CSE '07

I. BASIC CONCEPTS

9.1 INTRODUCTION

Simulation analysis is a natural and logical extension to the analytical and mathematical models inherent in operations research. The previous eight chapters of this text have been concerned with formulating decision models which not only closely approximate the real-world environment, but also produce answers through standard numerical and/or mathematical manipulations of the resulting equations. It is evident that there are many situations which cannot be represented mathematically due to the stochastic nature of the problem, the complexity of problem formulation, or the interactions needed to adequately describe the problem under study. For many situations defying mathematical formulation, simulation is the only tool that might be used, to obtain relevant answers.

The word simulation has been used rather loosely in the preceding discussion, so before we proceed it will be necessary to develop a working definition of this term. The following definition has been adopted from Naylor et al. (41).

> Simulation is a numerical technique for conducting experiments on a digital computer, which involves certain types of mathematical and logical relationships necessary to describe the behavior and structure of a complex real-world system over extended periods of time.

Simulation has often been described as the process of creating the essence of reality without ever actually attaining that reality itself. Within the context of this chapter, simulation will involve the construction, experimentation, and manipulation of a complex model on a digital or analog computer. For the sake of discussion purposes, the techniques described in this chapter are directly applicable to a high-speed digital computer, although they certainly will not be restricted to this mode of operation.

Although simulation is often viewed as a "method of last resort," often to be employed when all else fails, recent advances in simulation methodologies, software availability, and technical developments have made simulation one of the most widely used and accepted tools in systems analysis and operations research. In addition to the reasons previously stated, Shannon (57) has

suggested that simulation analysis might be appropriate for the following reasons.

1. Simulation makes it possible to study and experiment with the complex internal interactions of a given system whether it be a firm, an industry, an economy, or some subsystem of one of them.
2. Through simulation, one can study the effects of certain informational, organizational, and environmental changes on the operation of a system by making alterations in the model of the system and by observing the effects of these alterations on the system's behavior.
3. A detailed observation of the system being simulated may lead to a better understanding of the system and to suggestions for improving it, which otherwise would be unobtainable.
4. Simulation can be used as a pedagogical device for teaching both students and practitioners basic skills in theoretical analysis, statistical analysis, and decision making.
5. The experience of designing a computer simulation model may be more valuable than the actual simulation itself. The knowledge obtained in designing a simulation study frequently suggests changes in the system being simulated. The effects of these changes can then be tested via simulation before implementing them on the actual system.
6. Simulation of complex systems can yield valuable insight into which variables are more important than others in the system and how these variables interact.
7. Simulation can be used to experiment with new situations about which we have little or no information, so as to prepare for what may happen.
8. Simulation can serve as a "preservice test" to try out new policies and decision rules for operating a system, before running the risk of experimenting on the real system.
9. For certain types of stochastic problems the sequence of events may be of particular importance. Information about expected values and moments may not be sufficient to describe the process. In these cases, simulation methods may be the only satisfactory way of providing the required information.
10. Monte Carlo simulations can be performed to verify analytical solutions.
11. Simulation enables one to study dynamic systems in either real time, compressed time, or expanded time.
12. When new elements are introduced into a system, simulation can be used to anticipate bottlenecks and other problems that may arise in the behavior of the system.

Simulation analysis always begins with a representation of the system which needs to be studied. In digital computer simulation this step is accomplished through the construction of a computer program which "describes" the system under study to the appropriate computer configuration. This representation might be in the form of a FORTRAN program, graphical interactive displays, or a complex simulation language. Once this step has been completed, the model of the system is acted upon and the results of these actions are observed over long simulated periods of time. In essence, the experimenter is acting upon the created model rather than the actual system itself. Although simulation analysis is usually performed via high speed computer, such a representation is not always needed to study complex problems in operations research. Examples 9.1 and 9.2 in this chapter clearly illustrate this premise through two "hand" simulations. Regardless of the mode used to perform simulation analysis, a great deal of experience is desirable in order to adequately exploit the real powers of simulation. This background is often best

unique skills in this area. For this reason, simulation modeling is often more of an "art" than a science. This art is best cultivated rather than taught, although the basic tools and modeling logic can be gained through diligent study of simulation methodologies.

9.2 THE PHILOSOPHY, DEVELOPMENT, AND IMPLEMENTATION OF SIMULATION MODELING

Simulation is one of the easiest tools of management science to use, but probably one of the hardest to apply properly and perhaps the most difficult from which to draw accurate conclusions. With the widespread installation of digital computers, simulation is readily available to most managers and engineers engaged in operations research activities. With a reasonably knowledgeable programmer, a general purpose simulation language like GASP IV (45) (which costs less than $200 for the manual and master deck of cards), and a FORTRAN compiler, the manager is ready to attack complex problems with digital simulation. However the skills required to develop and operate an *effective* simulation model are substantial. The variability or dispersion of simulation results is a significant problem in itself and may require long and complex simulation analysis in order to draw meaningful conclusions from the simulation.

Regardless of these drawbacks, simulation is a useful technique and one that is particularly appropriate for complex operations research and systems analysis. Many of the recent textbooks on simulation give example applications of simulation to a wide variety of operational types of problems. [See Chorafas (6), Emshoff and Sisson (8), Martin (35), Meier, Newell, and Pazer (38) and Pritsker and Kiviat (44), for examples.]

It is the purpose of this chapter to examine the process of simulation and the necessary tools to perform such analysis. A special emphasis will be placed upon the problems associated with the mechanics of simulation modeling itself.

The Simulation Process

It is appropriate to examine the entire process by which simulation analysis is planned and performed. Design of the simulation model itself is a critical portion of any study, but not the only one with which the user must be concerned. The activities shown in Figs. 9.1, 9.2, and 9.3 are the major ones in any simulation study. Figure 9.1 covers the presimulation tasks while Figs. 9.2 and 9.3 are the actual simulation activities, broken down into design and operational groups of activities.

Presimulation Activities

In any general view of situations in which systems simulation is used, the first activity is a recognition of the problem. This recognition leads directly to the study and analysis of the system itself and culminates in the establishment of an objective directed toward solving the problem. Typical objectives could be categorized as follows:

Characterization of System Performance
- Selection of operation parameters for an existing system
- Selection of operation parameters for a proposed system
- Exploration of System Behavior

Modification of an existing system

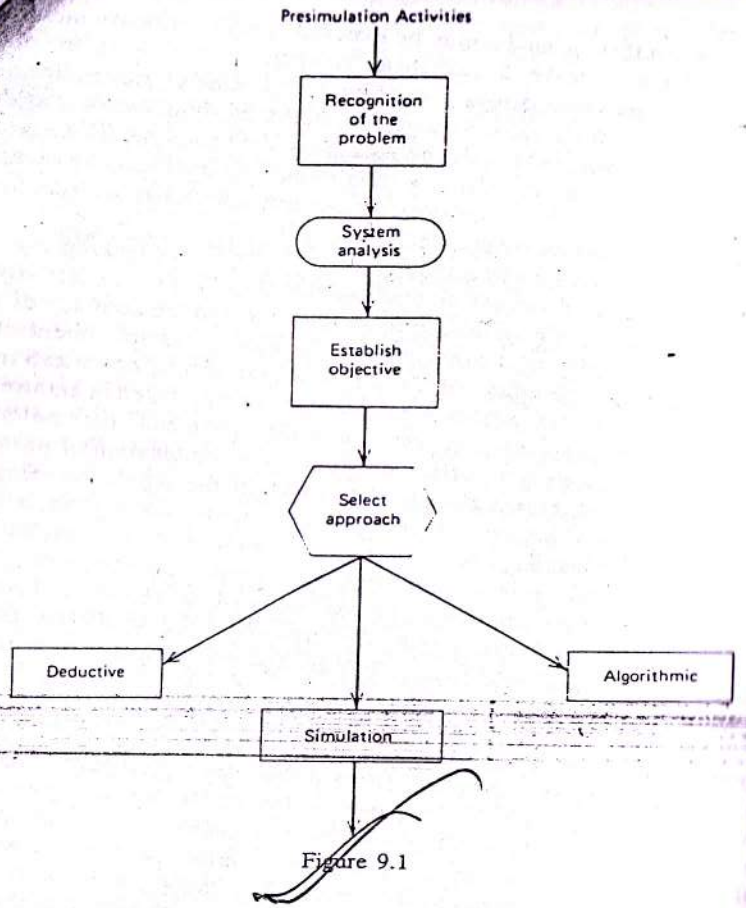


Figure 9.1

At this stage in the process, the user must evaluate the different tools or techniques available to him relative to his objective and the system with which he is dealing. If the problem is one of establishing optimal parameters and the system fits or can be made to fit one of the available techniques previously discussed in this text, then this is obviously the most appropriate technique to use. The user could also develop his own mathematical model if none of the textbook methods apply to his problem. These models are generally developed along two main lines of thought; *deductive* models and *algorithmic* models. Economic order quantities (EOQ) in inventory control techniques are examples of deductive solution methods. Algorithmic techniques are inductive, iterative techniques for developing numerical solutions to specific problems. Linear programming is an example of a technique in which an algorithm, usually the simplex algorithm, leads to an optimal solution. If the problem is inherently an optimization problem, and can be cast in a framework for which there is such an algorithm, then that approach might be preferred over simulation. Still another alternative is to experiment with the operating system itself. There are techniques, such as evolutionary operation (EVOP) which provide a means of systematically evaluating changes in the operating parameters of the

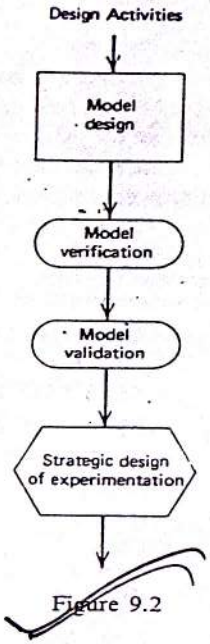


Figure 9.2

system. This approach should be appropriate in dealing with systems that are highly responsive to parameter changes.

Simulation is the appropriate technique where it is not feasible to experiment on the system itself or where direct analytic techniques are not available. In the first case, it may be too expensive to experiment with the existing system. Such experimentation might change the operating characteristics of the system itself, and thus the risk that such experimentation might harm the system's performance is so great that the possibility is discarded. It is also possible that the system is just not available for experimentation. Alternatively, if the problem is to design an inventory control system for a planned facility, the system does not even exist prior to the time it is needed. In the second case, the complexity of many production systems precludes the application of analytic techniques, either deductive or algorithmic. If the system contains many stochastic elements of a complex nature, the resulting model probably resists analytic treatment. The two situations just discussed, where experimentation on the real-world system is not possible and where appropriate analytical techniques are not available, are the general *raison d'etre* behind any simulation study. This rationale suggests the following definition of simulation:

A technique of problem solving based upon experimentation performed on a model of the real-world system.

