous section a discrete time semi-Markov process describing the queuing network's behavior. Without loss of generality, we consider as the regeneration state the state in which all jobs are in their home service centers. We are interested in what happens during the inter-regenerative cycles, i.e., between two consecutive entrances into the regeneration state. Namely, what is the length, Y1i, of the ith cycle and what is the number, N1i, of jobs of type 1 encountered (processed) during that cycle. The pairs of random variables <Y1i,N1i> are independent and identically distributed which allows computation of the point estimate (based on n cycles) for the mean response time for job of type 1 as

En(Yl) / En(Nl) - 1/lambdal.

Here, En denotes sample mean over n cycles.

An application of the central limit theorem (see [Hamacher]) provides values of the confidence intervals for the estimate, i.e., the size of the neighborhood in which the actual value lies with a given probability. Namely, a simulation of the queuing network for n regenerative cycles is used not only to obtain the sample estimates of E(Yl) and E(Nl), but also the unbiased estimates s11, s22, and s12 of the relevant (co-)variances: Var(Yl), Var(Nl), and Cov(Yl,Nl), respectively. (We include the well known formulea for these parameters in the Appendix for completeness.) These values, together with the number of regenerative cycles, n, and the desired confidence probability give the size of the confidence interval.

A value of the mean response time alone carries little information as to what a user may actually encounter in terms of the system performance. The analysis of the response time may be extended by the computation of point estimates of its distribution function. For each such point estimate the confidence interval is determined. The variability information for a random variable with the probability distribution function F(x) may be represented by the graph of -log(1-F(x)). The curvature of the graph (concave or convex) indicates the degree in which the variation coefficient of the random variable differs from 1 (in the over- and under-dispersed distribution character, respectively).

6. Simulation experiments

The discussion of the preceding sections was intended as an introduction to some new (though not original) concepts in a computer system performance analysis. Our knowledge of the parameters of the particular system was inadequate to issue recommendation based on the conducted simulation experiments described in this section. Rather, we want to use these experiments as illustration of the methodological approach we think appropriate in a preliminary study, such as ours.

Based on the model developed in the previous sections a discrete-event simulation program which allows a response time analysis has been written. This program has a highly modular structure accomodating possibilities of different central processor utilization strategies, different probability distributions of the non-standard job processing time, and different number and distribution prameter values of the standard job processing times. The generation of random values is an important implementation issue. We chose to use a number of pseudo-random number streams, one for each job type and service center, generated by a congruential generator from the seeds very widely spread in the cycle of all values produced by that generator (as recomended by [Iglehart]).

Our simulation experiments were conducted with the following two issues in mind. First, whether a model including more queuing and service stations corresponding to standard users with the same behavior parameters would lead to better simulation results than the simple model, thus justifying the increased computational effort. Second, what degradation of the application job performance could be observed in our model if the application were to timeshare the central processor with the interactive users.

The distribution parameters for the thinking and processing time of the interactive users were chosen quite arbitrarily, though they follow indications of some earlier experiments (Cary Campbell, private communication). In our experiments, each of two values for the mean thinking time has been used yielding predictably different results. We have conducted only one series of experiments with other than constant processing time for the application job. The observed veriability of the interactive user response time was not significantly different from that of the other distribution.

In the remainder of this section we present graphically results of the experiments conducted together with some brief comments. The confidence intervals shown correspond to 90% confidence.

6.1 Variability experiments for the interactive user response time in the presence of the total of N jobs.

(a) The over-dispersed variability of the response time increases when the saturation of the central processor by an increased number of jobs causes more of them to be preempted by the application job.

(b) With a larger initial utilization of the central processor by interactive jobs, an increase in their number causes more uniform, although larger, expected response time; hence, the variability of it exhibits under-dispersed character.

(c) No significant differences for the windowed exponential probability distribution of application job processing time.

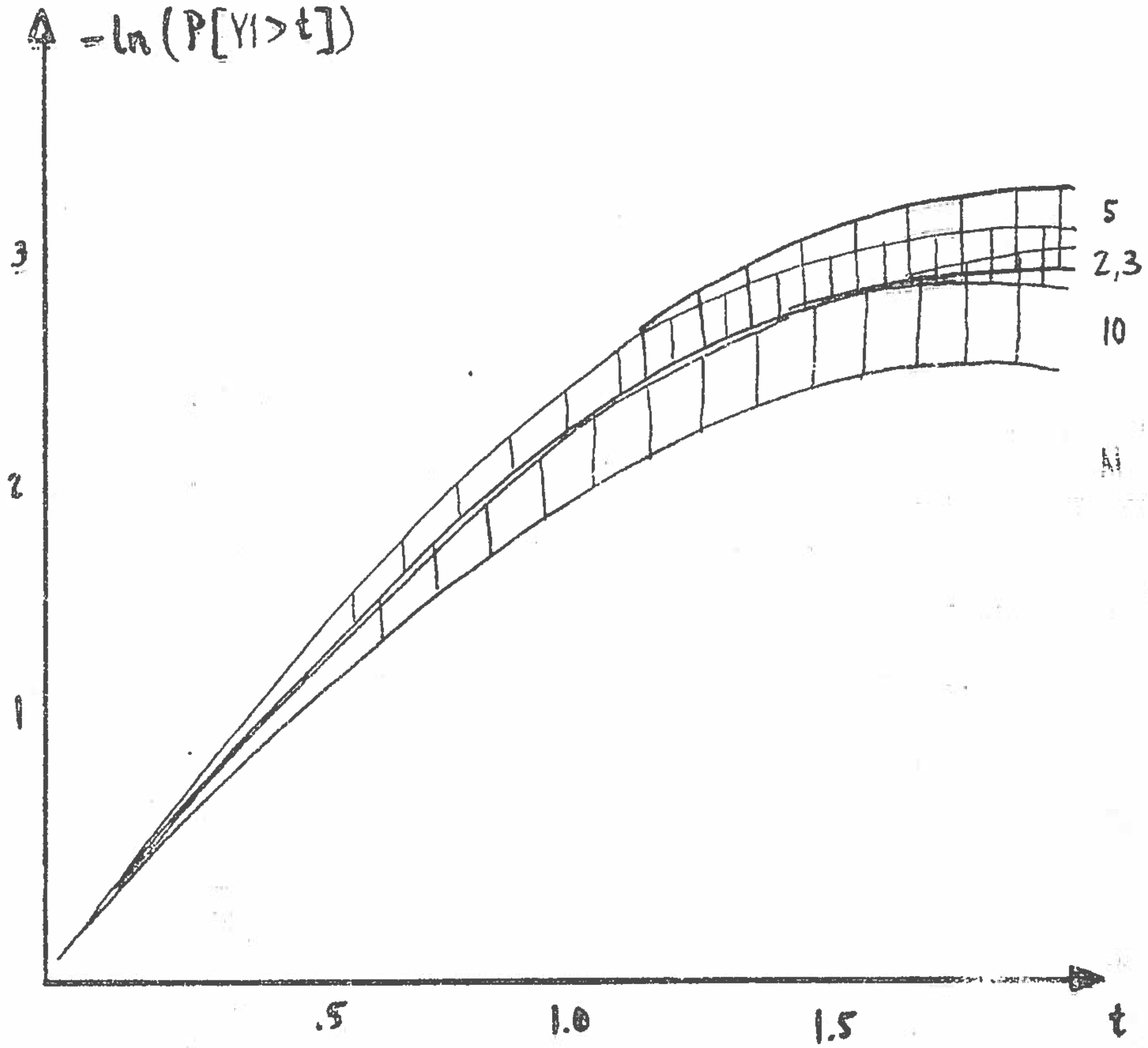
6.2. Comparison between two choices of the expected thinking time. Larger initial saturation of the central processor for the higher value of the rate parameter causes an initially lower value of the expected response time, as a lower percentage of jobs is preempted. However, the expected response time for the higher rate stays almost constant as the number,of jobs increases, since the higher rate jobs become more uniformly preempted with an incraese in numbers.

