A GPSS MODEL OF A QUEUING PROBLEM WITH COMPLEX DECISION MAKING BEHAVIOR

Leonidas C. Charalambides, Ph.D.
College of Business Administration
Marquette University
Milwaukee, Wisconsin 53233

ABSTRACT

This paper covers in detail the simulation model as well as the associated GPSS computer program of a complex queueing problem. Complexity is manifested in terms of the use of the model as well as the model itself. The former pertains to experimental design aspects. The latter, on the other hand, refers to structural considerations (i.e., parallel queueing systems sharing resources) as well as decision making behavior concerning the allocation of these scarce resources (i.e., a batch arrival and service queue with two different forms of balkling and reneging).

Following a general description of the model and program, a detailed presentation of the segment of code which represents decision making is made. Then several significant program highlights are discussed. Finally the technical considerations of the use of the program in a research investigation are described.

The paper is intended for seasoned users of simulation. These may either be research investigators or simulation practitioners who at one time or another have been challenged to bring together the modeling requirements of their problem with the modeling capabilities of GPSS.

INTRODUCTION

Since the introduction of GPSS more than twenty years ago a large number of papers describing simulation models using this language have appeared in the literature. Most of these application articles have attempted to convey to the reader an understanding of the essential features of the problem, be it an engineering, business or other type of system. They have also sought to instill some appreciation for the contribution of simulation to the "solution" of the problem ("solution" of course meaning different things to different people). By and large coverage of GPSS in these papers has been incidental, being limited to a GPSS flowchart and sometimes a small part of the code. Few such articles, a good example being Degen and Schriber(6), have taken up as their main theme the

use of GPSS in a particular class of problems. This paper belongs to the latter category of work. Thus, it seeks to demonstrate some of the ways in which the modeling capabilities of GPSS can be "stretched to their limit." By providing a considerable amount of technical detail, it is hoped that this presentation will serve as a case study of the rewards as well as penalties that a problem solver may expect to encounter when using GPSS to study a particularly challenging queueing-type problem.

In this case the problem is generalized in that it represents an abstraction of realistic situations. There is also a considerable amount of complexity. This is because, in addition to being subjected to dynamic and stochastic, environmental inputs, the system exhibits activity interdependence, i.e., the progress of a service activity is directly affected by the progress of another service activity elsewhere in the system. Therefore this problem is an example of a large category of queueing (usually network) problems where

a) in-parallel and/or in-series queueing systems affect one another's demand for and/or supply of service as well as
b) unusual customer behaviour (e.g., balkling) or server behaviour (e.g., preemptive priorities) occurs.

Such problems are encountered in production, communications and transportation, to name a few.

THE SIMULATION MODEL

The simulation model includes a number of Customer Queueing Systems (C.Q.S.) that are arranged in parallel. Each serves its own set of customers using a first-come-first-served discipline (see Figure 1). The total number of servers in the group of these C.Q.S.s--otherwise referred to as the Customer Service Echelon (C.S.E.)--is fixed. Inbound and outbound server transfers are initiated at a C.Q.S. independently of the others. Each C.Q.S. uses the same generic, heuristic, event-triggered operating policy. This
A GPSS Model of a Queueing Problem With Complex Decision-making Behavior (continued)

Policy is based upon simple feedback control of a performance criterion called the server utilization factor (u.f.). This criterion, calculated separately for each C.Q.S., is as follows:

\[
\frac{B + W}{S + T + R}
\]

where

B is the number of servers in use at a C.Q.S.,
W is the number of customers waiting to be served at that C.Q.S.,
S is the total number of servers (busy and idle) at the same C.Q.S.,
T is the number of servers in transit to that C.Q.S., and
R is the number of outstanding requests for servers from that C.Q.S. at the Depot (see below).

The policy includes four decision rules that control the timing and extent of the increase or decrease of the number of servers currently assigned to a C.Q.S. These decision rules are of the "stochastic review" type, i.e., the occasions on which the u.f. is computed and compared to the predefined control limits are the three independent events in the operation of the group. These are the events that are a function of the stochastic processes that "drive" the group. They are:

a) the arrival of a customer at a C.Q.S. (generated by the stationary Poisson arrival rate process),
b) the release of a server by a customer (generated by the stationary Erlang-k service time process), and
c) the completion of a requested inbound server transfer at a C.Q.S. (generated by the unknown request waiting and service process).

On any one of these occasions a C.Q.S. may place an order composed of one or more requests for additional servers (one request per server) with the Depot. The latter is not part of the C.S.E. but of a higher level echelon called the R.S.E., for Request Service Echelon. Thus the Depot acts as a neutral "traffic policeman" of sorts to which unneeded servers are also dispatched from various C.Q.S.s. Consequently, for statistics gathering purposes, there are four different kinds of queues in front of the Depot:

a) the order queue (from the C.S.E.),
b) the request queue (from the C.S.E.),
c) the "segmented" order queue (one from each C.Q.S.), and
d) the "segmented" request queue (one from each C.Q.S.).

It should be stressed that the Depot serves orders, not the requests that make up an order. Furthermore it is postulated that the server transfer times between the R.S.E. and the C.S.E. are deterministic and uniform for all Depot-to-C.Q.S. combinations (in both directions).

Ongoing research with this model (2) has investigated the effects upon C.Q.S. and C.S.E. performance of

a) features of the operating policy and
b) the structural parameters of the C.S.E.

For instance, under the former category, the model has been used to ascertain the control limit sensitivity of the operating policy. Experimental factors in the latter category, on the other hand, have included the size, degree of compactness and homogeneity of the group. Performance has been evaluated by means of criteria that measure various aspects of queuing system behaviour, e.g., quality of service to customers as well as utilization of servers.

Figure 2 is a flowchart of the logic of the simulation model. The first (upper) part shows the typical tasks associated with every discrete simulation model (searching the future events list, identifying the time of the next event, etc.), the three independent events mentioned above plus the two dependent events, i.e., server arrival at the Depot and end of the simulation run. The second part shows the three paths of activity that can be followed after the u.f. has been compared with the control limits. For instance, if a reduction in the u.f. denominator is indicated, the request queue and the segmented request queue associated with that C.Q.S. are checked for any waiting requests. If there are some waiting, a suitable number is cancelled ("mandatory" reneging). On the other hand, if it is desired to increase the u.f. denominator but the request queue contains as many requests as the total number of servers in both echelons, control is transferred to the portion of the program that searches the events list ("mandatory" balkings). "Mandatory" balkings and reneging should be distinguished from "optional" balkings and reneging. The "optional" characterization refers to features that are integral components of the operating policy in that they are under the jurisdiction of the imaginary "C.Q.S. manager."
As such they do not have to be invoked in every simulation run. In Figure 2 "optional" reneging is portrayed by the heavily drawn process box at the lower left hand side of the second part, while "optional" balking is depicted by the heavily drawn off-page connector on the right side of the figure.