Developmental Activities

The first developmental activity is the *design and implementation* of the simulation model. The specific tasks in this activity will not be discussed in detail in the remaining sections of this chapter. However, an excellent discussion of this phase is found in many simulation textbooks [Emshoff and Sisson (8), Meier et al. (38), and Naylor et al. (41)]. Assuming for the time

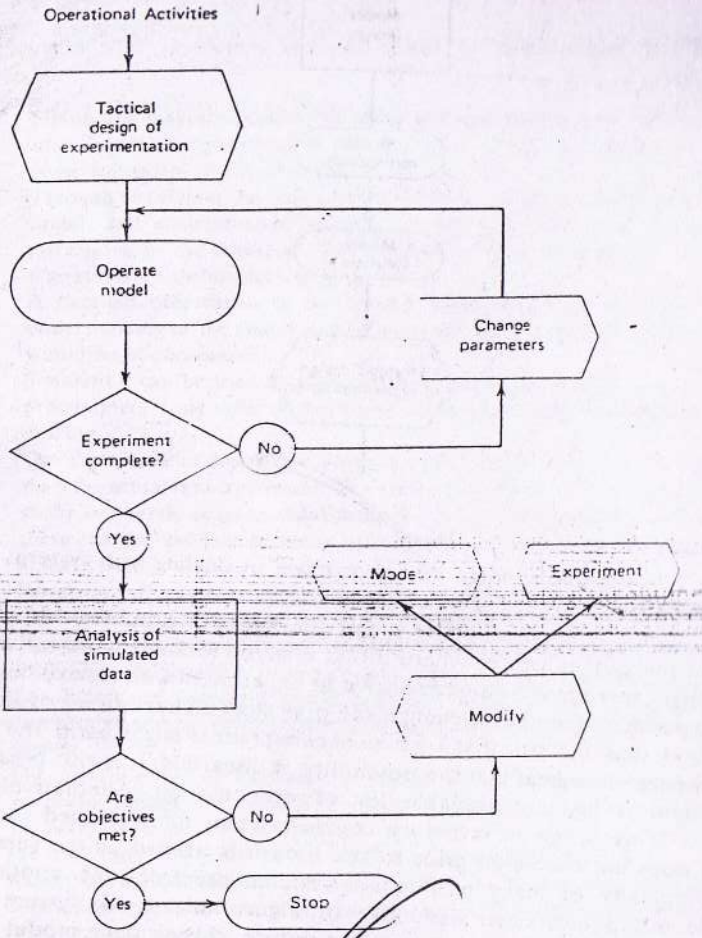


Figure 9.3

being that these tasks are completed, the next activity is *verification* of the model. A verified model is one that has been proven to behave as its designer intended. This is an important activity in that without satisfactory and explicit verification, it is possible to have a model that appears to work satisfactorily but which gives answers that are actually erroneous. Fishman and Kiviat (9) suggest techniques for verification, including statistical methods, that go far beyond the usual practices of simple comparative analysis and manual checking of calculations.

Fishman and Kiviat (9) also describe methods of *model validation*. Validation is one of the most critical activities performed in any simulation study. It is also one of the most difficult to accomplish satisfactorily. A validated model is one that has been proven to be a reasonable abstraction of the real-world system it is intended to represent. The usual approach to validation is to run

because of the fact that the model may be experimental or predictive in nature. It is also difficult to make a statistical comparison of results in many situations where such a comparison is appropriate, because of the requirement that equilibrium be reached before results are measured. It may take considerable time to reach equilibrium with the computer simulation model while the real-world system might never exist in a state of equilibrium—thereby severely complicating the comparison. Additional suggestions on validation techniques are provided by Van Horn's chapter in Naylor (57).

Strategic design refers to the activity of designing and planning the experimentation to be done with the simulation model and is the next activity. It includes specification of information to be determined and the accuracy of that information. The two classes of experimentation, as in the establishment of an objective, deal with exploration of system behavior and/or optimization of system parameters. Exploration of system behavior is undertaken in an attempt to explain the relationship between results and the controllable parameters of the simulation. Optimization is performed to find the combination of parameter levels which minimize or maximize the results of the simulation. Experimental designs, like full factorial, fractional factorial, etc., are appropriate for exploration experimentation. For optimization, optimum seeking techniques are available, and though in many cases they cannot guarantee a global optimum, they can give good practical results. The article by Hunter and Naylor (22) is a particularly good outline of experimental design for simulation studies, while Schmidt and Taylor (49) have a good discussion on simulation optimization techniques.

Operational Activities

At this stage in the simulation process the model has been designed, implemented, studied, and its use has been planned. The remaining activities are to carry out the actual simulation experimentation. This must include tactical design of the experiments that are to be performed as the first activity. Conway (7) describes this activity as determining how many simulation runs are to be executed and how data are to be collected from each run. This includes establishing initial conditions for variables in the model and estimating parameters so that the simulated system will reach a state of equilibrium as soon as possible. The user must also determine how equilibrium will be recognized so that data can be gathered without being biased by transients from start up of the run. Other considerations are the sample size required for the data to be gathered and the techniques to be used to compare alternative systems if this is the study objective. In this latter case, the user will be interested in relative results from the simulation runs. He can apply methods such as using the same sequence of random numbers for each run, which will reduce the residual variation between sets of results and thus will permit a reduction in length of the simulation runs themselves.

The simulation runs themselves can be represented as a loop, as shown in Fig. 9.3. In this loop, the model is run for the specific time, parameters changed, and the model run again. This is repeated until the specific experimentation has been completed. This is followed by an analysis activity in which the simulated data are processed and statistics are developed. Techniques such as regression analysis and analysis of variance are widely used

If the objective has been met, the simulation study is completed at this point. However, because simulation is a trial-and-error process, it will often be that the objective has not been satisfied, which leaves two general alternatives available. The first is to modify the model so that it will facilitate discrimination among simulated systems and then to rerun the experiment. The second alternative is to use the original model but to alter the design of the experimentation, using new search techniques or more powerful experimental designs.

9.3 DESIGN OF SIMULATION MODELS

A well-designed model is a key to meaningful simulation results. If the model builder overlooks the requisites of simulation that have previously been discussed, the simulation study may fall far short of accomplishing its objectives, or may indicate a false achievement when there is none. Likewise, if the model fails to take into consideration the important aspects of the real-world system, then the model cannot be validated. If such a model is used without validation, predictions of system behavior based on the model may be erroneous. The discussion of model design presented here will consider two groupings of design consideration: simulation technique related design considerations and practical restrictions due to computer software availability.

Simulation Related Considerations

The topics under this heading are generally given rather complete coverage in texts on simulation. Also, there may not be a design consideration if the user employs one of the generally available simulation languages. Each will be subsequently discussed in this chapter. The topics are:

- Time control
- Random number generation
- Stochastic variate generation
- Variance reduction techniques
- Simulation languages

Time control is generally divided into two classes, though Nance (58) advances the idea of a continuous time flow mechanism. The two typical types are *uniform time flow* and *variable time flow*. With uniform time flow, the model is advanced and processed through each and every time period simulated at fixed steps or intervals. Variable, or next-event, time-flow mechanism causes time to be incremented between only those periods that have events occurring. The model is processed only at these times and performance figures adjusted to account for the periods skipped. If the system has events that occur very frequently and with predescribed regularity (as in forecasting), then uniform time flow might be the best choice. However, if the events are irregular, then the time periods skipped will reduce computer running time and justify the additional programming generally required for variable time flow.

Simulation is a useful tool because of its ability to handle complex systems that require the modeling of interacting stochastic variates. These are the means for modeling empirical or theoretical distributions of real-world parameters. Stochastic treatment of relevant events greatly increases the realism and applicability of the simulation model. These stochastic variates are generated by manipulation of pseudorandom numbers calculated by the computer. Most computer facilities include such a generator in their software libraries, or can easily develop one. However, both random-number generation and stochastic-variate generation will be discussed in some detail as they are fundamental to any simulation analysis. Variance reduction methods, the next of the technique related topics, will not be discussed in any detail, even though this can be an important means of reducing the time and cost of using simulation. These methods are generally applied to the scheme in which the simulation model utilizes random numbers. The result is that the variance of response variables is reduced substantially which can be used as a basis for reducing the number of iterations required for developing results at the desired level of confidence.

The topics already discussed are often to some degree dependent on the simulation language chosen for the study. If a simulation language is utilized, the time-flow mechanism is already specified within the confines of the language, and often methods of random-number generation and stochastic-variate generation. Some of the general-purpose simulation languages being used today include the following:

- SIMSCRIPT 2.5
- GASP II/IV
- DYNAMO
- GPSS III
- SIMULA
- SOL

Many of the simulation textbooks contain detailed comparisons of the languages that can be helpful in selecting one to use. A brief discussion of simulation languages can be found in Section 9.18.

Conclusions

Simulation is a useful and appropriate management science technique for use in the analysis of complex problems. The technique permits the testing and evaluation of current, proposed, or conceptualized systems without risk to current system performance or need for real-world experimentation. It also permits the study of complex problems where direct analytic solution is not possible. Simulation is also an easy technique in which to develop proficiency, although it often requires a wide range of skills. The remainder of this chapter will be devoted to the task of developing these skills.

II. EXAMPLES OF SIMULATION MODELING

9.4 SALES OF LIFE INSURANCE

In order to illustrate the fundamental concepts in simulation analysis, consider the following illustrative example. Suppose that Joe Cool is working for the Cosmic Life Insurance Company. Cosmic Life sells insurance on a door-to-door basis, and has kept accurate records on their past sales history. Based on previous sales, there is a 50% chance that when Joe Cool makes a house call the resident will not be interested in purchasing a life insurance policy. In other words, approximately 50% of the time the resident will be willing to enter into further discussions regarding policy protection. However, this is not a guarantee of future sales. Even though Joe Cool is allowed the opportunity to discuss Cosmic policies, $\frac{1}{2}$ of the time the visit will result in a "no sale," $\frac{1}{3}$ of the time in the sale of a \$10,000 life insurance policy, and $\frac{1}{6}$ of the time in the sale of a

$20,000 policy. Using a simulated sample of 20 household visits, determine the probability that a sale will be made at each house and the "expected" policy value for each policy holder assuming a sale is made.