6.3 The application job timesharing the central processor.

(a) For the smaller rate of the interactive jobs, the expected response time for the application seems to grow linearly with the number of jobs.

(b) For the larger rate of interactive jobs, the growth in the application job's response time seems to be more pronounced.

(c) When the total load of interactive jobs is kept constant by adjusting their arrival rate to their number, the observed response times grow rather insgnificantly with the number of jobs.

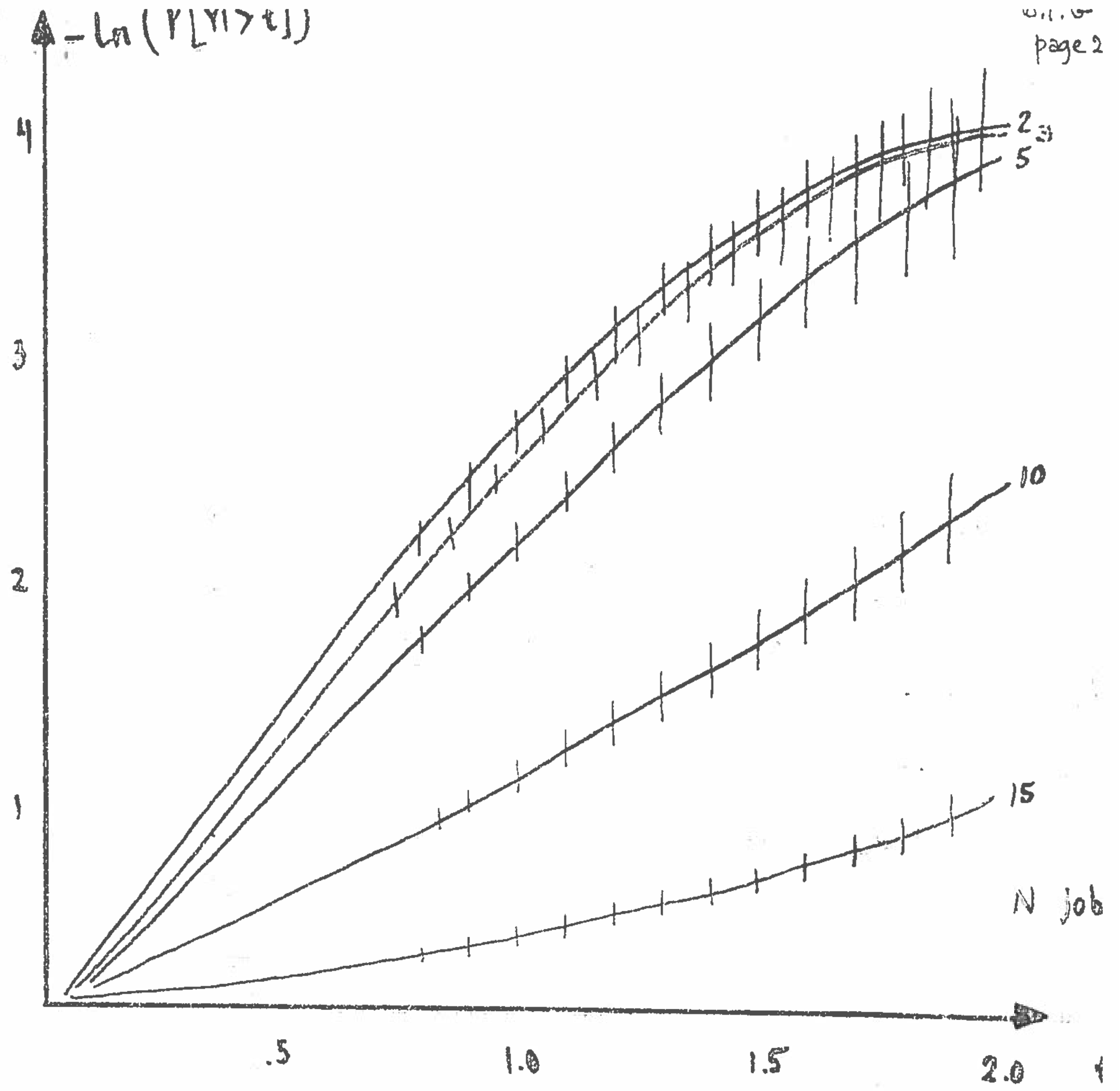