**THE GPSS COMPUTER PROGRAM**

**General**

The digital simulation literature advocates the choice of a high level simulation language (such as GPSS) over a general purpose programming language (such as FORTRAN) because of the provision of such "built-in" housekeeping chores as maintenance of events lists, data gathering, etc. In this case these were essentially the same reasons behind the elimination of FORTRAN. On the other hand, choosing GPSS over other high level languages was prompted more by availability than anything else. Still, the preference of this investigator for the transaction orientation of GPSS was a strong motivating force. Another was readability. Bobillier et al. (1) are among the many authors who maintain that the "readability (of GPSS) is a special advantage whenever several people must work on the problem" [p. 384].

It should also be added that this feature is very helpful when one has to "overhaul", i.e., drastically modify, the program—especially after a long period of inactivity! This particular program went through at least three such overhauls. It was originally written and tested in a GPSS V environment (7). However, it subsequently had to be modified to be run under the Xerox version of GPSS which is called GPDS (for General Purpose Discrete Simulator) (9). Conversion would have been completely painless were it not for a strange problem with MACRO expansions, to this date remaining unresolved due to the almost complete lack of vendor support. Nevertheless GPDS itself is a truly outstanding product. As evidence is offered the fact that, even though it was developed in the early seventies, it possesses interactive execution capability. It is the opinion of this author (3) that for anybody with an "unusual" simulation model this feature should more than make up for any drawbacks a particular version of GPSS may have.

**Program Structure**

Transaction activity in a C.Q.S. is modeled using six distinct operating modules of code (see Table 1). A module is a group of blocks of which the first is a GENERATE block and the last is a TERMINATE block. Transactions of course assume different roles depending upon the function of the module. In the first they are customers, in the second server inventory clerks, in the third C.Q.S. decisionmakers, in the fourth order and requests for servers and in the fifth and sixth servers. Since eight queueing systems are postulated, there are a total of forty-eight C.Q.S. operating modules plus a single Depot module which models the imaginary Depot clerk. Each module collects statistics associated with the respective C.Q.S. It also contributes to the updating of the relevant C.E. statistics.

**The Operating Modules**

The first operating module is made up of four segments of code:

a) modeling of the interarrival and service distributions using ASSIGN and LOOP blocks
Model of a Queueing Problem With Complex Decision-making Behavior (continued)

b) QUEUE & DEPART blocks at C.Q.S. and C.S.E.
queues,
c) ENTER, ADVANCE & LEAVE blocks at C.Q.S.
and C.S.E. storages to simulate the number
of busy servers, and
d) LINK & UNLINK blocks to deactivate and
activate respectively customers
transactions waiting to be served.

The second operating module contains the following
three segments of code:

a) the "initialization" of the appropriate
C.Q.S. and C.S.E. storages which model
current server capacity as follows:

  ENTER   RENT1,R$RENT1
  LEAVE   RENT1,X$INCR1

assuming of course that the following
statements have been inserted elsewhere:

  STORAGE  S$RENT1,50
  INITIAL  X$INCR1,6

b) the increase of the C.Q.S. server
"capacity", upon a "signal" (to be
explained below) from the fifth module,
using LEAVE blocks as above, and
c) the decrease of the C.Q.S. server
"capacity", upon a signal from the sixth
module, using ENTER blocks as above.

<table>
<thead>
<tr>
<th>Operating Module Number</th>
<th>Function</th>
<th>Number of Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST</td>
<td>Simulation of the processing of customers by a C.Q.S. (arrival, service, exit).</td>
<td>44</td>
</tr>
<tr>
<td>SECOND</td>
<td>Stock control of all incoming and outgoing servers to and from a C.Q.S. respectively</td>
<td>32</td>
</tr>
<tr>
<td>THIRD</td>
<td>Activation of the decision rules at a C.Q.S. Sailing (optional and mandatory) plus mandatory reneging. Management of Depot queue departures due to balinking or mandatory reneging.</td>
<td>96</td>
</tr>
<tr>
<td>FOURTH</td>
<td>Simulation of the processing at the Depot of orders and requests from a C.Q.S. Optional reneging.</td>
<td>81</td>
</tr>
<tr>
<td>FIFTH</td>
<td>Simulation of the transfer of servers from the Depot to a C.Q.S.</td>
<td>19</td>
</tr>
<tr>
<td>SIXTH</td>
<td>Simulation of the transfer of servers from a C.Q.S. to the Depot.</td>
<td>26</td>
</tr>
</tbody>
</table>

In the third module a comparison of the u.f. with
the appropriate upper and lower control limits
takes place first. Then the decision making
transaction is routed to one of two HELP blocks
which use the same FORTRAN subroutine to determine
the size of the intended server capacity change.
If an increase is indicated, a message is sent to
the fourth module after the current contents of
the request queue have been checked, for possible
"mandatory" balinking. On the other hand, if a
decrease is warranted, the length of the
appropriate segmented request queue is first
checked. If it is zero, a message is sent to the
sixth module. Otherwise, first, optional reneging
of an order currently in the process of being
served is considered, followed by optional
reneging of any orders still waiting to be served
at the Depot. Obviously partial reneging of an
order is possible, i.e., only some of the requests
that comprise a particular order (whether waiting
or in the process of being served at the Depot)
may be cancelled.

The fourth module contains segments which
a) initialize the orders from the C.Q.S.,
b) QUEUE & DEPART from the appropriate order
queues and segmented order queues,
c) SEIZE & RELEASE facility SERVE, which
represents the Depot clerk,
d) represent the process of being served by
SERVE when
  1. there is currently a number of
servers at the Depot greater than or
equal to the number of requests in
the order, and
  2. there are currently not enough
servers and consequently some
requests have to wait for servers to
become available,
e) model optional reneging which is either
"ship the requested servers to the C.Q.S.
in an unlimited number of batches" or "in
only one batch," i.e., don't wait for
more servers to become available, and
f) generate the appropriate requests using
SPLIT blocks and route them in and out of
the appropriate request queues and
segmented request queues.