In order to simulate the behavior of a salesman, two events must be defined probabilistically. The first event is an *interest* or *no interest* event at the door. The second event is a *dollar value* sale given that entrance to the house is granted. In order to simulate these events, suppose that we choose a sequence of 20 visits. At each visit, the outcome of the first event must be determined in such a way as to obey the rule established for interest/no interest. Since this is an event with two discrete outcomes, each with a probability of occurence equal to 0.50, a coin flip will be used to determine the outcome. If a *head* occurs, the salesman is allowed to further discuss the policies. If a *tail* occurs, he will proceed to the next customer. The following information is only relevant if the flip is a head. If interest is indicated by a prospective customer, subsequent discussions will result in one of three outcomes: (1) no sale, (2) sale of a $10,000 policy, or (3) sale of a $20,000 policy. Since there are three possible outcomes, a fair die will be rolled in order to randomly choose the appropriate result. Define "no sale" to occur if a 1, 2, or 3 appears on the die; "$10,000 sale" if a 4 or 5 appears, and "$20,000 sale" if a 6 appears. Note that the probability of these respective outcomes are 1/2, 1/3, and 1/6, respectively, which obey the sale probabilities previously defined. The results of twenty visits are given in Table 9.1.

From Table 9.1, 13 of the visits resulted in a "no sale." Hence $P(\text{no sale}) = 13/20 = 0.65$. In addition, there were four $10,000 policies sold and

Table 9.1
SIMULATION OF 20 INSURANCE CALLS.

Trial	Flip	Interest	No Interest	Roll	$ Value 0	10	20	Value
1	H	X		4		X		10,000
2	H	X		4		X		10,000
3	T		X					0
4	H	X		1	X			0
5	H	X		2	X			0
6	T		X					0
7	H	X		3	X			0
8	H	X		6			X	20,000
9	T		X					0
10	T		X					0
11	H	X		2	X			0
12	T		X					0
13	H	X		3	X			0
14	T		X					0
15	H	X		6			X	20,000
16	H	X		4		X		10,000
17	H	X		2	X			0
18	H	X		4		X	X	10,000
19	T		X					0
20	H	X		6			X	20,000

Table 9.2
COMPARISON OF SIMULATED AND THEORETICAL RESULTS.

	Probability of No Sale	Probability of Sale	Given a Sale: Probability of $10,000	Probability of $20,000	Expected Value of Policy
Simulated	0.65	0.35	0.5714	0.4286	$14,286
Theoretical	0.75	0.25	0.667	0.333	$13,333

three $20,000 policies sold. Therefore, given that a policy is sold, there is a 57.14% chance that it will be a $10,000 policy and a 42.86% chance that it will be a $20,000 policy. Hence, the expected value of a policy given that a policy is sold is estimated as $E(V) = 0.5714(\$10,000) + 0.4286(\$20,000) = \$14,286$. Because of the simplicity of this illustrative problem, these results can be compared to the theoretical results. This comparison is made in Table 9.2.

Table 9.2 reveals that the simulated results agree favorably with the theoretical calculations. A moments reflection will reveal the source of the disagreement, that is, the period over which the results were calculated. It is intuitively clear that the number of visits simulated (20) could drastically effect the results, since the procedure actually relies on probabilistic sampling of stochastic events. Viewed as a sampling procedure, the more samples that are taken the more accurate are the desired results. In general, the interaction of several events might necessitate a large number of trials. It is also clear that extensive bookkeeping procedures might be needed if complex situations are to be simulated. In addition, a great number of "random deviates" might be required for complex event simulations over long periods of time.

In this example, the probabilistic nature of simulation modeling was illustrated, and due to the specialized structure of the (discrete) probability density functions used to describe the problem a simple coin flip and a single die were used to generate a random sequence of events. The following example will serve to illustrate the same procedure in an example too complex to represent in this fashion.

9.5 PRODUCTION LINE MAINTENANCE [Schmidt (59)]

Five production lines are to be maintained by one repair crew. When one of the lines fails it is repaired unless the repair crew is occupied on another line, in which case it must wait for service. The repair crew services the lines in the order in which they fail. The production system operates three shifts per day, 5 days per week.

The time elapsed between start-up of a line and its failure has been observed to vary from one run to another in an unpredictable or random manner. Similar variation has been observed for the time to repair a line when it fails. The observed frequency distribution of failure and service times for line number 1 are given in Table 9.3 and graphically in Figs. 9.4 and 9.5. This data is based on an observation period of 20 weeks. Data of a similar nature were collected on the remaining four lines during the same period but in

Table 9.3
FREQUENCY DISTRIBUTION OF FAILURE AND SERVICE TIMES

Failure Time		Service Time	
Time Interval (weeks)	Observed Frequency	Time Interval (weeks)	Observed Frequency
0.00–0.02	35	0.000–0.002	35
0.02–0.04	20	0.002–0.004	23
0.04–0.06	16	0.004–0.006	28
0.06–0.08	20	0.006–0.008	13
0.08–0.10	17	0.008–0.010	16
0.10–0.12	16	0.010–0.012	13
0.12–0.14	8	0.012–0.014	6
0.14–0.16	6	0.014–0.016	11
0.16–0.18	5	0.016–0.018	3
0.18–0.20	2	0.018–0.020	3
0.20–0.22	5	0.020–0.022	5
0.22–0.24	8	0.022–0.024	4
0.24–0.26	3	0.024–0.026	1
0.26–0.28	5	0.026–0.028	3
0.28–0.30	1	0.028–0.030	2
0.30–0.32	1	0.030–0.032	2
0.32–0.34	1	0.032–0.034	2
0.34–0.36	1	0.034–0.036	2
0.36–0.38	1	0.036–0.044	0
0.38–0.40	0	0.044–0.046	1
0.40–0.42	2	0.046–0.074	0
0.42–0.44	0	0.074–0.076	1
0.44–0.46	1		
Total	174	Total	174

order to avoid unnecessary complications in the model development they will not be presented.

In this system, once the times of failure and repair are known the status of all of the lines and the repair crew can be determined. Therefore, if the analyst can predict these times in some manner he can write a computer simulation program that will reproduce the characteristics of the system. Failure time and service time are random variables and cannot be predicted with certainty, however, and the simulation analyst therefore attempts to generate these times in a manner which will reproduce the variability observed in the past on the assumption that similar variability can be anticipated in the future. However this does not imply that behavior realized in the past will be reproduced identically in the future. The latter point is important. Twenty-week's data has been collected on the system considered here. If the system were observed for an additional 20 weeks one would not expect an identical repetition of the first 20 weeks. Assuming that the system does not change radically from the first 20 weeks to the second, however, one would expect to find similar distributions of service and failure times for the two periods. That is, one would not expect

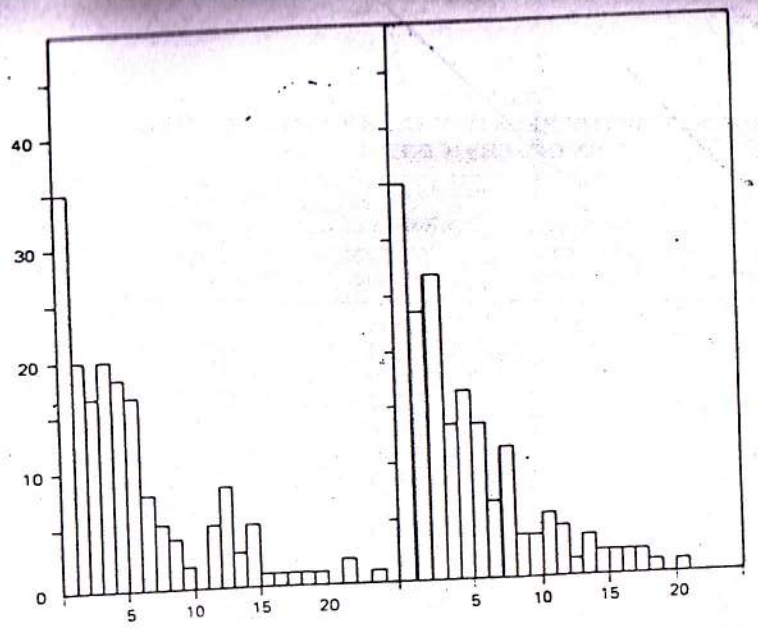


Figure 9.4
Observed failure time (weeks).

Figure 9.5
Observed repair time (weeks).

radical departures from Figs. 9.4 and 9.5 for the second 20 weeks. Therefore, for this problem the analyst would attempt to reproduce the variability of service time and failure time without reproducing identical values of these variables.

The variability of a random variable is generally represented by a *frequency distribution* such as those given in Table 9.3 and Figs. 9.4 and 9.5 or a relative frequency distribution. A relative frequency distribution is the same as a frequency distribution except that the frequency counts in each interval or category are divided by the total number of observations.

An elaboration of the methods for generating random variables will be discussed later in this chapter. The essential characteristic of a realistic random variable generator is that it will reproduce the variability observed in the random variable studied.

The first step in developing a random variable or process generator is calculation of the *cumulative distribution function* of the random variable. Assume that the range of values for the random variable has been broken into n intervals and that f_i is the frequency count for the ith interval. Then the cumulative frequency up to and including the jth interval, F_j, is given by;

$$F_j = \frac{1}{n} \sum_{i=1}^{j} f_i$$

To illustrate, calculation of the cumulative frequency distribution is summarized in Table 9.4 for the service-time data given in Table 9.3.

The next step in generating a random variable is to select or generate a random number. A random number will be defined as a random variable that is

Table 9.4
DEVELOPMENT OF THE RELATIVE CUMULATIVE DISTRIBUTION OF SERVICE TIME.