PREEMPTIVE QUEUE

$\lambda_1 = \cdots = \lambda_{N-1} = 1/30$

$\lambda_{01} = \cdots = \lambda_{0N-1} = 3.0$

$\lambda_N = 1/300$; $t_{0N} = 20.0$

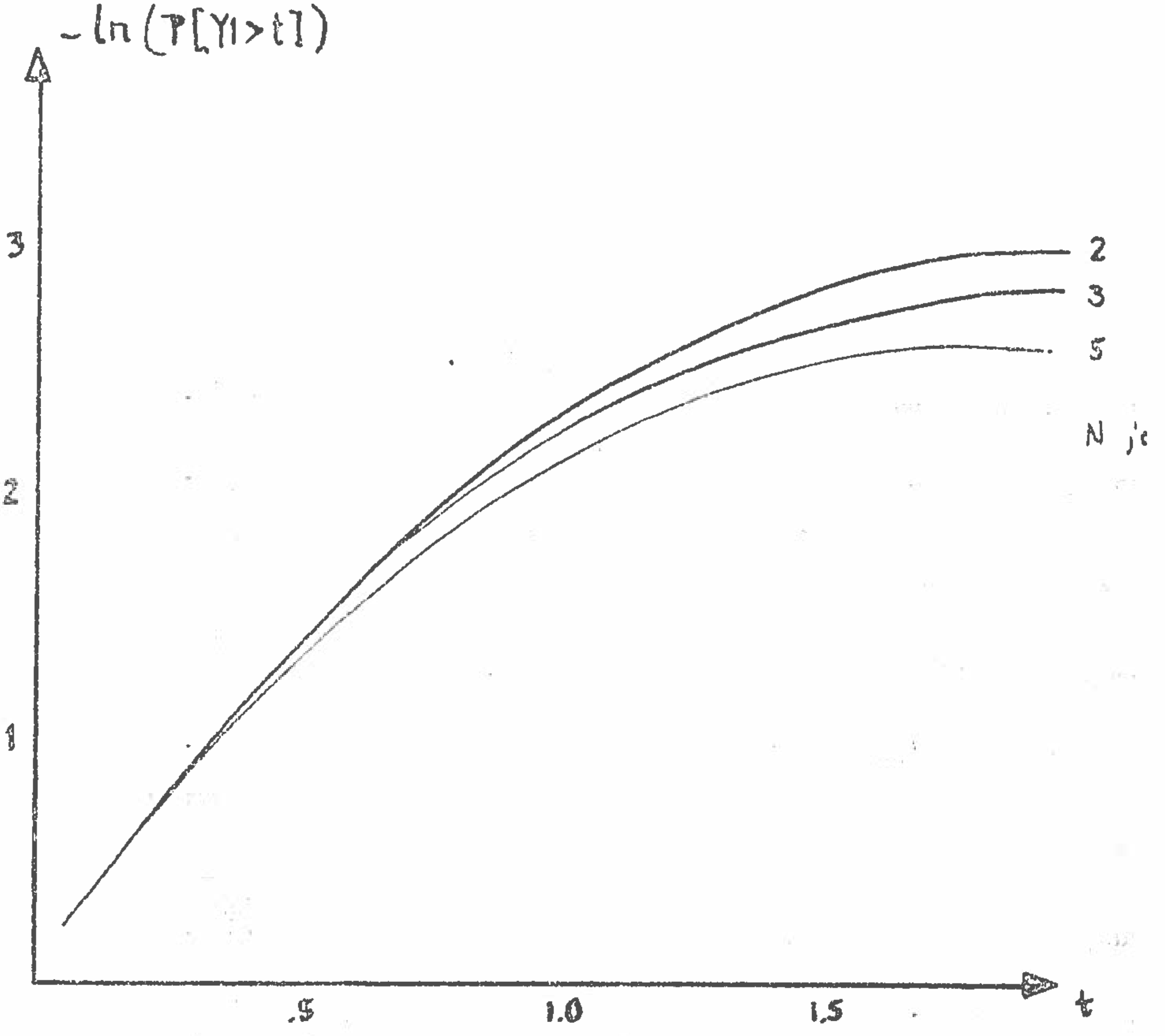


PREEMPTIVE QUEUE

$\lambda_1 = \cdots = \lambda_{N-1} = 1/3$

$\lambda_{01} = \cdots = \lambda_{0N-1} = 3.0$

$\lambda_N = 1/300$; $t_{0N} = 20.0$



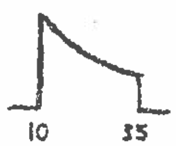
PREEMPTIVE QUEUE

$\lambda_1 = \cdots = \lambda_{N-1} = 1/30$

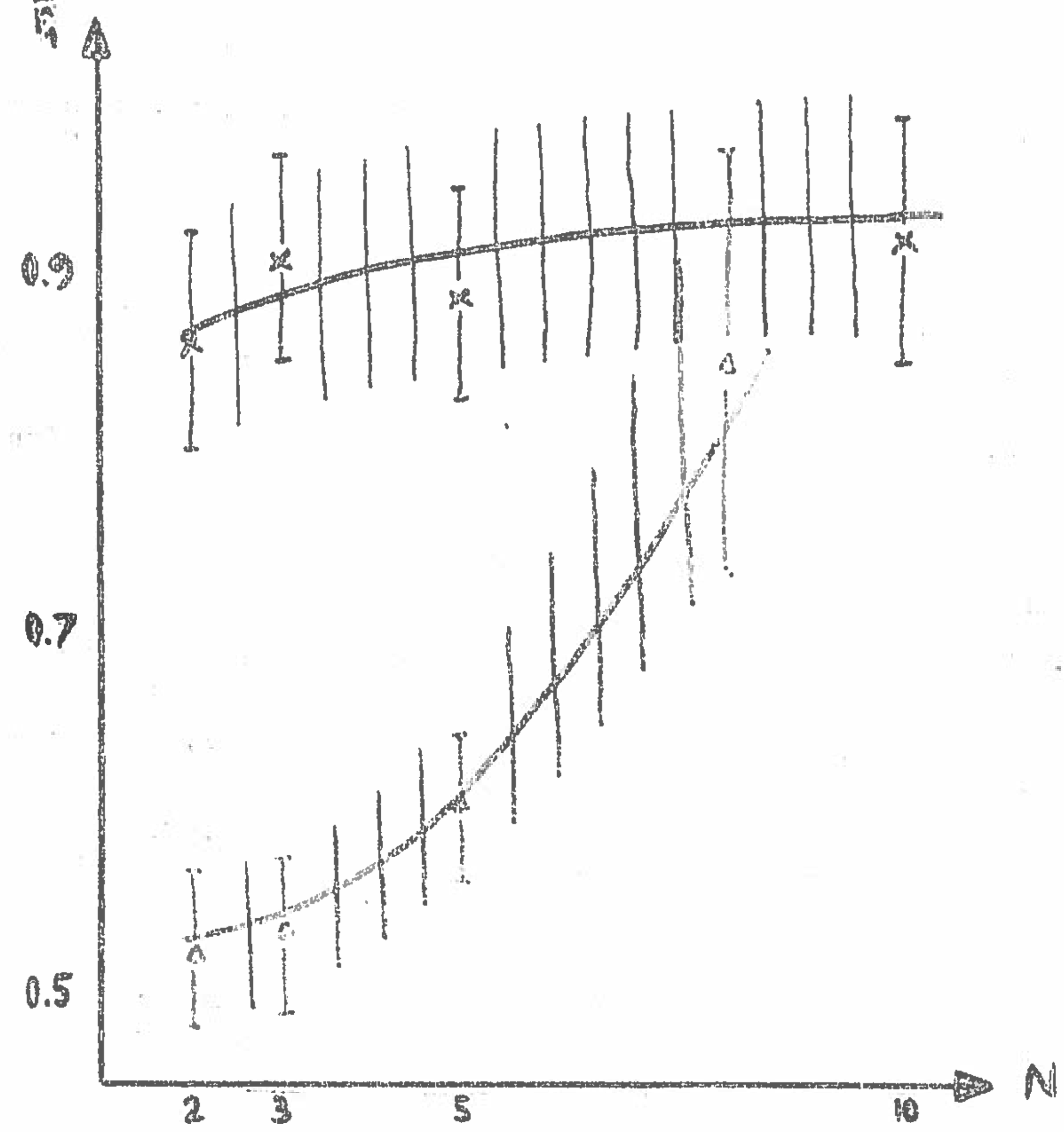
$\lambda_{01} = \cdots = \lambda_{0N-1} = 3.0$