A facility is used to model the Depot because at
any one time only one order can be in the process
of being served, i.e., occupy the Depot until
either all the requests are transformed into
servers dispatched to that C.Q.S. or reneging--
mandatory or optional--takes place.

The fifth module generates a server and an
offspring, "parks" the parent at a user chain,
on a signal from the fourth module moves that
transaction through ENTER, ADVANCE and LEAVE
blocks to simulate shipment time and finally sends
a signal to the second module to notify it that
the server has arrived. Obviously the sixth module
is similar to the fifth except that the initiating
signal is received from the third module and the
notification signal is sent to the Depot module.
Also, along the way a signal is sent to the second
module for records updating.

Table 1 shows that the third and fourth
modules make up more than sixty percent of the
part of the program that is devoted to the
representation of a C.Q.S. To explain balking in the third module, it is necessary that the transaction flow in the fourth module be presented first. A complete description of the fourth and third modules is provided in Appendix 1 and 2 respectively. Both of them refer to C.Q.S. #1. In these and subsequent Appendices the essential blocks are shown sequentially followed immediately by a short explanation of points deemed to be necessary to the understanding of the modeling approach. It is understood of course that some blocks not essential to transaction flow, such as TABULATE for instance, have been omitted. Finally, to facilitate the reference of blocks by other blocks in the same or a different Appendix, a specific block numbering scheme is employed. Thus a two digit number in bold typeface appears on the extreme left hand side of each line of code. The first digit identifies the Appendix number while the second refers to the sequential position of that block within that Appendix.

HIGHLIGHTS OF THE PROGRAM

Intermodule Communications

As is demonstrated in Appendices 1 and 2, one of the challenges to the programming of this simulation model is the need for "communication" between modules. This is achieved by means of logic traps, i.e., strategically placed logic switches which operate in connection with transactions LINKed on and UNLINKed from user chains. That is, when the time comes for a transaction to resume its normal flow, the transaction that instigates the UNLINKing first checks to see if the switch is reset (see line 2-10 for instance). If it is, the simulation stops after the FORTRAN error flagging subroutine EROR1 is called. If it isn't, then the switch is set (line 2-11) followed by the UNLINK block (line 2-12). The latter routes the released transaction to a block which resets the switch (line 1-4) before anything else can take place. Setting and resetting the switches is so important to proper event scheduling that BUFFER blocks are liberally used (see lines 1-31, 1-37 and 2-38). As a matter of fact to ensure that the current events chain is scanned in the manner that the investigator desires, many transaction priorities are dynamically modified, e.g.,

GENERATE , 1, 45

...;

PRIORITY 30, BUFFER

PRIORITY 45

Thus these logic traps prevent activity in one module from proceeding without required activity in other modules having been completed.

Error Flagging

The error flagging subroutine does not output the exact location of the error. To do that would require the transaction to "carry" an additional parameter. Then, upon detection of an error condition, the transaction would be routed through an ASSIGN block which would put in that parameter a predetermined code indicating the error location. Subsequently the FORTRAN subroutine would output the contents of that parameter. Naturally an alternative to the use of an error flagging subroutine would be the routing of the offending transaction through a TRACE, UNTRACE and TERMINATE sequence. The advantage of this option is that the address of the previous block is part of the standard full trace output.

HELP Blocks

The limitation upon the maximum number of arguments allowed in HELP blocks (i.e., six) was a serious obstacle to get around. It was done by using two intermediate variables (V17 and V20) to provide input data to the FORTRAN subroutine that calculates the change to the u.f. denominator (see lines 2-7 and 2-17). The cost of this option is of course the additional storage required by the sixteen—since there are 8 C.Q.S.'s—variables. An alternative would be the use of two, rather than one, subroutines which are called by the GPSS program one after the other. Thus the first would handle half of the calculations and pass the intermediate result back to the GPSS program. This value would then be turned over to the second subroutine to obtain the final result. The disadvantage of this approach is the extra core required by the second subroutine as well as the penalty, in terms of CPU time, associated with an additional subroutine access.

Statistics Gathering

The most appealing feature of a special purpose language like GPSS to an investigator with limited time is the availability of standard statistics gathering entities. Sometimes this can obviously be a disadvantage in terms of high computer storage and execution requirements. However, if resource constraints are not severe, use of redundant statistics gathering entities can be a blessing when trying to verify particularly large and complex programs. Thus a model attribute, such as current queue length (QJ), may be recorded by more than one queue. For example, in this program C.Q.S. as well as C.S.E. queues are used at the lower echelon and C.Q.S. as well as R.S.E. queues at the upper echelon. Alternately the average length of a queue may be maintained in attribute QAj as well as in Thj, i.e., using a TABLE entity. Unfortunately, use of these entities may sometimes lead to certain awkward programming problems when the information contained in their attributes must be directly used in the program itself rather than be simply outputted. This is particularly true of TABLES which are the only entities capable of gathering dispersion statistics. Thus weighted means and standard deviations cannot be directly referenced. Furthermore, when unweighted means and standard deviations are referenced, they are truncated. For this investigation accuracy is so important, especially when it comes to u.f. computations, that all the entities are subjected to entry and exit counts with a much larger order of magnitude than shown elsewhere in this paper. Thus

| QUEUE | LINA, 10000 |

19
A GPSS Model of a Queuing Problem With Complex
Decision-making Behavior (continued)

is used rather than

\texttt{QUEUE LINA,1}

However, this means that additional parameters and
variables have to be created to handle the
situations, like \texttt{LOOP} control for instance, in
which the proper order of magnitude is needed. A
good example is parameter \texttt{E} which is initialized
in 1-6 and used in 1-15.