Time Interval (Weeks)	Observed Frequency Distribution	Relative Frequency Distribution	Relative Cumulative Frequency Distribution
0.000–0.002	35	0.2011	0.2011
0.002–0.004	23	0.1322	0.3333
0.004–0.006	28	0.1609	0.4942
0.006–0.008	13	0.0747	0.5689
0.008–0.010	16	0.0920	0.6609
0.010–0.012	13	0.0747	0.7356
0.012–0.014	6	0.0345	0.7701
0.014–0.016	11	0.0632	0.8333
0.016–0.018	3	0.0173	0.8506
0.018–0.020	3	0.0173	0.8679
0.020–0.022	5	0.0287	0.8966
0.022–0.024	4	0.0230	0.9196
0.024–0.026	1	0.0057	0.9253
0.026–0.028	3	0.0173	0.9426
0.028–0.030	2	0.0115	0.9541
0.030–0.032	2	0.0115	0.9656
0.032–0.034	2	0.0115	0.9771
0.034–0.036	2	0.0115	0.9886
0.036–0.044	0	0.0000	0.9886
0.044–0.046	1	0.0057	0.9943
0.046–0.074	0	0.0000	0.9943
0.074–0.076	1	0.0057	1.0000

uniformly distributed over the interval 0 to 1. That is, each number between 0 and 1 has an equal and independent chance of occurring. Let RN be the random number selected and let x_i be the upper limit of the ith interval. If

$$F_{i-1} < RN \leqq F_i$$

then the value of the random variable will be defined (arbitrarily) as x_i.

In summary, random times to failure and line repair times can be generated through the use of the cumulative distribution functions associated with these random events. Random deviates are obtained through the use of *random numbers* distributed uniformly on the interval 0–1. Generation of random numbers is discussed in great detail in Section 9.3. For the present time, assume that an ample supply of these random numbers is available for use.

Returning to the illustration in Table 9.4, if the random numbers, RN, are uniformly distributed between 0 and 1, then we would expect 20.11% of these numbers to be less than 0.2011 and 20.11% of the service times generated would have a value of 0.002. We would expect 13.22% of the random numbers to fall between 0.2011 and 0.3333 leading to a value of service time of 0.004. Continuing in this manner, it is readily seen that the generative process given here will lead, in the long run, to proportions of times in each interval which

correspond to the proportions observed. Although the use of the upper boun for each interval will create a slight bias with regard to the continuous time scale, this is used only for illustrative purposes and could be corrected through a continuous approximation.

To illustrate the simulation of this system, a 5.5 hour period beginning at midnight will be simulated. The logic of the simulation model is summarized in Fig. 9.6. The first step is to generate the time of the first failure for each line. These times are given as follows:

First failure, line 1 = 2:17
First failure, line 2 = 4:12
First failure, line 3 = 10:24
First failure, line 4 = 3:08
First failure, line 5 = 3:21

The first event which alters the operating status of the system is a failure on the first line at 2:17. As defined above, a repair of line one should commence. Therefore a service time for line 1 is generated and is 27 minutes. This creates the next status-changing event which occurs at 2:44. Since the line will fail again, a new failure time is generated for line 1 and is 30 hours and 15 minutes. Therefore line 1 fails again at 8:59 the next day.

The next event is a failure of line 4 at 3:08. Since no other lines are already down, a service time for line 4 is generated and is 32 minutes. Thus line 4 is up again at 3:40. Before line 4 is repaired, line 5 fails at 3:21 and is the next status-changing event. However, since the repair crew is busy on line 4, line 5 must wait for service. The next event is the repair of line 4 at 3:40, at which time repairs commence on line 5. Placing line 4 in service requires generation of its next failure time and is 31 hours 12 minutes, or at 10:52 the next day. Repair time for line 5 is 22 minutes and repairs are completed at 4:02 and is the next event. At 4:02 a new failure time is generated for line 5 and is 5 minutes or occurs at 4:07. Since all lines are operating at 4:02, the next status-changing event is a line failure. The next line to fail is 5 at 4:07. Repairs for line 5 commence at 4:07 since no other lines are down at this time. Repair time on line 5 is generated and is 18 minutes; line 5 going into operation again at 4:25. The next event which occurs is a failure of line 2 at 4:12. Since the repair crew will be occupied on line 5 until 4:25, line 2 must wait for repairs until this time. The next event in the simulation is a repair of line 5 at 4:25 requiring generation of a new failure time for line 5 and a service time for line 2 since repairs for line 2 start upon completion of repairs for line 5. Generated failure time for line 5 is 58 minutes and service time for line 2 is 19 minutes. Therefore line 2 starts operating at 4:44. At 4.44 a new failure is generated for line 2, is 9 minutes, occurs at 4:53, and is the next event. Service on line 2 starts at 4:53 since line 2 is the only line down at that time. Generated service time for line 2 is 83 minutes. Therefore line 2 is placed in operation at 6:16. At 5:23 line 5 fails but must wait for service since the repair crew is occupied on line 2 at this time.

The above sequence of events is summarized in Table 9.5.

Although an extensive simulation analysis of this system by hand could become tedious if continued for long periods of study, using Fig. 9.6 and the given data a computer program is easily constructed and extended analysis can be conducted. A computer program was utilized for this particular example,

Table 9.5
FIVE AND ONE-HALF-HOUR PERIOD OF SIMULATION FOR FIVE PRODUCTION LINES

Time of change in system Status hr. (min)	Event altering system Status	Status						Cumulative Time		
		Line 1	Line 2	Line 3	Line 4	Line 5	Crew	Lines Down	Repair (min)	Waiting (min)
Midnight	None	Operating	Operating	Operating	Operating	Operating	Idle	0	0	0
2:17	Line 1 fails	Down (In service)					Line 1	1	0	0
2:44	Line 1 repaired	Operating								
3:08	Line 4 fails						Idle	0	27	0
					Down (In service)		Line 4	1	27	0
3:21	Line 5 fails									
						Down (Waiting)	Line 4	2	40	0
3:40	Line 4 repaired				Operating					
						Down (In service)	Line 5	1	59	19
4:02	Line 5 repaired									
4:07	Line 5 fails						Idle	0	81	19
						Down (In service)	Line 5	1	81	19
4:12	Line 2 fails									
			Down (Waiting)				Line 5	2	86	19
4:25	Line 5 repaired									
			Down (In service)			Operating	Line 2	1	99	32
4:44	Line 2 repaired									
4:53	Line 2 fails		Operating				Idle	0	118	32
			Down (In service)				Line 2	1	118	32
5:23	Line 5 fails									
						Down (Waiting)	Line 2	2	148	32

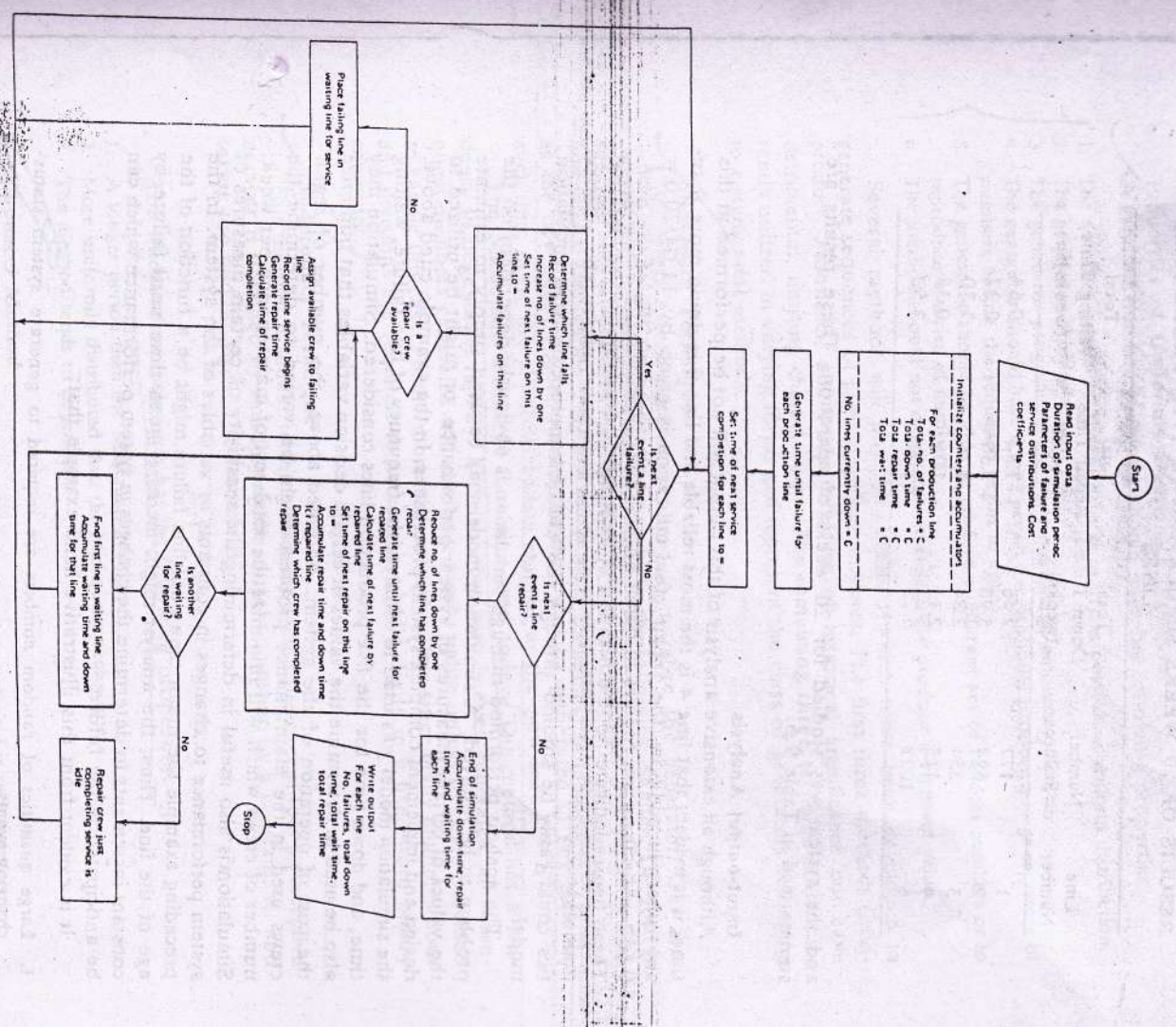


Figure 9.6
Flowchart for the maintenance simulator.

Table 9.6
RESULTS OF 20 WEEKS SIMULATION OF FIVE PRODUCTION LINES

Line Number	Number of Failures	Total Down Time (weeks)	Total Repair Time (weeks)	Total Waiting Time (weeks)
1	177	2.46	1.82	0.64
2	694	5.90	3.58	2.32
3	354	2.84	1.16	1.70
4	115	2.11	1.76	0.35
5	1093	6.25	2.72	3.53

and the system was studied for 20 weeks of operation. These results are summarized in Table 9.6.