$\lambda_N = 1/300$

$\lambda_{0N} = 1/20$ (windowed exponential



)



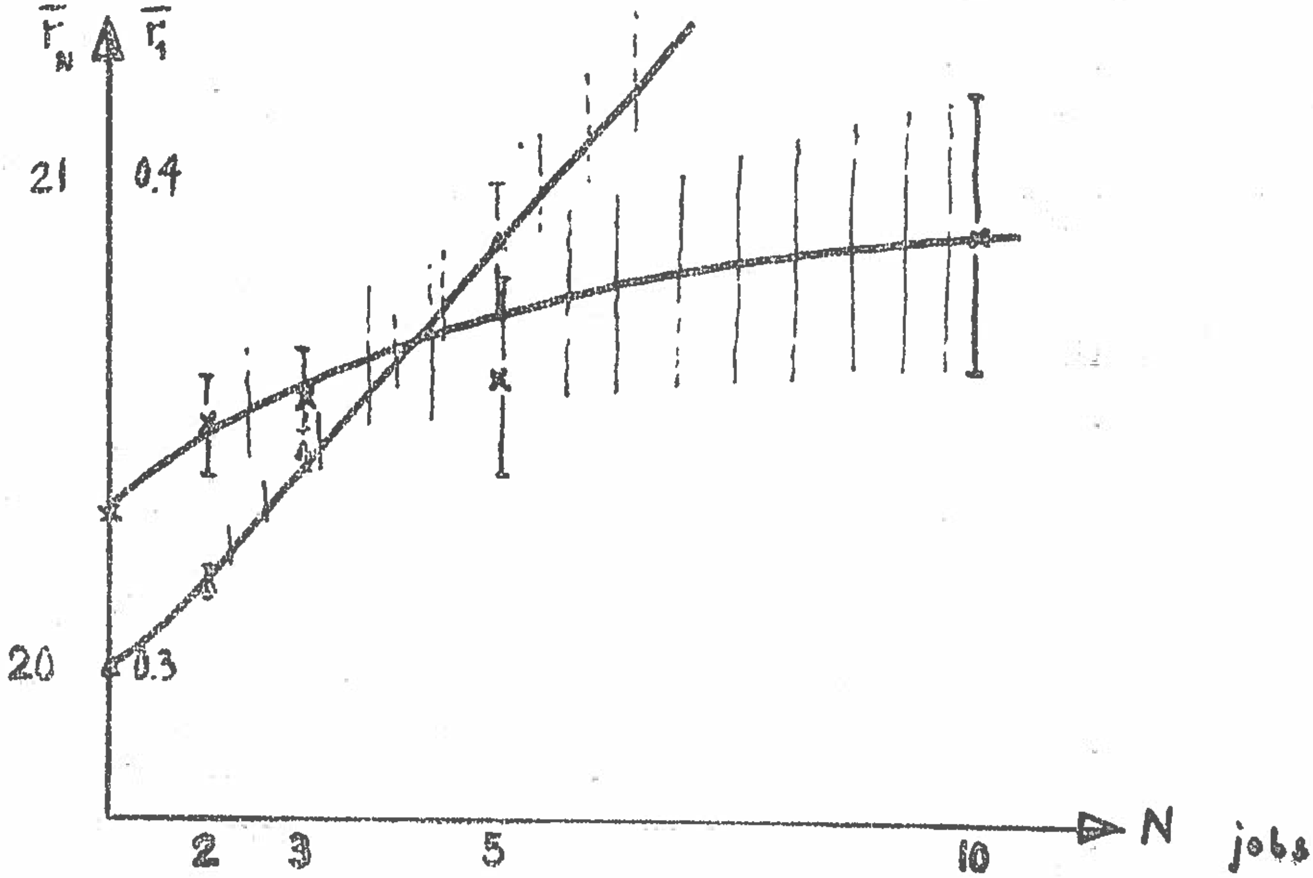
PREEMPTIVE QUEUE

$\lambda_1 = \dots = \lambda_{N-1} = 1/30$

$1/3$

$\lambda_{01} = \dots = \lambda_{0N-1} = 3.0$

$\lambda_N = 1/300$; $t_{0N} = 20.0$

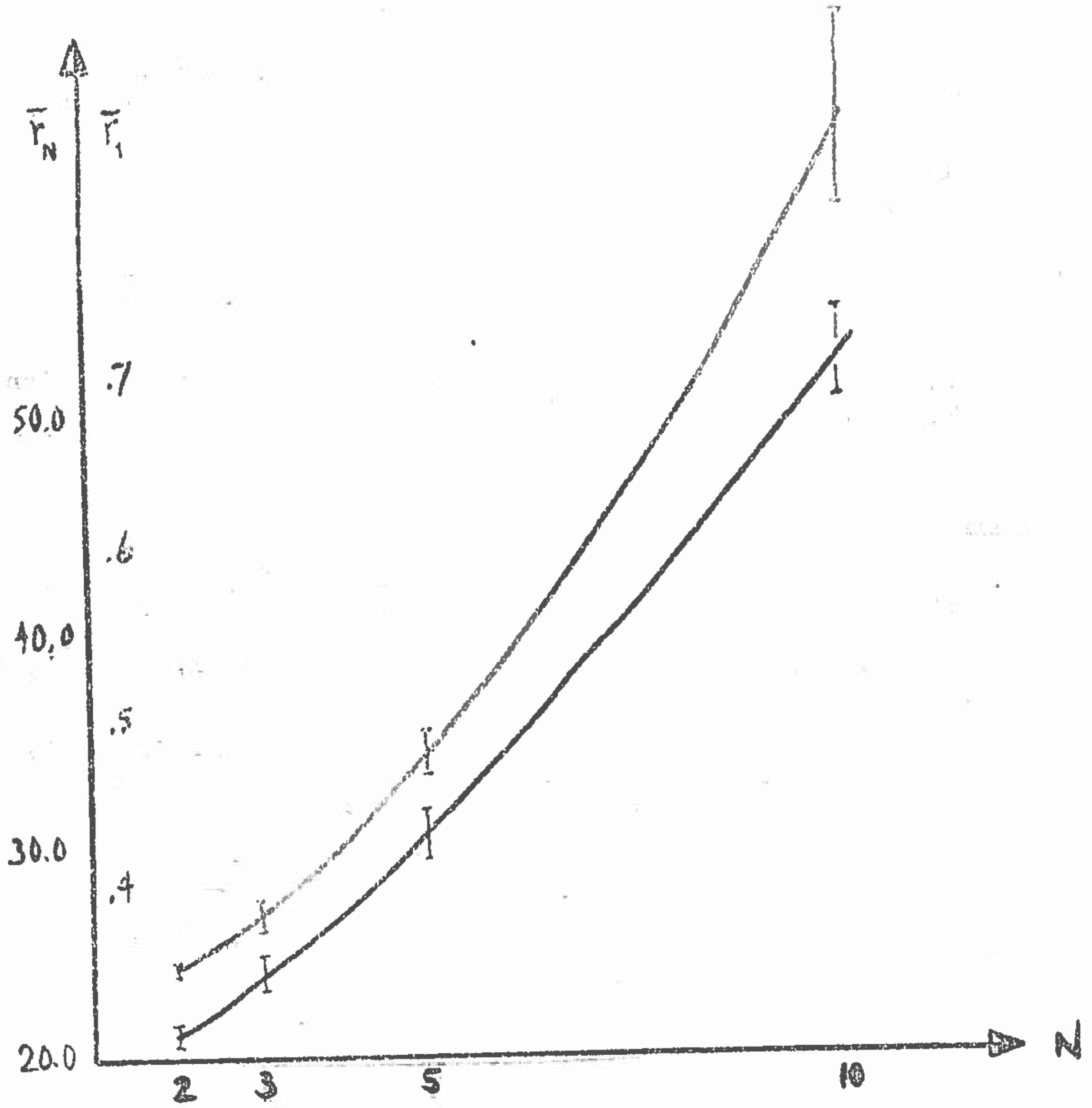


PROCESSOR SHARING

$\lambda_1 = \ldots = \lambda_{N-1} = 1/30$

$\lambda_{01} = \ldots = \lambda_{0N-1} = 3.0$

$\lambda_N = 1/300 \; ; \; t_{0N} = 20.0$

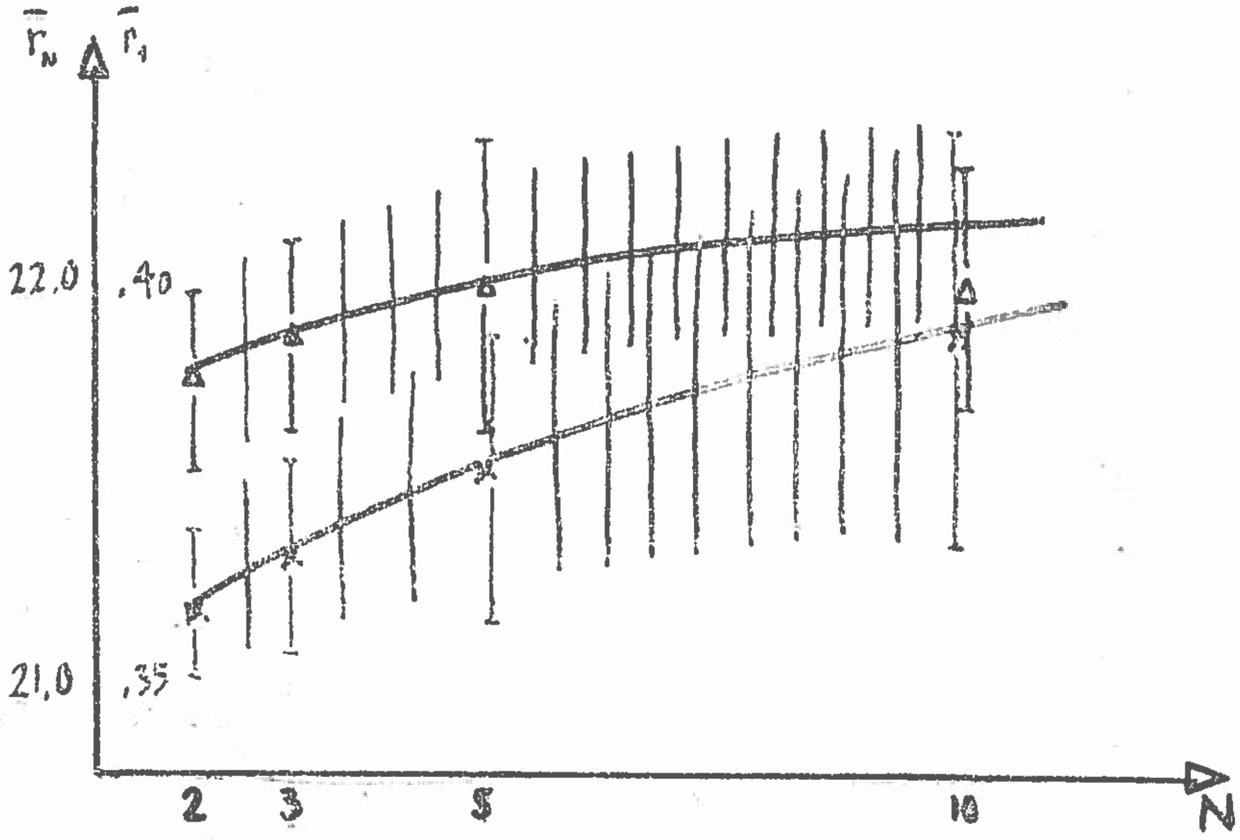


PROCESSOR SHARING

$\lambda_1 = \cdots = \lambda_{N-1} = 1/3$

$\lambda_{01} = \cdots = \lambda_{0N-1} = 3.0$

$\lambda_N = 1/300$; $t_{0N} = 20.0$



PROCESSOR SHARING

$\lambda_1 = \dots = \lambda_{N-1} = 1/3 \cdot 1/_{N-1}$

$\lambda_{01} = \dots = \lambda_{0N-1} = 3.0$

$\lambda_N = 1/300$; $t_{0N} = 20.0$

Acknowledgements

This work has been made possible by the people at IBM's LLCD lab lead by Dr. George Cheroff. Their encouragment and support is gratefully acknowledged. Knowledge of modern simulation analysis methods has been laboriously imparted on me by Dr. Gerry Shedler from SanJose Research lab, to whom I extend my sincere thanks.