Data Capturing

The program collects data (experimental
observations) of global as well as local
performance criteria, i.e., referring to the
C.S.E. and C.Q.S. respectively, for a number of
simulation subrums. Subsequently the data is
analyzed by means of separate linear, additive,
regression models. The tabulation of this data is
handled by a separate \texttt{FORTRAN} subroutine called
at the end of each subrun. This subroutine stores
output values on a disk file in binary mode. In
this way the preparation of the master data file,
upon the completion of all the runs, is a
relatively straightforward but nevertheless time
consuming process. This is not only due to the
size of the data base but also to the fact that
the only way to output weighted \texttt{TABLE} means and
standard deviations is through the \texttt{REPORT} editor
using \texttt{TEXT} cards. The problem is that, for this
\texttt{GPSS} implementation, the device (data set
reference) number of the binary mode disk file
cannot be the same with the device number of the
\texttt{REPORT} editor. To make matters worse, the latter
can only write to a tape. Thus each simulation
run has two output files, one on disk and the
other on tape. Not only that, but due to
experimental design considerations imposed by the
investigator, the data items of the tape files
have to be inserted into, rather than added at the
end of, the respective disk files to come up with
the master data subfile pertaining to a particular
run. For a complete discussion of the data
gathering module please see Appendix 4.

Simulation Subrums

The subrums of this model are of the "time
duration" rather than the "customer arrival count"
type. According to Conway (5) the former is
appropriate for measuring attributes of permanent
entities (utilization of storages for instance)
while the latter best suits the measurement of
temporary entity attributes (for example mean
waiting time of customers). In this case arrival
count subrums can be used if, immediately after
entering the C.S.E. customer queue,

\texttt{QUEUE LINAA,1}

a transaction trips the testing of the number of
entries

\texttt{TEST L \$Q$LINAA,\$NUM,USE19}

where \texttt{USE19} is a block that increments the
appropriate arrival counter, adds another 1000 to
\texttt{\$NUM} (if that is the size of the arrival count
subrun) and then proceeds to output the
appropriate statistics. The difficulty comes in
when one attempts to reconcile the two schemes in
the same \texttt{GPSS} program. This is because of the
inability of maintaining two separate sequences of
\texttt{START} & \texttt{RESET} cards, where one sequence
selectively resets one class of statistics and the
other sequence all the rest. That is, \texttt{GPSS} does
not provide the capability of "routing" an
imaginary transaction through one or the other
kind of \texttt{RESET} card.

Macros

Due to the size as well as the repetitive
nature of the program (approximately 2600 blocks),
\texttt{MACROS} are heavily used. Thus, each operating
module, irrespective of C.Q.S., consists of
anywhere from two to eleven \texttt{MACROS}. With the use
of parallel streams of \texttt{MACROS}, one stream per
C.Q.S., proofreading as well as correcting becomes
very easy. This is achieved by editing on-line
the files that contains the \texttt{GPSS} program and using
the \texttt{EDIT} command of the computer system editor that
search the file and type out all the occurrences
of the same kind of \texttt{MACRO}, e.g., the first \texttt{MACRO}
in the second module of all the C.Q.S's. Obviously the existence of a very carefully
tought out entity naming convention is
invaluable. This applies to tags of such entities
as queues, storages and the like as well as the
labels of blocks and \texttt{MACROS}. In writing this
program particular care was taken with dummy
arguments of a \texttt{MACRO} that refer to a block label
in another \texttt{MACRO} in the same or another module.
A block labeling scheme utilizing a three letter
prefix and a two digit suffix was found
particularly helpful in tracing areas of unusually
heavy transaction flow. The prefix, admittedly
not very descriptive, pertains to the type of the
operating module, e.g., \texttt{GET} for the fourth module
and \texttt{COR} for the third. On the other hand, the
first digit of the suffix identifies the C.Q.S.
while the second pertains to the location of that
specific block relative to all the others in that
particular module.

"Good Housekeeping" Practices

Early in the development of the program it
was decided to establish and adhere to very strict
"housekeeping" practices. It was felt that, for a
program of such complexity, a logical physical
arrangement of the various categories of code
would be invaluable not only to program
verification but to subsequent modification as
well. As Table 2 shows therefore, definitions of
\texttt{MACROS} and such important entities as \texttt{FUNCTIONS}
precede \texttt{SAVEVALUE} initializations. Furthermore
the operating modules are placed separately from
such ancillary modules like the statistics
collecting and timer modules. Also the primary
(parent) transaction flow at each module is
segregated from the secondary (offspring) flow as
much as possible. Secondary flow may also refer
to transactions which place transactions on (or
remove them from) the current events chains.
Segments of code which include looping may also be
placed at a separate location of a module.

Finally it should be pointed out that a
conscious effort was made to adhere as much as possible to certain simple common sense rules like:

a) the placement of such "attention distracting" blocks as TABULATE, MARK or ASSIGN away from such key blocks as GATE LR or TEST, and

b) the placement of QUEUE/DEPART or ENTER/LEAVE blocks for local (C.Q.S.) entities in a consistent manner relative to those of global (C.S.E.) entities (see for instance lines 1-9 and 1-10 as well as 1-64 and 1-65).

RUNNING THE PROGRAM

Thus far the program has been used in several distinct research phases. A research phase is a set of experiments (simulation runs) carried out for the purpose of testing a specific hypothesis concerning the behavior of the system. To change the configuration of the model from one research phase to the next, the following modifications to the program are required:

- the initial server allocation to the Depot,
- the size of the subrun,
- the duration of the simulation, and
- the target value of the u.f. which is the midpoint between the upper and lower control limit.

Also, in the second operating module, the storage capacity of a C.Q.S. has to be established using an appropriate STORAGE block.

On the other hand two kinds of programming modification are required to change, within the same research phase, the configuration of the model from one experimental run to another. The first calls for the setting of a number of savevalues using INITIAL blocks. These savevalues represent the experimental factors the values of which specify the experimental levels in effect for a certain run. For example, the values of 0 or 1 in savevalue XPACE indicate that there are four or eight active C.Q.S.s respectively. The values of these savevalues along with the reference numbers of the run and subrun are the first data items to be written out on the binary disk file record which contains all the data for a particular experimental replication. This is because these experimental factors constitute the independent variables of the respective regression equation. There is of course one regression equation for each performance (dependent) variable that is to be analyzed.

The second type of programming alterations consists of the changes to the values of the actual GPSS entities with which a particular experimental factor is associated. All of these entities are savevalues. A description of the experimental factors, the GPSS entities used to model them, along with the maximum number for each, (i.e., equal to the maximum number of C.Q.S.s) is given in Table 3. For instance, Factor E, i.e., size of the C.S.E., is modeled by eight X$BASEs which are used in the initialization interval subfield of the GENERATE blocks in each of the eight customer service C.Q.S. modules.