Input-output Analysis

Although an extensive analysis of this output will not be performed at this time, it is evident that line 4 is the most reliable and line 5 is down more than any other. In addition, the repair crew utilization is given by $11.04/20.0 = 0.552$ while there are repairs in queue $(8.54)(100)/20.0 = 42.7\%$ of the time. These figures indicate poor utilization of crew repair availability. Preventative maintenance procedures, increased crew sizes, and other system characteristics could easily be studied through the use of a more extensive simulation model.

The analysis performed through simulation is obviously dependent on the problem to be resolved. A simulation model may be used simply to estimate the values of certain measures of system performance, or might be utilized to design and implement complex system procedures. In the example cited above, the simulation model was used to estimate the frequency of line failure, waiting time, and down time for the five production lines considered. Simulation may also be used to determine the values of certain decision variables that minimize the cost of operation of the system. As noted above, the number of repair crews used in the maintenance problem might be varied to determine the number of crews which will minimize the total cost of maintenance per week. Simulation is also useful in determining the sensitivity of certain measures of system performance to changes in constants or variables of the system. In the preceding example the mean time until line failure might be a function of the age of the line. Thus, the analyst might increase mean time until failure by constant increments to determine the changes in system performance which can be anticipated in the future.

It is evident from this illustrative hand example that:

1. Large quantities of random numbers are required to generate system status-changing events.
2. Refinements of the random deviate generation scheme would be desirable in order to more closely approximate the stochastic nature of the system under study. In order to deal with these two problems, the next section will be devoted to the generation of random numbers, while the succeeding section will be devoted to advanced techniques useful in generating a wide variety of random deviates.

III. PSEUDORANDOM NUMBERS

9.6 GENERATION OF RANDOM DEVIATES

In nearly all simulation experiments there exists a need for generating random statistical deviates from a certain distribution. The distribution will be the one that adequately describes and represents the physical process involved at that point in the experiment. During an actual simulation experiment, this process of generating a random deviate from a particular distribution may have to be done many times for many distributions, depending on the complexity of the model being investigated by the experiment.

This section will be concerned with the techniques of generating the required statistical deviates on a digital computer.

The general process for generating a random deviate from a specific distribution will nearly always follow this pattern:

1. Generate a random number from the uniform distribution.
2. Perform a mathematical transformation of the uniform random number or numbers which produces a random deviate from the desired distribution.
3. Use the transformed deviate in the experiment as required.

These three steps are then repeated many times for the chosen distribution (and other distributions as required) as the experiment proceeds in time. Of course, it should be realized that at times the simulation model will require variates directly from the random-number generator, in which case step 2 will not be necessary.

Since the generation of a random number is the beginning for the generation of deviates from the more complicated distributions, the uniform distribution on the interval [0, 1] will be discussed first and then followed by discussions of (1) the techniques for transforming the uniform variate into a variate from a more complicated distribution, and (2) the more complicated distributions themselves.

9.7 THE UNIFORM DISTRIBUTION AND ITS IMPORTANCE TO SIMULATION

The uniform distribution over the interval [0, 1] is given by:

$$f(x_0) = 1 \qquad \text{for } 0 \leqq x_0 \leqq 1 \tag{9.1}$$

$$F(x_0) = x_0 \qquad \text{for } 0 \leqq x_0 \leqq 1 \tag{9.2}$$

where the probability density function $f(x_0)$ = probability $(x = x_0)$ and the cumulative distribution function $F(x_0)$ = probability $(x \leqq x_0)$. The uniform distribution will be used as described above in deriving statistical generators for other probability distributions throughout this chapter. The term *uniform distribution* will refer to the above distribution over the interval [0, 1]. and uniform random numbers or uniform random variates will refer to numbers generated from a uniform distribution over this range. Uniform deviates over a general range are easily generated through a scale transformation.

The probability density function (Fig. 9.7) and the cumulative distribution function (Fig. 9.8) are shown on p. 378.

The imporatance of the uniform distribution stems from its use as the foundation in generating random deviates (variates) from more complicated distributions required in simulation experiments. As a simulation experiment

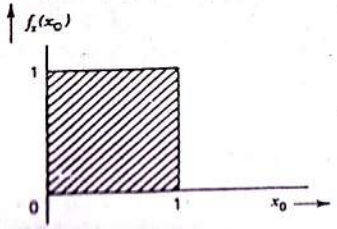


Figure 9.7
Uniform density function.

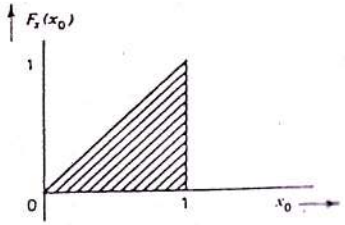


Figure 9.8
Uniform distribution function.

proceeds in time, uniformly distributed random numbers are repeatedly being generated, giving rise to various combinatorial operations on these deviates to produce random variates from any other statistical distribution required by the simulation.

Tocher (52) has suggested three modes of providing uniform random numbers on digital computers. These include (1) external provision, (2) internal generation by a random physical process, and (3) internal generation of sequences of digits by a recursive equation. The first method involves the recording of random-number (47) tables (*Rand* Tables) on magnetic tape for input into a digital computer and then treat these random numbers as data for the problem of interest. A major objection to this method is the slow process of input to the machine. The second method uses an outside physical process whose random results can be read into the computer as a sequence of digits. Among the external processes which have been used are the random decay of radioactive material and thermal noise in an electric valve circuit. The major objections to this method are that the results are not reproducible and that the random physical processes may develop a defect in their randomness. The third alternative, and the one that is most acceptable for digital computer simulations involves the generation of "pseudorandom numbers" by an algorithmic-recursive-type equation. Being algorithmic and recursive indicates that the results of the previous calculations will be used in determining the next calculation. The ith term is used in a formula to calculate the $(i+1)$st term; the $(i+1)$st term is used to calculate the $(i+2)$nd term, and so on. Any series of numbers created by this method can never be truly random but for all practical purposes formulas have been developed which prove highly satisfactory. To determine whether or not a given sequence is satisfactory, various statistical tests have been designed to test the properties of the variates in question. These tests are discussed in some detail after a brief discussion of current and historical methods used to generate these sequences of uniform deviates.

9.8 GENERATION OF RANDOM NUMBERS

Producing a number by some algorithm takes away some of the randomness that it should possess. A truly "random" number or occurrence can be produced only by some physical phenomenon, such as white noise. For this reason, numbers produced by algorithms are correctly referred to as "pseudorandom numbers." Understanding this, pseudorandom numbers shall from this point be referred to as random numbers.

Properties of Uniformly Distributed Numbers

A random number generator should have the following properties:

1. The numbers generated should have as nearly as possible a uniform distribution.
2. The generator should be fast.
3. The generator program should not require large amounts of core.
4. The generator should have a long period (i.e., it should produce a large sequence of numbers before the sequence begins to cycle).
5. The generator should be able to produce a different set of random numbers or to reproduce a series of numbers.
6. The method should not degenerate to repeatedly produce a constant value.

Several methods will now be examined which have been developed to generate sequences of pseudorandom numbers. The first three methods (Midproduct, Fibonacci, and Midsquare) are of historical significance but have detrimental, limiting characteristics. The remaining three techniques are currently utilized in varying forms and represent the class of generators belonging to *congruential methods*.

Midsquare Technique

We select a four-digit integer seed to initialize the generator. Our first random number is obtained from the seed in the following manner: The seed is squared and all digits except the middle four are ignored. The result is then normalized to give the first random number; this number is subsequently used as the new seed. Pseudorandom numbers can be generated in this manner, each time using the previous random number as the new seed.

The Midsquare technique is seldom used today. The method has a tendency to degenerate rapidly. If the number zero is ever generated, all subsequent numbers generated will also have a zero value unless steps are provided to handle this case. Furthermore, the method is slow since many multiplications and divisions are required to access the middle digits in a fixed-word binary computer.

Midproduct Technique

This method is similar to the Midsquare technique except that a successive number is obtained by multiplying the current number by a constant, K, and taking the middle digits. The formula is

$$S_{n+1} = K(S_n)$$

The Midproduct technique has the following properties:

1. A longer period than the Midsquare technique
2. More uniformly distributed than the Midsquare technique
3. The method tends to degenerate

Fibonacci Method

This generator is based on the Fibonacci sequence and is represented by:

$$X_{n+1} = (X_n + X_{n-1}) \bmod m$$

The method usually produces a period of length greater than m; however, the pseudorandom numbers obtained by using the Fibonacci method fail to pass the tests for randomness. Consequently the method does not give satisfactory results.

then the total is divided by 100 (the number of digits considered). This gives the observed probabilities for the numbers generated. The expected values are obtained from the above formulas. A simple chi-square test is now performed with the chart below and the seven categories of possible hands (seven cells).

Actual	0.25000	0.5600	0.080	0.1000	0.0500	0.0000	0.0100
Expected	0.30204	0.5040	0.108	0.0720	0.0045	0.0001	0.0090
Categories	All different	One pair	Two pair	Three of a kind	Four of a kind	Five of a kind	Full house

In summary, it should be noted that even if the numbers tested are truly uniform, the probability that they will fail at least one test may still be high. This is true since each test is actually a hypothesis test run at a level of significance (type I error) of α. By definition α is the probability of rejecting the null hypothesis even if it is true.

EXAMPLE 9.10-3

Perform five tests, each at level of significance $\alpha = 0.05$. The probability of failing at least one of the five tests is:

$$p(\text{fail at least one test}) = 1 - (1 - \alpha)^5$$
$$= 1 - (0.95)^5$$
$$= 0.226$$

even though the numbers are truly acceptable!

There are other tests which can be run to test the "randomness" in a set of pseudorandom numbers. Additional tests are:

1. Serial tests—serial tests are used to check the randomness of successive numbers in a sequence.
2. Product tests—product tests measure the independence (correlation) between sequences of random numbers.
3. Tests for autocorrelation—autocorrelation tests examine the dependency of a particular number on another number in a sequence. These tests attempt to discern whether there are patterns or relationships among sets of random numbers.
4. The Maximum Test—examines sequences of numbers to detect abnormally high or low numbers.

IV. TECHNIQUES FOR GENERATING RANDOM DEVIATES

9.11 THE INVERSE TRANSFORMATION METHOD

The inverse transformation technique deals with the cumulative distribution function, $F(x)$, of the distribution to be simulated. Since $F(x)$ is defined over the interval (0, 1), we can generate a uniform random variate R [also defined over the interval (0, 1)] and set $F(x) = R$. Then x is uniquely determined by the relation $F(x) = R$ and $x = F^{-1}(R)$ is the variate desired from the given distribution. The difficulty with this technique lies in finding the inverse transformation

such that $F^{-1}(R) = x$. If this inverse function can be established, we only need generate various uniform random numbers and perform the inverse function on them to obtain random deviates from the distribution desired.