References

[Crane] M.A.Crane and A.J.Lemoine, An Introduction to the Regenerative Method for Simulation Analysis, Lecture Notes in Control and Information Sci., vol.4, Springer-Verlag;

[Hamacher] V.C.Hamacher and G.S.Shedler, Collision-free local area bus network performance analysis, IBM j. Research and Development;

[Iglehart] D.L.Iglehart and G.S.Shedler, Regenerative Simulation of Response Times in Networks of Queues, Lecture Notes in Control and Information Sci., vol.26, Springer-Verlag;

[Kobayashi] H.Kobayashi, Modeling and Analysis, Addison-Wesley, 1981;

[Sauer] C.H.Sauer and K.M.Chandy, Computer System Performance Modeling, Prentice-Hall, 1981;

Appendix 1 Computation of confidence interval

For n observed i.i.d. pairs $(Y1_i, N1_i)$

$$E_n(Y1) = \frac{1}{n}\sum Y1_i$$

$$E_n(N1) = \frac{1}{n}\sum N1_i$$

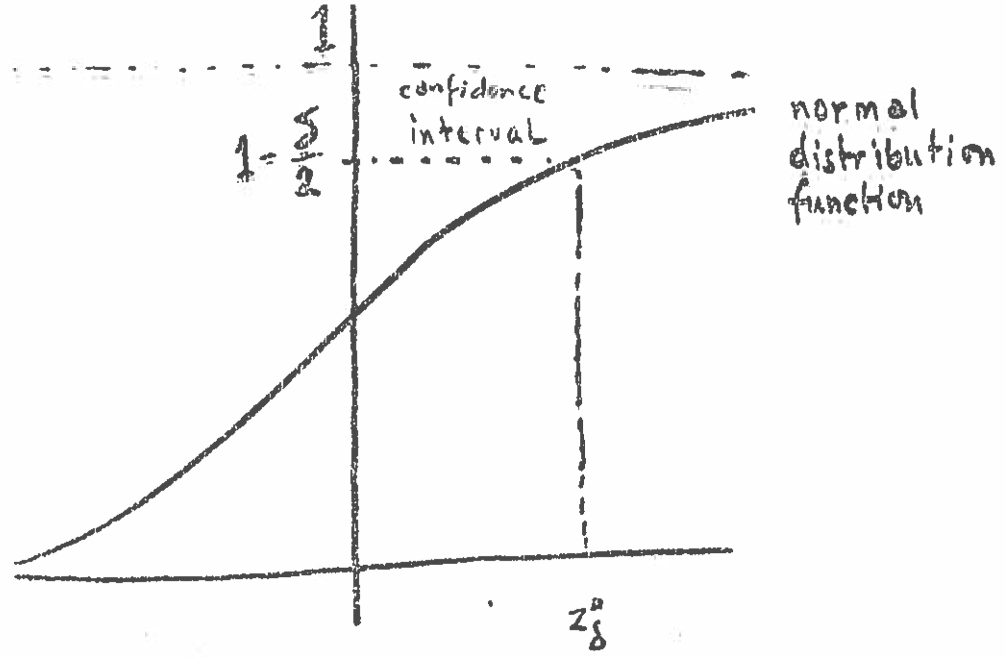
$$E_n(C1) = E_n(Y1) / E_n(N1)$$

$$s11 = \frac{1}{n-1}\sum \left(Y1_i - E_n(Y1)\right)^2 = \frac{1}{n-1}\sum (Y1_i)^2 - \frac{1}{n-1}\left(\sum Y1_i\right)^2$$

$$s22 = \frac{1}{n-1}\sum \left(N1_i - E_n(N1)\right)^2 = \frac{1}{n-1}\sum (N1_i)^2 - \frac{1}{n-1}\left(\sum N1_i\right)^2$$

$$s12 = \frac{1}{n-1}\sum \left(Y1_i - E_n(Y1)\right)\left(N1_i - E_n(N1)\right) =$$

$$= \frac{1}{n-1}\sum Y1_i N1_i - \frac{1}{n(n-1)}\left(\sum Y1_i\right)\left(\sum N1_i\right)$$