<table>
<thead>
<tr>
<th>Table 2: Sequence of Program Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. MACRO Definitions</td>
</tr>
<tr>
<td>b. Tag Definitions (EQU cards)</td>
</tr>
<tr>
<td>c. FUNCTION, VARIABLE, TABLE Definitions</td>
</tr>
<tr>
<td>d. SAVEVALUE Initializations</td>
</tr>
<tr>
<td>e. Operating Modules</td>
</tr>
<tr>
<td>C.Q.S. #1 FIRST Module</td>
</tr>
<tr>
<td>C.Q.S. #1 SECOND Module</td>
</tr>
<tr>
<td>C.Q.S. #5 SIXTH Module</td>
</tr>
<tr>
<td>C.Q.S. #2 FIRST Module</td>
</tr>
<tr>
<td>C.Q.S. #2 SIXTH Module</td>
</tr>
<tr>
<td>C.Q.S. #6 SIXTH Module</td>
</tr>
<tr>
<td>C.Q.S. #8 Depot Module</td>
</tr>
<tr>
<td>C.Q.S. #8 Auxiliary Module</td>
</tr>
<tr>
<td>f. Statistics Gathering Module</td>
</tr>
<tr>
<td>g. Timer Module</td>
</tr>
<tr>
<td>h. Control Cards</td>
</tr>
<tr>
<td>i. Output Editor Cards</td>
</tr>
</tbody>
</table>

The most difficult experimental factor to model is factor G. This refers to the form of the relative frequency distribution of the four or eight (depending upon the value of factor E) C.Q.S. arrival rate means. Thus, there is an experimental requirement that the total, i.e., C.S.E.-wide, "demand for service" has to be the same for all runs of an experimental phase. In other words, the "supply of service", i.e., the total number of servers initially allocated to the C.Q.S.s should be the same for all runs of a research phase. Consequently the initial allocations of servers to the C.Q.S.s will vary from one run to another. For instance, to model a small (four C.Q.S.s) "heterogeneous" C.S.E. with Poisson arrival rate distributions, the interarrival time means of three C.Q.S.s are set at 300 time units while that of the fourth is set at 150. The initial server assignments (X$INCRj) of each of the first three are 48 while that of the fourth is 96. On the other hand, the numbers for a large heterogeneous C.S.E. are 720, 240, 20 and 60 for the first six and the remaining two C.Q.S.s respectively. These figures along with a common service time distribution mean (X$SOHLDj) of 10799 time units ensure that the mean of the C.S.E., not C.Q.S., interarrival rate distribution (X$SINTJ)—which of course is also Poisson—will be 1/60 while the total number of servers will be 240. All these values were determined prior to the execution of the simulation runs using interactively a FORTRAN program of a heuristic numerical search procedure.

Factor H stands for the presence or absence of "interaction " among the C.Q.S.s. That is, whether the C.Q.S.s are to share servers through the Depot or are to operate independently of one another. A simple way of representing this factor.
TABLE 3
THE EXPERIMENTAL FACTORS

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
<th>GPSS Entity</th>
<th>Maximum Number of Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Degree of sensitivity $X$UNOZj of the upper control limit of the operating policy</td>
<td>XH$LINLj</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>Degree of sensitivity $X$LNOZj of the lower control limit of the operating policy</td>
<td>XH$NSHPj</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>Presence or absence of &quot;optional&quot; balking</td>
<td>XH$LINLj</td>
<td>8</td>
</tr>
<tr>
<td>D</td>
<td>Presence or absence of &quot;optional&quot; reneging</td>
<td>XH$SINTj</td>
<td>8</td>
</tr>
<tr>
<td>E</td>
<td>Number of C.Q.S.s</td>
<td>XH$BASEj</td>
<td>8</td>
</tr>
<tr>
<td>F</td>
<td>Inter-echelon transportation time</td>
<td>XH$MOVE</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>Composition of Source Population</td>
<td>XH$INCRj</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XH$SINTj</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XH$HOLDj</td>
<td>8</td>
</tr>
<tr>
<td>H</td>
<td>Presence or absence of operational interaction among the C.Q.S.s</td>
<td>XH$STRAj</td>
<td>8</td>
</tr>
<tr>
<td>I</td>
<td>Variance of interarrival time distribution</td>
<td>XH$KAPA1</td>
<td>1</td>
</tr>
<tr>
<td>J</td>
<td>Variance of service time distribution</td>
<td>XH$KAPA2</td>
<td>1</td>
</tr>
</tbody>
</table>

is to strategically place a TEST block in the third operating module. This block will compare the current absolute clock with savevalue STRAj. Thus, just as with BASEj above, if STRAj contains a very large value, the decision making code will be bypassed and no requests or disposals of servers will be initiated. On the other hand if STRAj contains a very small value, but not zero due to computational complications with the u.f. when the simulation is still "cold", then decision making may take place.

Finally, factors I and J stand for the interarrival and service time distribution variance when both of them are Erlang-k respectively. To model them, all that has to be done is set savevalues KAPA1 and KAPA2 to the appropriate values of k, i.e., a large value of k for small variance and a small value of k for large variance (see Appendix 5).

Simulation run lengths have ranged from 25 to about 42 minutes of CPU time on a Xerox Sigma 9 with 512K bytes of main memory operating under the Control Program Five (CP-V) operating system. As an indication of model complexity, one thousand customers at the C.S.E. take on the average about a second of CPU time to be processed through the model (simulation run duration has been 1,300,000 time units). As a measure of size, there have been peak memory requirements of around 450K bytes. This, despite the fact that the FORTRAN library (needed by the FORTRAN data gathering subroutine) was overlaid with the GPSS load modules to obtain the relocatable object module. Table 4 shows that a significant amount of reallocation had to take place to accommodate

a) the large number of tables necessitated by the extraordinary amount of data gathering,
b) the number of blocks (which is a direct function of the number of C.Q.S.s), and
c) the amount of COMMON which is dependent upon the large number of active transactions.

Obviously a lot of "fat" had to be trimmed by employing such well-known practices as minimizing the number of parameters per transaction as well as using halfword, instead of fullword, parameters and savevalues whenever possible (see lines 1-30 and 1-32 for instance).