EXAMPLE 9.11-1

Generate a random variate from:

$$f(x) = \begin{cases} 3x^2 & 0 \leqq x \leqq 1 \\ 0 & \text{Elsewhere} \end{cases}$$

then:

$$F(x) = \int_{t=0}^{x} 3t^2\, dt$$

Hence,

$$F(x) = x^3$$

Since $F(x)$ is the cumulative distribution function of $f(x)$, we can replace $F(x)$ by a random number R. The inverse transformation is therefore:

$$x = F^{-1}(R) = (R)^{1/3}$$

Hence, in order to generate a random deviate from the probability density $f(x) = 3x^2$, a random deviate from the uniform distribution is generated (call it R) and the desired deviate $x = (R)^{1/3}$.

The problem with the inverse transformation technique is that for many distributions, the inverse function, $F^{-1}(R)$, does not exist or it is so complicated as to be impractical. When this is the case, either approximations or other techniques must be used.

The Exponential Distribution

In simulation experiments, the exponential distribution is used to describe the time interval between occurrences of like events, which are often arrivals in queueing problems. If the probability that an event will occur in a small-time interval is very small, and if the occurrence of this event is independent of the occurrence of other events, then the time interval between occurrence of these events is exponentially distributed and the process is a Poisson process. In addition to queueing processes, the exponential distribution is also used to describe component failure rates in reliability analysis.

The probability density function and cumulative distribution function are given as

$$f(x) = \lambda e^{-\lambda x} \qquad \text{for } x \geqq 0$$
$$F(x) = 1 - e^{-\lambda x} \qquad \text{for } x \geqq 0$$

To generate a random deviate from the exponential distribution, the inverse transform technique is easily applied. We have $F(x) = 1 - e^{-\lambda x}$; so we generate a uniform random deviate R and let

$$R = F(x) = 1 - e^{-\lambda x}$$

To find the inverse transform:

$$R = 1 - e^{-\lambda x}$$

or:

$$1 - R = e^{-\lambda x}$$

but $1-R$ is also from the uniform distribution, so let:

$$R = e^{-\lambda x}$$

or:

$$x = -\frac{\ln R}{\lambda}$$

Hence

$$F^{-1}(R) = -\frac{\ln R}{\lambda} = x$$

The procedure is to generate uniform random deviates and apply $F^{-1}(R)$ to generate exponential variates.

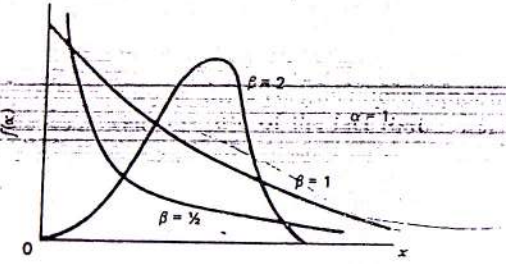
Weibull Distribution

The Weibull distribution is a family of density functions widely used in describing failure rate characteristics in reliability analysis.

The density function is

$$f(x) = \alpha\beta x^{\beta-1} e^{-\alpha x^{\beta}} \qquad x \geq 0$$

for $x > 0$, $\alpha > 0$, $\beta > 0$, the Weibull density function generates a family of probability density curves as α and β change their values.



To generate a Weibull random deviate, the inverse transform technique can be used.

EXAMPLE 9.11-2

Define the cumulative distribution function.

$$F(x) = \alpha\beta \int_{t=0}^{x} t^{\beta-1} e^{-\alpha t^{\beta}}\, dt$$

let $y = \alpha t^{\beta}$

$$dy = \alpha\beta t^{\beta-1}\, dt$$

then:

$$F(x) = \alpha\beta \int_{0}^{\alpha x^{\beta}} t^{\beta-1} e^{-y} \frac{dy}{\alpha\beta t^{\beta-1}}$$

Hence,

$$F(x) = 1 - e^{-\alpha x^{\beta}}$$

We must now find the inverse transform $F^{-1}(R)$.

$F(x) = R = 1 - e^{-\alpha x^{\beta}}$ where R is from the uniform distribution

$1 - R = e^{-\alpha x^{\beta}}$ but $1-R$ is also from the uniform distribution, so

$R = e^{-\alpha x^{\beta}}$

and

$$x = \left[-\frac{1}{\alpha} \ln R\right]^{1/\beta}$$

Hence in order to generate a series of deviates from the Weibull distribution, we only need to generate random deviates from the uniform distribution and apply the inverse transform

$$F^{-1}(R) = \left[-\frac{1}{\alpha} \ln R\right]^{1/\beta} = x$$

The Geometric Distribution

A random variable x, defined as the number of failures in a sequence of Bernoulli trials before the first success occurs, is known as a geometric random variable. This distribution is related to the binomial distribution and has been used in the area of quality control and for lag distributions in econometric models.

The probability density function for the geometric distribution is

$$f(x) = pq^{x} \qquad x = 0, 1, 2, \ldots$$

By definition, p is the probability of success for each Bernoulli trial and $q = 1 - p$. The cumulative distribution function is given by:

$$F(x) = \sum_{k=0}^{x} pq^{k}$$

To generate a random deviate from the geometric distribution, we make use of the fact that

$$1 - F(x) = q^{x+1}$$

and that $[1-F(x)]/q$ has unit range. It will be left as an exercise for the student to show that these relationships are correct.

Once the above is accepted, let

$$R = q^{x} \qquad \text{(inverse technique)}$$

$$\ln R = x \ln q$$

Since x must be an integer, choose x such that x is the largest integer that satisfies

$$x = \ln R / \ln q$$

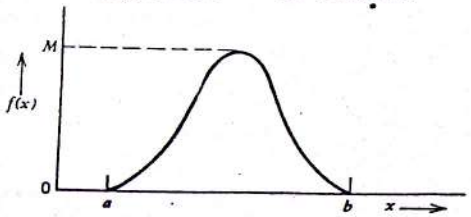
9.12 THE REJECTION TECHNIQUE

The rejection technique consists of drawing a random value from an appropriate distribution and subjecting it to a test to determine whether or not it will be accepted for use. To illustrate, let $f(x)$ be a frequency density function such that

$$f(x) = 0 \qquad \text{for } a > x > b$$

and

$$0 \leqq f(x) \leqq M \qquad \text{for } a \leqq x \leqq b$$



CSE '07

Networks of Queues

9.1 INTRODUCTION

We have studied Markov processes of the birth death type in Chapter 8. Such processes are characterized by a simple product-form solution [equation (8.20)] and *a large number of applications.* When we remove the restriction of nearest neighbor transitions only, we may not have the convenient product-form solution. It is natural to ask whether there is a class of Markov processes that subsumes birth-death processes and that possesses a product-form solution. One important generalization of the birth-death process that we consider is a network of queues. Such networks can model problems of contention that arise when a set of resources are shared. Each resource is represented by a node or a service center. Thus, in a model for computer-system performance analysis, we may have a service center for the CPU(s), a service center for each I/O channel, and possibly others. A service center may have one or more servers associated with it. If a job requesting service finds all the servers at the service center busy, it will join the queue associated with the center, and at a later point in time, when one of the servers becomes idle, a job from the queue will be selected for service according to some scheduling discipline. After completion of service at one service center, the job may move to another service center for further service, reenter the same service center, or leave the system.

We shall consider two types of networks: open and closed. An **open queuing network** is characterized by one or more sources of job arrivals and correspondingly one or more sinks that absorb jobs departing from the network. In a **closed queuing network**, on the other hand, jobs neither enter nor depart from the network.

The behavior of jobs within the network is characterized by the probabilities of transitions between service centers and the distribution of job service times at each center. For each center the number of servers, the scheduling discipline, and the size of the queue must be specified. Unless stated otherwise, we assume that the scheduling is FCFS and that each server has a queue of unlimited capacity. For an open network, a characterization of job-arrival processes is needed, and for a closed network, the number of jobs in the network must be specified.

Consider the two-stage tandem network shown in Figure 9.1. The system consists of two nodes with respective service rates μ_0 and μ_1. The external arrival rate is λ. The output of the node labeled 0 is the input to the node labeled 1. The service-time distribution at both nodes is exponential and the arrival process to the node labeled 0 is Poisson.

This system can be modeled as a stochastic process whose states are specified by pairs (k_0, k_1), $k_0 \geq 0$, $k_1 \geq 0$, where k_i $(i = 0, 1)$ is the number of jobs at server i in the steady state. The changes of state occur upon a completion of service at one of the two servers or upon an external arrival. Since all interevent times are exponentially distributed (by our assumptions), it follows that the stochastic process is a Markov chain with the state diagram shown in Figure 9.2.

For $k_0, k_1 > 0$, the transitions into and out of that state are shown in Figure 9.3. Let $p(k_0, k_1)$ be the joint probability of k_0 jobs at node 0 and k_1 jobs at node 1 in the steady state. Equating the rates of flow into and out of the state, we obtain the following balance equations:

$$(\mu_0 + \mu_1 + \lambda)p(k_0, k_1) = \mu_0 p(k_0 + 1, k_1 - 1) + \mu_1 p(k_0, k_1 + 1) + \lambda p(k_0 - 1, k_1), \qquad k_0 > 0, \; k_1 > 0.$$

For the boundary states, we have:

$$(\mu_0 + \lambda)p(k_0, 0) = \mu_1 p(k_0, 1) + \lambda p(k_0 - 1, 0), \qquad k_0 > 0,$$

$$(\mu_1 + \lambda)p(0, k_1) = \mu_0 p(1, k_1 - 1) + \mu_1 p(0, k_1 + 1), \qquad k_1 > 0,$$

$$\lambda p(0, 0) = \mu_1 p(0, 1).$$

The normalization is provided by:

$$\sum_{k_0 \geq 0} \sum_{k_1 \geq 0} p(k_0, k_1) = 1.$$

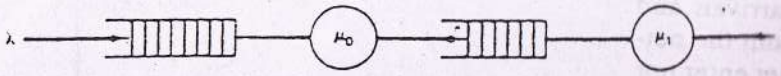


Figure 9.1 A two-stage tandem network

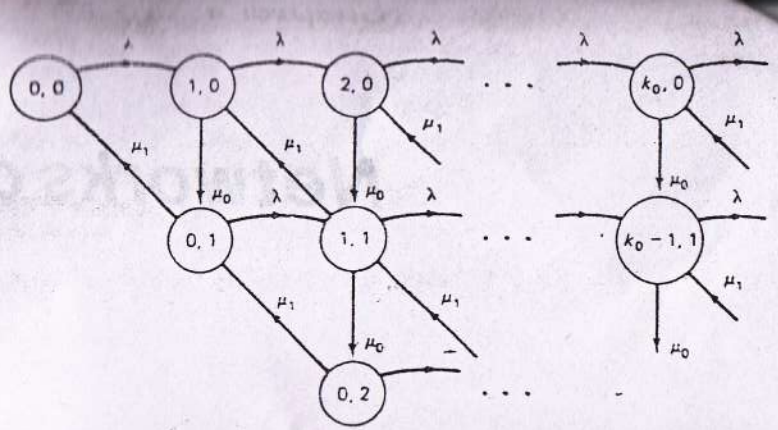


Figure 9.2 The state diagram for the two-stage tandem network

It is easily shown by direct substitution that the following equation is the solution to the above balance equations:

$$p(k_0, k_1) = (1 - \rho_0)\rho_0^{k_0}(1 - \rho_1)\rho_1^{k_1}, \tag{9.1}$$

where $\rho_0 = \lambda/\mu_0$ and $\rho_1 = \lambda/\mu_1$. The condition for stability of the system is that both ρ_0 and ρ_1 are less than unity.