Half-width of the δ confidence interval

is given by

$$\frac{z^*_\delta \cdot s}{E_n(N1)\sqrt{n}}$$

where

$$s^2 = s11 - 2E_n(C1) \cdot s12 + E_n(C1)^2 \cdot s22$$

```
PROGRAM PS;
(*---- PROCESSOR SHARING QUEUE DISCIPLINE AT SERVICE CENTER 0 ----*)
(* SIMULATION OF A SIMPLE QUEUING NETWORK MODEL OF A COMPUTER
   SYSTEM LOADED WITH 'N_OF_JOBS' JOBS, ONE OF WHICH (THE LAST
   ONE) IS A SPECIAL HEAVY-DUTY APPLICATION. USING THE
   REGENERATIVE METHOD THE VARIABILITY OF THE EXPECTED RESPONSE
   FOR JOB 1 WILL BE CALCULATED   ------------------------ *)
CONST F_INVERSE=1.645; (*CORRESPONDING TO NORMAL DISTRIBUTION OF .9 *)
      M = 2147483647.0; A = 16807.0; (*PARAMS OF CONGRULENTIAL GENRTR*)
      LO=10.0; HI=35.5;
TYPE RANDINT = 1..2147483646;
     POINTER = -> EVENT;
     EVENT = RECORD TIME:REAL; ID:INTEGER; NEXT:POINTER END;
 VAR TRACE:CHAR; NEXT_EVENT_TIME, START : ARRAY (. 1..8 .) OF REAL;
     SIMCO:TEXT; CLOCK:REAL; CURRENT_JOB,CPU_STATE,I : INTEGER;
     CPU_Q_HEAD, TEMP:POINTER; LAMBDA:ARRAY (. 1..8 , 0..1 .) OF REAL;
     CPU_Q_LENGTH: INTEGER; Z:ARRAY (.1..10.) OF RANDINT;
     TABLE: ARRAY(. 0..127,1..10 .) OF RANDINT;
     N_OF_JOBS, N_OF_JOBS_1, CYCLES: INTEGER; MEAN_SERV_TIME: REAL;
(*-------VARIABLES FOR REGENERATIVE ANALYSIS------------*)
     RESP_IN_CYCLE,SUM_RESP,SUM_RESP_SQ,SUM_MIXED:ARRAY(.1..8.) OF REAL;
     NUM_IN_CYCLE,SUM_NUM,SUM_NUM_SQ: ARRAY (. 1..8 .) OF INTEGER;
     NUM_BAR, RESP_HAT, S11, S12, S22, CONF_INT: REAL;
(*---------------------------------------------------------*)
PROCEDURE DIALOG;
   VAR I:INTEGER;
   BEGIN WRITELN('QUEUING NETWORK MODEL OF TIMESHARED PERFORMANCE');
      WRITELN('PLEASE ENTER NUMBER OF JOBS AND RATE PARAMETERS');
      WRITELN('NO. OF JOBS?'); READLN(N_OF_JOBS); WRITELN;
      N_OF_JOBS_1:=N_OF_JOBS - 1;
      WRITELN('FOR THINK TIME..LAMBDA11?'); READLN(LAMBDA(.1,1.));
      FOR I:=2 TO N_OF_JOBS DO
      BEGIN WRITELN('                    LAMBDA1',I:2,'?');
            READLN(LAMBDA(.I,1.)) END;
      WRITELN('FOR PROCESSING..LAMBDA01?'); READLN(LAMBDA(.1,0.));
      FOR I:=2 TO N_OF_JOBS_1 DO
      BEGIN WRITELN('                  LAMBDA0',I:1,'?');
            READLN(LAMBDA(.I,0.)) END;
            WRITELN('MEAN SERVE TIME FOR JOB',N_OF_JOBS:2,'?');
            READLN(MEAN_SERV_TIME);
            WRITELN('NO. OF REGENERATIVE CYCLES?'); READLN(CYCLES);
            WRITELN('TRACE ON THE TERMINAL(Y/N)?'); READLN(TRACE)
      END; (* OF TERMINAL DIALOG *)

FUNCTION RAND(VAR Z: RANDINT ; WHICH:INTEGER) : REAL;
   VAR I,J:INTEGER;
    BEGIN
      (*Z:=(A*Z) MOD M*) Z:=TRUNC(A*Z - (TRUNC((A*Z)/M)*M));
      I:=Z MOD 128;
      RAND:=TABLE(. I,WHICH .)/M;
      TABLE(. I,WHICH .):=Z
  END;  (* RAND *)

FUNCTION DISTRIBUTION(TYPEV,JOB:INTEGER):REAL;
   VAR J:INTEGER;
```

```
    BEGIN J:=JOB+N_OF_JOBS*TYPEV;
          IF  (TYPEV=0)&(JOB=N_OF_JOBS)
           THEN DISTRIBUTION:=MEAN_SERV_TIME
           ELSE DISTRIBUTION:=-LN(RAND(Z(.J.),J))/LAMBDA(. JOB,TYPEV .)
          END;

 PROCEDURE INITIALIZE;
    VAR I,J:INTEGER;
    BEGIN Z(.1.):=377003613; Z(.2.):=646473574; Z(.3.):=1396717679;
          Z(.4.):=2027350275; Z(.5.):=1356162430; Z(.6.):=1752629996;
          Z(.7.):=745806097; Z(.8.):=201331468; Z(.9.):=1393552473;
          Z(.10.):=1966641861;
          FOR J:=1 TO 10 DO
             FOR I := 0 TO 127 DO
             BEGIN Z(.J.):=TRUNC(A*Z(.J.)-(TRUNC((A*Z(.J.))/M)*M));
               TABLE(. I,J .):=Z(.J.);
               END;
          FOR I:=1 TO N_OF_JOBS DO
          BEGIN NUM_IN_CYCLE(.I.):=0; SUM_NUM(.I.):=0;
                RESP_IN_CYCLE(.I.):=0.0; SUM_RESP(.I.):=0.0;
                SUM_NUM_SQ(.I.):=0; SUM_RESP_SQ(.I.):=0.0;
                NEXT_EVENT_TIME(.I.):=DISTRIBUTION(1,I);
                SUM_MIXED(.I.):=0.0 END;
          LAMBDA(.N_OF_JOBS,0.):=MEAN_SERV_TIME;
          CLOCK:=0.0; CPU_STATE:=0;
          NEW(CPU_Q_HEAD); CPU_Q_LENGTH:=0;
          WITH CPU_Q_HEAD-> DO
          BEGIN TIME:=0.0; ID:=0; NEXT:=NIL END
          END; (* OF INITIALIZATION *)

PROCEDURE UPDATE(VAR CLOCK:REAL; VAR JOB:INTEGER);
   VAR T:REAL; I:INTEGER;
   BEGIN T:=NEXT_EVENT_TIME(. 1 .); JOB:=1;
         FOR I:=2 TO N_OF_JOBS DO
         IF NEXT_EVENT_TIME(. I .) < T
         THEN BEGIN T:=NEXT_EVENT_TIME(. I .); JOB:=I END;
         CLOCK:=T
      END; (* OF UPDATING THE CURRENT EVENT *)

PROCEDURE STATS(JOB:INTEGER); (* COLLECTING STATISTICS ABOUT RESP TIME *)
   VAR I:INTEGER;
   BEGIN NUM_IN_CYCLE(.JOB.):=NUM_IN_CYCLE(.JOB.) + 1;
      RESP_IN_CYCLE(.JOB.):=CLOCK-START(.JOB.)+RESP_IN_CYCLE(.JOB.);
      WRITELN('LENGTH OF JOB',JOB:3,CLOCK-START(.JOB.):7:3);
      IF CPU_Q_HEAD->.NEXT->.NEXT=NIL
      THEN FOR I:=1 TO N_OF_JOBS DO
           BEGIN SUM_NUM(.I.):=SUM_NUM(.I.)+NUM_IN_CYCLE(.I.);
                 SUM_NUM_SQ(.I.):=SUM_NUM_SQ(.I.)+