<table>
<thead>
<tr>
<th>TABLE 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPUTER PROGRAM ENTITY REALLOCATIONS</td>
</tr>
<tr>
<td>Entity</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Facilities</td>
</tr>
<tr>
<td>Queues</td>
</tr>
<tr>
<td>Functions</td>
</tr>
<tr>
<td>Storages</td>
</tr>
<tr>
<td>User Chains</td>
</tr>
<tr>
<td>Transactions</td>
</tr>
<tr>
<td>Logic Switches</td>
</tr>
<tr>
<td>Savevalues (fullword)</td>
</tr>
<tr>
<td>Savevalues (halfword)</td>
</tr>
<tr>
<td>Groups</td>
</tr>
<tr>
<td>Variables</td>
</tr>
<tr>
<td>Random Number Generators</td>
</tr>
<tr>
<td>Tables</td>
</tr>
<tr>
<td>Blocks</td>
</tr>
<tr>
<td>COMMON (in K bytes)</td>
</tr>
</tbody>
</table>

SUMMARY

This paper has provided a detailed description of the GPSS program of a complex queueing problem. Particular emphasis was placed upon the decision making behaviour which generates customers to a bulk arrival queueing system. The process that serves these customers also received considerable attention. Validation and verification topics were not covered inasmuch as they are addressed elsewhere (3).
REFERENCES


APPENDIX I: THE FOURTH OPERATING MODULE

The fourth module is divided into two parts: one which handles order processing at the Depot and another that carries out request processing. These two parts can be visualized as two physically distinct sections of code—i.e., representing separate transaction flow—connected by appropriate TRANSFER blocks.

Part A: Order Processing

1-1  GENERATE ,X$BASE1,1,65,5,F

BASE1 is a fullword savevalue, previously initialized, which is used to activate or deactivate a C.Q.S. #1 depending upon the value of experimental factor E in effect during a particular simulation run, i.e., either all eight C.Q.S.s are "open" or C.Q.S.s #5 through #8 are "closed."

1-2  ASSIGN 2,1

Transaction #2 will be used to identify to which C.Q.S. does this order belong (in this case it is the first).

1-3  GET10  LINK  SEVE1,FIFO

1-4  GET11  LOGIC R  17

Logic switch 17 is set in the third module (line 2-11). This of course is the way by which modules communicate, i.e., the "signals" that are mentioned in the text.

1-5  SPLIT 1,GET10

(GET10 is in line 1-3)

Create an offspring transaction which will act as an order next time one is dispatched from the third module (i.e., from the C.Q.S. to the Depot, see line 2-12).

1-6  ASSIGN 3,X$INT01

INT01 is the size of the current order which is calculated in the third module (line 2-7).

1-7  ASSIGN 4,X$INT01

Parameter #3 is used in group operations associated with waiting time only (group LINEG in lines 1-8, 1-18, 2-34, 2-36, 2-43, 2-46) while parameter #4 is used in group operations associated with time spent being served by the Depot (group WORK in lines 1-19, 1-41, 2-20, 2-23).

1-8  JOIN  LINEG

1-9  QUEUE  LINA,1

1-10  QUEUE  LINA1,1

LINA & LINA1 are the order queue and segmented order queue for C.Q.S. #1 respectively.

1-11  QUEUE  LINO,1

1-12  QUEUE  LINO1,1

LINO & LINO1 are the order queue and segmented order queue which record waiting time plus time to be served at the Depot respectively.

1-13  GETK1  SPLIT 1,GETL1,1

(GETL1 is in line 1-53)

Generate and route, in the request processing segment described below, a request which will be part of this order. Copy parameter #1 of the parent transaction only.

1-14  PRIORITY 65,BUFFER

Move that request now!

1-15  LOOP 3,GETK1

(GETK1 is in line 1-13)

Repeat the process as many times as required by the size of the order.

1-16  GATE NU  SERVE,IDL1

IDL1 is in line 3-1)

If the facility is currently tied up, "park" this order transaction at a user chain in the auxiliary
A GPSS Model of a Queueing Problem With Complex Decision-making Behavior (continued)

 module (described in Appendix 3).

1-17 SEIZE SERVE

If the Depot is free, i.e., if there is no other C.Q.S. in the process of being served, get in!

1-18 GETP1 REMOVE LINEG
1-19 JOIN WORK
1-20 DEPART LINA,1
1-21 DEPART LINA,1
1-22 UNLINK WAIT1,GETN1,P3
(GETN1 is in line 1-61)

WAIT1 is a user chain for requests residing at the request queue or segmented request queue (see line 1-57).

1-23. GET12 TEST GE XSTOKD,P4,GET15
(STOKD is a savevalue, previously initialized, which contains the current number of servers at the Depot. An alternative to the use of this savevalue would have been using a storage entity whose capacity would be redefined, everytime a change would take place, using a STORAGE card with a VARIABLE second operand.}

1-24 SAVEVALUE STOKD-P4
1-25 SAVEVALUE SHIP1,P4
1-26 GET13 ASSIGN 4-,XSSHIP1

If there are enough servers at the Depot make the appropriate records changes. SHIP1 is the number of servers to be dispatched to the respective C.Q.S. while STOKD is incremented at the Depot module (not shown).

1-27 GATE LR 16,ERROR1

Before a signal can be sent, a check has to be made. If logic switch 16 is reset, that means that a scheduling conflict exists which is evidence of a program flaw. Therefore the originating transaction is routed to block ERROR1 where an error flagging FORTRAN subroutine is called and simulation is terminated (not shown).

1-28 LOGIC S 16
1-29 UNLINK SIX1,MOV12,1

MOV12 is a block address in the fifth module which activates the dormant server as previously discussed. The LINK SIX1,FIFO block precedes the MOV12 block in that module.

1-30 UNLINK WAIT1,GET01,XSSHIP1

Remove the appropriate number of requests from "parking status" (where they are placed in line 1-

60) and direct them to the designated exit from the model (line 1-64).

1-31 BUFFER
1-32 TEST GT X$NSHIP1,1,GET19

NSHIP1 is the indicator for the type of optional reneging (see table 3). If the value is 1, then optional reneging is "off", i.e., shipment of servers from the Depot to the C.Q.S. can be in an unlimited number of batches.