Equation (9.1) is a product-form solution similar to that of an $M/M/1$ queue. Observe that the node 0 in Figure 9.1 has a Poisson arrival source of rate λ and exponentially distributed service time. Therefore, the node la-

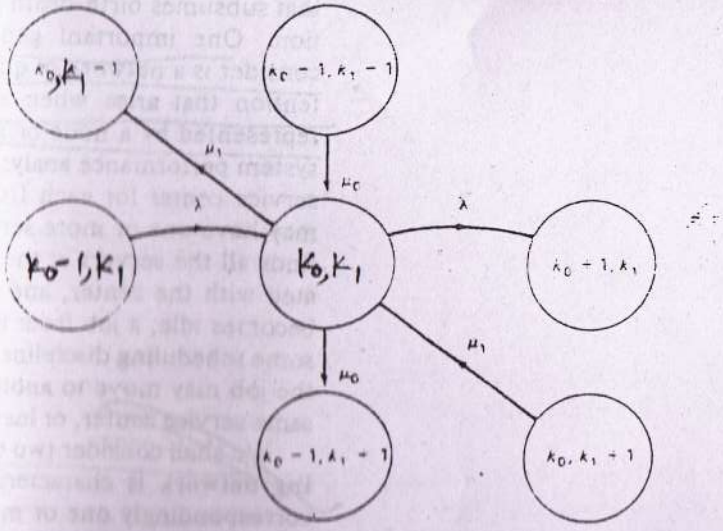


Figure 9.3

beled 0 is an $M/M/1$ queue. It follows that the pmf of the number of jobs N_0 at node 0 in the steady state is given by [formula (8.20)]:

$$P(N_0 = k_0) = p_0(k_0) = (1 - \rho_0)\rho_0^{k_0}.$$

Burke [BURK 1956] has shown that the output of an $M/M/1$ queue is also Poisson with rate λ (you are asked to verify this result in problem 2 at the end of this section). Thus, the second queue in Figure 9.1 is also an $M/M/1$ queue with server utilization $\rho_1 = \lambda/\mu_1$ (assumed to be < 1). Hence, the steady-state pmf of the number of jobs N_1 at node 1 is given by:

$$P(N_1 = k_1) = p_1(k_1) = (1 - \rho_1)\rho_1^{k_1}.$$

The joint probability of k_0 jobs at node 0 and k_1 jobs at node 1 is given by equation (9.1):

$$p(k_0, k_1) = (1 - \rho_0)\rho_0^{k_0}(1 - \rho_1)\rho_1^{k_1} = p_0(k_0)p_1(k_1).$$

Thus the joint probability $p(k_0, k_1)$ is the product of the marginal probabilities $p_0(k_0)$ $p_1(k_1)$; hence random variables N_0 and N_1 are independent in the steady state. Therefore, the two queues are independent $M/M/1$ queues. As the arrival rate λ increases, the node with the larger value of ρ will introduce instability. Hence the node with the largest value of ρ is called the "bottleneck" of the system.

The product form solution (9.1) can be generalized to an m-stage tandem queue.

Example 9.1

A repair facility shared by a large number of machines has two sequential stations with respective rates one per hour and two per hour. The cumulative failure rate of all the machines is 0.5 per hour. Assuming that the system behavior may be approximated by the two-stage tandem queue of Figure 9.1, determine the average repair time.

Given $\lambda = 0.5$, $\mu_0 = 1$, $\mu_1 = 2$, we have $\rho_0 = 0.5$ and $\rho_1 = 0.25$. The average length of the queue at station i ($i = 0, 1$) is then given by [using formula (8.21)]:

$$E[N_i] = \frac{\rho_i}{1 - \rho_i};$$

hence:

$$E[N_0] = 1 \quad \text{and} \quad E[N_1] = \tfrac{1}{3}.$$

Using Little's formula, the repair delay at the two stations is respectively given by:

$$E[R_0] = \frac{E[N_0]}{\lambda} = 2 \quad \text{and} \quad E[R_1] = \frac{E[N_1]}{\lambda} = \tfrac{2}{3} \text{ hours.}$$

Hence the average repair time is given by:

$$E[R] = E[R_0] + E[R_1] = \tfrac{8}{3} \text{ hours.}$$

This can be decomposed into waiting time at station 0 (= 1 hour), the service time at station 0 ($= 1/\mu_0 = 1$ hour), the waiting time at station 1 ($= \frac{1}{6}$ hour), and the service time at station 1 ($1/\mu_1 = \frac{1}{2}$ hour). The probability that both service stations are idle is given by:

$$p(0, 0) = (1 - \rho_0)(1 - \rho_1) = \tfrac{3}{8}.$$

Station 0 is the bottleneck of the repair facility. #

Problems

1. In Chapter 8 we derived the distribution function of the response time of an isolated $M/M/1$ FCFS queue [see equation (8.25)]. Using this result, derive the distribution function of the response time for the tandem network of Figure 9.1. From this obtain the variance of the response time.

*2. Consider an $M/G/1$ queue with FCFS scheduling. Let the random variables A, B, and D, respectively denote the interarrival time, the service time, and the interdeparture time. By conditioning on the number of jobs in the system and then using the theorem of total Laplace transforms show that in the steady state:

$$L_D(s) = \rho L_B(s) + (1 - \rho)L_A(s)L_B(s).$$

Point out why the assumption of Poisson arrival stream is needed to derive this result. Then, specializing to the case of $M/M/1$ queue, show that:

$$L_D(s) = L_A(s).$$

This verifies Burke's result that the output process of an $M/M/1$ FCFS queue is Poisson. Note that the independence of successive interdeparture times needs to be shown in order to complete the proof.

3. Using the result of problem 2, show that in the $M/G/1$ case, the coefficient of variation of the interdeparture time is given by:

$$C_D^2 = 1 + \rho^2(C_B^2 - 1).$$

*4. Show that the interdeparture-time distribution of an $M/M/m$ FCFS queue is exponential. To simplify the problem, first consider an $M/M/2$ queue. Let D_i denote the interdeparture time conditioned on the number of jobs in the system $N = i$. Then show that $D_i \sim \text{EXP}(2\mu)$ for $i \geq 2$. Next show that:

$$L_{D_1}(s) = \frac{\lambda \cdot 2 \cdot \mu}{(s + \lambda + \mu)(s + 2\mu)} + \frac{\mu}{s + \lambda + \mu}$$

and

$$L_{D_0}(s) = \frac{\lambda}{s + \lambda} L_{D_1}(s)$$

(a tree diagram may be helpful here). Then obtain the required result using the theorem of total Laplace transforms. The generalization to the $M/M/m$ case proceeds in a similar fashion.

The argument given in the previous section for the product-form solution of tandem queues can be generalized to any feed-forward network of exponential queues (in which a job may not return to previously visited nodes) that is fed from independent Poisson sources. Jackson [JACK 1957] showed that the product-form solution also applies to open networks of Markovian queues with feedback. Besides requiring that the distributions of job interarrival times and service times at all nodes be exponential, assume that the scheduling discipline is FCFS.

First we consider several examples illustrating Jackson's technique.

Example 9.2

Consider the simple model of a computer system shown in Figure 9.4a. Jackson's result is that two queues will behave like independent $M/M/1$ queues, and hence:

$$p(k_0, k_1) = (1 - \rho_0)\rho_0^{k_0}(1 - \rho_1)\rho_1^{k_1}. \tag{9.2}$$

where $\lambda_0/\mu_0 = \rho_0$ and $\lambda_1/\mu_1 = \rho_1$. To apply this result, we have to compute the average arrival rates λ_0 and λ_1 into the two nodes. Note that in the steady state the departure rates from the two nodes will also be λ_0 and λ_1, respectively. Arrivals to the CPU node occur either from the outside world at the rate λ or from the I/O node at the rate λ_1. The total arrival rate to the CPU node is, therefore,

$$\lambda_0 = \lambda + \lambda_1.$$

Given that a job just completed a CPU burst, it will next request I/O service with probability p_1. Therefore, the average arrival rate to device 1 is given by:

$$\lambda_1 = \lambda_0 p_1.$$

Thus:

$$\lambda_0 = \frac{\lambda}{1 - p_1} = \frac{\lambda}{p_0} \quad \text{and} \quad \lambda_1 = \frac{p_1\lambda}{p_0}$$

This implies that:

$$\rho_0 = \frac{\lambda}{p_0\mu_0} \quad \text{and} \quad \rho_1 = \frac{p_1\lambda}{p_0\mu_1}$$

If we let B_0 denote the total CPU service requirement of a program, then $E[B_0] = 1/(p_0\mu_0)$. Similarly, $E[B_1] = p_1/(p_0\mu_1)$ denotes the expected value of the total service time required on the I/O device for a typical program. If $\rho_0 > \rho_1$ (that is, $E[B_0] > E[B_1]$), then the CPU is the bottleneck, in which case the system is said to be CPU-bound. Similarly, if $\rho_0 < \rho_1$, then the system is I/O-bound.