                       NUM_IN_CYCLE(.I.)*NUM_IN_CYCLE(.I.);
                 SUM_MIXED(.I.):=SUM_MIXED(.I.)+
                       NUM_IN_CYCLE(.I.)*RESP_IN_CYCLE(.I.);
                 SUM_RESP(.I.):=SUM_RESP(.I.)+RESP_IN_CYCLE(.I.);
                 SUM_RESP_SQ(.I.):=SUM_RESP_SQ(.I.)+
                       RESP_IN_CYCLE(.I.)*RESP_IN_CYCLE(.I.);
                 RESP_IN_CYCLE(.I.):=0.0; NUM_IN_CYCLE(.I.):=0
```

```
                END
     END; (* OF GATHERING STATISTICS *)

PROCEDURE INSERT_CPU_QUEUE(VAR JOB:INTEGER);
   VAR P,Q:POINTER; PROCESSED:REAL; LOOKING:BOOLEAN;
   BEGIN P:=CPU_Q_HEAD; Q:=P->.NEXT;
     PROCESSED:=Q->.TIME-(NEXT_EVENT_TIME(.Q->.ID.)-CLOCK)/CPU_Q_LENGTH;
      CPU_Q_LENGTH:=CPU_Q_LENGTH+1;
      LOOKING:=TRUE;
      WHILE Q¬=NIL DO
      BEGIN Q->.TIME:=Q->.TIME-PROCESSED;
            IF (Q->.TIME>=TEMP->.TIME) AND LOOKING
            THEN BEGIN P->.NEXT:=TEMP; TEMP->.NEXT:=Q;
                        LOOKING:=FALSE END
            ELSE P:=Q;
            Q:=Q->.NEXT
            END;
      IF LOOKING THEN P->.NEXT:=TEMP;
      JOB:=CPU_Q_HEAD->.NEXT->.ID;
      NEXT_EVENT_TIME(.CPU_Q_HEAD->.NEXT->.ID.):=
            CLOCK+CPU_Q_HEAD->.NEXT->.TIME*CPU_Q_LENGTH
      END; (* OF INSERTING NEW JOB IN CPU QUEUE *)

PROCEDURE REMOVE_FROM_QUEUE(VAR JOB:INTEGER);
   VAR P,Q:POINTER; PROCESSED:REAL;
   BEGIN P:=CPU_Q_HEAD->.NEXT; CPU_Q_HEAD->.NEXT:=P->.NEXT;
         PROCESSED:=P->.TIME; CPU_Q_LENGTH:=CPU_Q_LENGTH-1;
         P:=P->.NEXT;
         WHILE P¬=NIL DO
         BEGIN P->.TIME:=P->.TIME-PROCESSED; P:=P->.NEXT END;
         IF CPU_Q_LENGTH>0
         THEN BEGIN JOB:=CPU_Q_HEAD->.NEXT->.ID;
              NEXT_EVENT_TIME(.JOB.):=CLOCK+CPU_Q_HEAD->.NEXT->.TIME*
                                           CPU_Q_LENGTH END
         ELSE JOB:=0
      END; (* OF REMOVING A JOB FROM THE CPU QUEUE *)

BEGIN (*--------- MAIN PROGRAM ----------*)

REPEAT
   TERMIN(INPUT); TERMOUT(OUTPUT);
   DIALOG;        (* SETTING OF SIMULATION PARAMETERS *)
   INITIALIZE;    (* RANDOM TABLE AND THE REGENERATION STATE *)

   WHILE SUM_NUM(.1.) < CYCLES DO
   BEGIN UPDATE(CLOCK,CURRENT_JOB);
         IF CPU_STATE = CURRENT_JOB
         THEN BEGIN IF TRACE='Y' THEN
                    WRITELN('     END PROCESSING JOB',CURRENT_JOB:2,
                                              ' AT',CLOCK:8:2);
                       NEXT_EVENT_TIME(.CURRENT_JOB.):=CLOCK+
                                         DISTRIBUTION(1,CURRENT_JOB);
                       STATS(CURRENT_JOB);
                       REMOVE_FROM_QUEUE(CURRENT_JOB);
                       CPU_STATE:=CURRENT_JOB
```

```
                   END
         ELSE BEGIN IF TRACE='Y' THEN
                 WRITELN('REQUESTING CPU FOR JOB',CURRENT_JOB:2,
                                                ' AT',CLOCK:8:2);
              NEW(TEMP); START(.CURRENT_JOB.):=CLOCK;
              WITH TEMP-> DO
              BEGIN TIME:=DISTRIBUTION(0,CURRENT_JOB);
                  ID:=CURRENT_JOB; NEXT:=NIL END;
              IF CPU_STATE=0   (* START PROCESSING *)
              THEN BEGIN NEXT_EVENT_TIME(. CURRENT_JOB.):=
                                               CLOCK+TEMP->.TIME;
                      CPU_Q_HEAD->.NEXT:=TEMP;
                      CPU_Q_LENGTH:=1;
                      CPU_STATE:=CURRENT_JOB END
              ELSE BEGIN              (*WAIT IN QUEUE *)
                      NEXT_EVENT_TIME(.CURRENT_JOB.):=M;
                      INSERT_CPU_QUEUE(CURRENT_JOB);
                      CPU_STATE:=CURRENT_JOB END
              END
        END;
(*-------------- COMPUTING CONFIDENCE INTERVAL -----------------------*)
    WRITELN(SIMCO,' CONSTANT ',
                 'APPLICATION SERVICE TIME',MEAN_SERV_TIME:6:2);
    WRITELN(SIMCO,'JOB: SERVICE RATES, MEAN RESPONSE +/- 90% CONF.INT'
                                      ,'  AND NUMBER OF CYCLES');
    FOR I:=1 TO N_OF_JOBS DO
    BEGIN S11:=(SUM_RESP_SQ(.I.)-SUM_RESP(.I.)*SUM_RESP(.I.)/CYCLES)/
                                            (CYCLES-1);
        S12:=(SUM_MIXED(.I.)-SUM_RESP(.I.)*SUM_NUM(.I.)/CYCLES)/
                                            (CYCLES-1);
        S22:=(SUM_NUM_SQ(.I.)-SUM_NUM(.I.)*SUM_NUM(.I.)/CYCLES)/
                                             (CYCLES-1);
        RESP_HAT:=SUM_RESP(.I.)/SUM_NUM(.I.);
        NUM_BAR:=SUM_NUM(.I.)/CYCLES;
        CONF_INT:=F_INVERSE/NUM_BAR*
           SQRT((S11-2*S12*RESP_HAT+S22*RESP_HAT*RESP_HAT)/CYCLES);
        WRITELN  (LAMBDA(.I,1.):6:3,LAMBDA(.I,0.):6:3,
                RESP_HAT:6:3,'  +/-',CONF_INT:6:3,CYCLES:10);
        WRITELN(SIMCO,I:2,LAMBDA(.I,1.):10:3,LAMBDA(.I,0.):6:3,
                          RESP_HAT:10:3,CONF_INT:18:3,CYCLES:10);
        WRITELN(SIMCO)
        END;
    WRITELN('ANOTHER RUN  (Y/N)?'); READLN(TRACE);
UNTIL TRACE¬='Y'
END.
```