1-33 TEST GT P4,0,GET17
(3ET17 is in line 1-48)
1-34 TRANSFER ,GETQ1
(GETQ1 is in line 1-38)
1-35 GET19 TEST NE P4,0,GETQ1
1-36 UNLINK WAIT1,GET01,P4
(GET01 is in line 1-64)

If optional reneging is "on", destroy the remaining requests. It should be noted that optional reneging may only take place immediately after service begins at the Depot, i.e., for all practical purposes the order which will optionally renge will spend no time at the Depot (it will be serviced "instantaneously"). On the other hand, mandatory reneging—as described in lines 2-26 to 2-33—may very well involve an order which has been already partially filled and is awaiting for more servers to become available.

1-37 BUFFER
1-38 GETQ1 RELEASE SERVE
1-39 DEPART LINO,1
1-40 DEPART LINO,1
1-41 REMOVE WORK
1-42 UNLINK WAITD,IDLE2,1
(IDLE2 is in line 3-3)

User chain WAITD, which is located in the auxiliary module, contains all the order transactions—irrespective of originating C.Q.S.—currently waiting to be served at the Depot. Therefore this UNLINK block "releases" the next order and through block IDLE2 sends it to the fourth module of the respective C.Q.S.

1-43 TERMINATE
1-44 GET15 TEST G X$STOKD,0,GET17

This is where an order transaction is routed if there are fewer servers at the Depot than (remaining) requests in the order.

1-45 SAVEVALUE SHIP1,X$STOKD
1-46 SAVEVALUE STOKD,0
1-47 TRANSFER ,GET13
(GET13 is in line 1-26)
If logic switch #100 is not set, that means that no servers have arrived at the Depot yet. Therefore the order transaction should be routed to block SITL in the auxiliary module to be placed in a special user chain called BORED.

When servers do become available, the Depot module sets logic switch #100, UNLINKs the transaction from BORED and sends it to block GET18.

Inasmuch as logic switch #100 can also be "tripped" from the third module (line 2-27), i.e., to simulate mandatory reneging of an order in the process of being served, logic switch #13 is inserted here to find out the origin of the transaction that set switch #100. Therefore logic switch #100 is a global entity (associated with the order queue) while logic switch #13 is a local entity (associated with the segmented order queue belonging to C.Q.S. #1).

If switch #100 were set in the third module, then there is only need to remove the order transaction from the model through the appropriate order queues.

APPENDIX 2: THE THIRD OPERATING MODULE

The third operating module starts off the same way as the fourth:

2-1 GENERATE ,X$BASE1,1,45,2,F
2-2 COR10 LINK ONE1,FIFO
2-3 COR11 LOGIC R 11

Logic switch #11 is tripped by the three independent events that were identified in the section that describes the simulation model. Two of these, i.e., customer arrival and departure, take place in the first module while server arrival at the C.Q.S. is recorded in the second module.

2-4 SPLIT 1,COR10
2-5 TEST C V16,X$UNKZ1,COR14 (COR14 is in line 2-16)

Variable #16 has been previously defined as the u.f. of C.Q.S. #1 while UNKZ1 is a savalue which has been initialized to the appropriate level of the upper control limit (see Table 3).

2-6 TRANSFER FN,17

Through the use of function #17 control is transferred either to block COR12 or block COR13. In the former no optional balking may take place while in the latter optional balking is allowed. The X-coordinate values of this function are the current contents of XHS$LINLj (see Table 3).

2-7 COR12 HELPFL GPDF1,XSTARG1,X$USE1,V17, X$INTO1,0,V20

Five of the arguments of FORTRAN subroutine GPDF1 are used for input purposes. The only output argument is X$INTO1 which contains the intended order size.

2-8 TEST NE X$INTO1,0,COR1 (COR1 is in line 2-47)

If the intended order size is zero, remove the transaction from the module.

2-9 TEST L Q$LINED,X$INCR,BALKT

INCK is a savalue which contains the number of total servers in the C.S.E. Therefore if the size of the request queue is equal to that number, the transaction is removed from the program through block BALKT (mandatory balking). The latter is in a SSAVEVALUE block which accumulates the total number of mandatory balking occasions (for the C.S.E.).
A GPSS Model of a Queuing Problem With Complex
Decision-making Behavior (continued)

2-10 GATE LR 17,EROR1
2-11 LOGIC S 17
2-12 UNLINK SEVE1,GET11,1
(SET11 is in line 1-4)

Activate the dormant order transaction in the
fourth module.

2-13 TRANSFER ,CORY1
(CORY1 is in line 2-47)
2-14 COR13 TEST E Q$SERV,O,CORAL

According to the form of "optional" balking used
in this program, the order will not join the order
queue if there happens to be any other order
either waiting to be served or actually being
served at the Depot. CORAL is a TERMINATE block
(not shown) which is used to record the number of
optional balking occurrences, i.e., N$CORAL.

2-15 TRANSFER ,COR12
(COR12 is in line 2-7)
2-16 COR14 TEST L V16,X$LOZ1,CORY1

LNOZ1 is a savevalue which contains the lower
control limit for the particular simulation run.

2-17 COR15 HELPF
GDPF1,X$STAR11,SS$SS11,V17,
X$OUT1,RS$RENT1,V20
2-18 TEST NE X$OUT1,0,CORY1

OUT1 is the size of the intended reduction to the
u.f. denominator at the C.Q.S.

2-19 TEST G Q$SERV,O,CORX1
(CORX1 is in line 2-55)

If there are no requests from this C.Q.S. at the
Depot, waiting or being served, there is no
possibility for "optional" reneging. Therefore
the transaction is routed to a segment (see below)
which sends a signal to the sixth module.

2-20 SCAN WORK,2,1,4,1,COR17

If there are requests at the Depot, the first
thing to do is to check and see if there are some
currently being served, i.e., if there is an
outstanding order from that C.Q.S. Thus group
WORK is scanned for the first transaction from
C.Q.S. #1, i.e., a transaction that has the value
of 1 in parameter #2. If such a transaction is
found, the contents of its parameter #4 (the
current size of the order) are placed in parameter
#1 of the scanning transaction. Block COR17 (in
line 2-34) is the alternate exit in case such a
transaction is not located.

2-21 TEST L X$OUT1,PL,COR16

If there are fewer requests being served than
X$OUT1, go to block COR16 (in line 2-26).

2-22 ASSIGN 1-,X$OUT1
2-23 ALTER WORK,1,4,PL,2,1

Modify the current size of the order at the Depot
by passing the contents of parameter #1 of the
scanning transaction to parameter #4 of the first
transaction encountered in group WORK which
happens to have the value 1 in parameter #2.

2-24 UNLINK WAT1,GET1,PL
(GET1 is in line 1-64)

Remove the requests that are being cancelled from
the program.

2-25 TRANSFER ,CORY1
2-26 COR16 GATE LR 100,EROR1
2-27 LOGIC S 100
2-28 GATE LR 13,EROR1
2-29 LOGIC S 13

If the size of the order currently being served is
smaller than X$OUT1, the entire order must be
cancelled. Therefore the proper set of signals
must be prepared to be sent to the fourth module
(see lines 1-49 to 1-51).

2-30 UNLINK WAT1,GET1,PL
(GET1 is in line 1-64)
2-31 SAVEVALUE OUT1-,PL
2-32 UNLINK BORED,SIT2,1
(SIT2 is in line 3-6)

The signal will be conveyed by the order
transaction which will be removed from user chain
BORED and, through block SIT2 in the auxiliary
module, sent to block GET18 in the fourth module
(line 1-49).

2-33 TEST G X$OUT1,0,CORBI

If there is no need for any further request
cancellations, remove the transaction through
CORBI which is a TERMINATE block (not shown).

2-34 COR17 SCAN LINEG,2,1,3,1,CORX1

If there is a need for further request
cancellations, see if there is an order in group
LINEG from C.Q.S. #1, i.e., an order that has yet
to receive service. It should be noted that
parameter #3, rather than #4 as in scanning WORK
above, is used because the value of parameter #3
does not change every time a batch of servers
arrives at the Depot to satisfy the remainder of
an order, as of course parameter #4 does.

2-35 TEST LE P1,X$OUT1,COR18
(COR18 is in line 2-42)
2-36 REMOVE LINEG,1,,2,1

If the size of this order is smaller than the
current (remaining) size of the reduction to the
u.f. denominator, terminate that order transaction
from membership in group LINEG.

2-37             UNLINK    WAITD,COR19,1,2,1
Also remove that order transaction from user chain
WAITD (which it entered through line 1-16) and
route it to block COR19 (line 2-48).

2-38             BUFFER
2-39             SAVEVALUE OUT1,-PI
2-40             TEST G X$OUT1,0,COR1
2-41             TRANSFER ,COR17
                   (COR17 is in line 2-34)
If there is no need for more request
 cancellations, TERMINATE the transaction through
block COR1 (not shown); otherwise go back to see
if there are more orders from C.Q.S. #1 waiting in
line.

2-42             COR18 ASSIGN 1-,X$OUT1
2-43             ALTER     LINEG,1,3,PI,2,1,ERROR1
2-44             ALTER     LINEG,1,4,PI,2,1,ERROR1
If the remaining size of the u.f. denominator
reduction is smaller than the current size of the
order, modify the order accordingly, i.e., change
parameters #3 and #4.

2-45             UNLINK    WAIT1,GETN1,X$OUT1
2-46             UNLINK    WATE1,GETO1,X$OUT1
Destroy the appropriate number of requests (GETN1
is in line 1-61 while GETO1 is in line 1-64).

2-47             CORY1 TERMINATE
2-48             COR19 DEPART LINA,1
2-49             DEPART LINA1,1
2-50             DEPART LINO,1
2-51             DEPART LINO1,1
2-52             UNLINK    WAIT1,GETN1,P4
2-53             UNLINK    WATE1,GETO1,P4
2-54             TERMINATE
Block COR19 is entered by an order transaction
from the fourth module, not a decision making
transaction from the third. Consequently this is
a segment that in a way "belongs" to the fourth
module.

2-55             CORX1 GATE LR 12,ERROR1
2-56             LOGIC S 12
2-57             UNLINK    TWOL,GIV12,1
2-58             TERMINATE
Logic switch #12 is reset by a server transaction
going through block GIV12 in the sixth module.

APPENDIX 3: THE AUXILIARY MODULE

The auxiliary module is a segment of the program
where code which is common to all the members of
the C.S.E is segregated. The blocks which are
relevant to the third and fourth operating modules
are the following:

3-1             IDLE1 PRIORITY 69
3-2             LINK    WAITD,FIFO
3-3             IDLE2 SEIZE    SERVE
3-4             TRANSFER FN,25
3-5             SIT1 LINK    BORED,FIFO
3-6             SIT2 TRANSFER FN,26
The function selection mode (FN) of the TRANSFER
block is used to direct the order transaction that
is being UNLINKed back to the C.Q.S. to which it
belongs. Discrete numerical value functions have
to be used. For instance function #25 is defined
as follows:

25             FUNCTION P2,D8
               1,GETP1/2,GETP2/3,GETP3/4,GETP4/......
               8,GETP8/
where GETP1 returns the order transaction to
C.Q.S. #1 (line 1-18), GETP2 to C.Q.S. #2, GETP3
to C.Q.S. #3 etc.

APPENDIX 4: THE DATA GATHERING MODULE

The outputting of data onto the binary disk
file takes place in a separate module, the first
block of which is naturally a GENERATE block,
i.e.,

GENERATE X$STAT,,,1,19,0

STAT contains either the value of zero, for the
option of recording statistics, or a very large
value for the option of not recording, as in test
runs for instance.

ADVANCE 100000

This block simulates the duration of the transient
period.

SPLIT 1,STAT1
TERMINATE 1

The termination count is of course linked to a
sequence of START and RESET blocks.

STAT1 SAVEVALUE COUNT+,1
COUNT is the subrun counter.

ADVANCE X$BLOCT
BLOCT is the duration of the subrun. Next to
To model the generation of C.Q.S. customer transactions according to the Erlang-k distribution in the first operating module, the following blocks may be placed after the GENERATE $,X$BASE1,1,35,4,F block:

```
USE10 SPLIT 1,USE16
PRIORITY 34,BUFFER
USE16 ASSIGN 3,0
ASSIGN 4,XH$KAPAL
```

where KAPAL is the type of the Erlang-k distribution and

```
USE17 ASSIGN 3+,V97,1
```

where V97 contains the ratio of the mean customer interarrival time over KAPAL.

```
LOOP 4,USE17
ADVANCE *3
TRANSFER ,USE10
```