The average response time may be computed by summing the average number of jobs at the two nodes and then using Little's formula:

$$E[R] = \left(\frac{\rho_0}{1 - \rho_0} + \frac{\rho_1}{1 - \rho_1}\right)\frac{1}{\lambda}$$

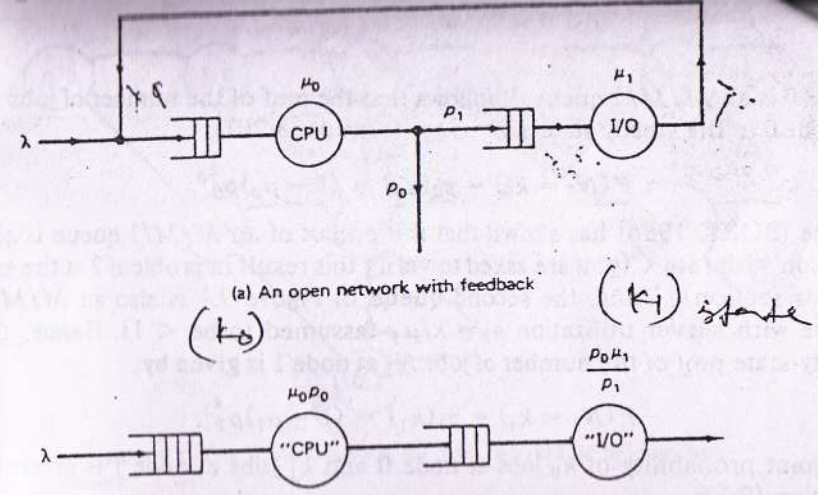


Figure 9.4

or

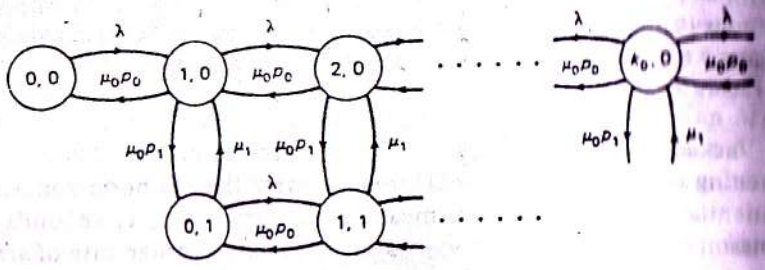
$$E[R] = \frac{1}{p_0\mu_0 - \lambda} + \frac{1}{\frac{p_0\mu_1}{p_1} - \lambda}$$

$$= \frac{E[B_0]}{1 - \lambda E[B_0]} + \frac{E[B_1]}{1 - \lambda E[B_1]} \tag{9.3}$$

It is easily seen that this formula also gives the average turnaround time of the "unfolded" tandem network shown in Figure 9.4b. The service rate of the "equivalent" CPU in this system is $\mu_0 p_0$. Thus, a job requests an uninterrupted average CPU time equal to $E[B_0] = 1/(\mu_0 p_0)$ in the equivalent system, while in the original system a job requires $1/p_0$ CPU bursts of average time $1/\mu_0$ each. Therefore, to determine $E[R]$ and $E[N]$, it is sufficient to know only the aggregate resource requirements of a job; in particular, details of the pattern of resource usage are not important for computing these average values. We caution the reader that the equivalence between the networks of Figures 9.4a and 9.4b does not hold with respect to the distribution function $F_R(x)$ of the response time. Computation of response-time distribution is difficult even for Jacksonian networks without feedback [MELA 1980, SIMO 1979].

It is instructive to solve the network in Figure 9.4a by directly analyzing the stochastic process whose states are given by pairs (k_0, k_1), $k_0 \geq 0$, $k_1 \geq 0$. By the assumption of exponentially distributed interevent times, the process is a Markov process with the state diagram shown in Figure 9.5. For a state (k_0, k_1) with $k_0 > 0$, $k_1 > 0$, the steady-state balance equation is obtained by equating the rate of flow into the state with the rate of flow out of the state:

$$(\lambda + \mu_0 + \mu_1)\, p(k_0, k_1) = \lambda p(k_0 - 1, k_1) + \mu_0 p_1 p(k_0 + 1, k_1 - 1)$$
$$+ \mu_0 p_0 p(k_0 + 1, k_1) + \mu_1 p(k_0 - 1, k_1 + 1)$$



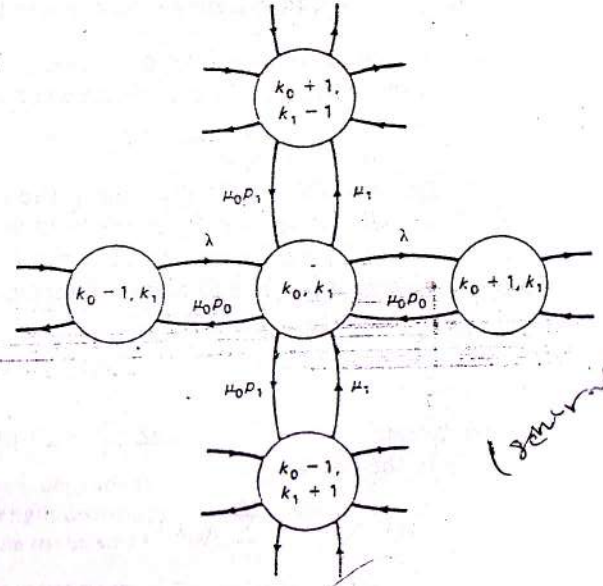


Figure 9.5

Similarly:

$$(\lambda + \mu_0)p(k_0, 0) = \lambda p(k_0 - 1, 0) + \mu_0 p_0 p(k_0 + 1, 0) + \mu_1 p(k_0 - 1, 1), \quad k_0 > 0,$$

$$(\lambda + \mu_1)p(0, k_1) = \mu_0 p_0 p(1, k_1) + \mu_0 p_1 p(1, k_1 - 1), \quad k_1 > 0,$$

$$\lambda p(0, 0) = \mu_0 p_0 p(1, 0).$$

Also:

$$\sum_{k_0, k_1} p(k_0, k_1) = 1.$$

It may be verified by direct substitution that the solution (9.2) indeed satisfies these equations.

Example 9.3

Consider the (open) central server queuing model of a computer system shown in Figure 9.6. Let us follow the path of a tagged program that just arrived. Temporarily

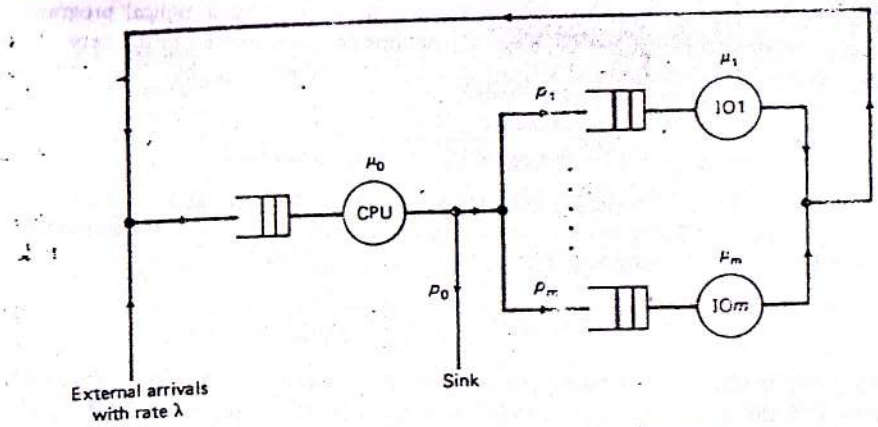


Figure 9.6 The open central server network

ignoring queuing delays, the program will be occupying one of $m + 1$ nodes at a time. With appropriate assumptions, we can model the behavior of a tagged program as a discrete-parameter Markov chain as in Example 7.17; the corresponding state diagram is given in Figure 7.18. [Note that we have added an absorbing state, labeled $m + 1$ (or STOP).] The transition probability matrix of this Markov chain is given by:

$$P = \left[\begin{array}{cccc|c} 0 & p_1 & \cdots & p_m & p_0 \\ 1 & 0 & & 0 & 0 \\ 1 & 0 & & 0 & 0 \\ 1 & \cdot & & \cdot & \cdot \\ 1 & \cdot & & \cdot & \cdot \\ 1 & \cdot & & \cdot & \cdot \\ 1 & 0 & & 0 & 0 \\ \hline 0 & 0 & \cdots & 0 & 1 \end{array}\right]$$

As discussed in Chapter 7 (Section 7.8), the boxed portion of this matrix, denoted here by X, is of interest:

$$X = \begin{bmatrix} 0 & p_1 & p_2 & \cdots & p_m \\ 1 & 0 & 0 & & 0 \\ 1 & \cdot & \cdot & & 0 \\ 1 & \cdot & \cdot & \cdot & 0 \\ 1 & \cdot & \cdot & & 0 \\ 1 & & & & 0 \\ 1 & 0 & 0 & & 0 \end{bmatrix}$$

X is known as the transition probability matrix of the queuing network of Figure 9.6. Analyzing the behavior of the tagged program, we are able to obtain the average number of visits (or visit counts) V_j as in Example 7.17 (we assume that $p_0 \neq 0$):

$$V_j = \begin{cases} 1/p_0, & j = 0, \\ p_j/p_0, & j = 1, 2, \ldots, m. \end{cases}$$

That is, the average number of visits made to node j by a typical program is p_j/p_0 ($j \neq 0$) and $1/p_0$ ($j = 0$). Since λ programs per unit time enter the network on the average, the overall rate of arrivals, λ_j, to node j is then given by:

$$\lambda_j = \begin{cases} \lambda/p_0, & j = 0, \\ \lambda p_j/p_0, & j = 1, 2, \ldots, m. \end{cases}$$

The utilization, ρ_j, of node j is given by $\rho_j = \lambda_j/\mu_j = \lambda V_j/\mu_j$ and we assume that $\rho_j < 1$ for all j. Jackson has shown that the joint probability of k_j customers at node j ($j = 0, 1, \ldots, m$) is given by:

$$p(k_0, k_1, k_2, \ldots, k_m) = \prod_{j=0}^{m} p_j(k_j). \tag{9.4}$$

This formula implies that the queue lengths are mutually independent in the steady state, and the steady-state probability of k_j customers at node j is given by the $M/M/1$ formula:

$$p_j(k_j) = (1 - \rho_j)\rho_j^{k_j}.$$

The validity of this product-form solution can be established using the direct approach as in Examples 9.1 and 9.2. We leave this as an exercise.

The form of the joint probability (9.4) can mislead a reader to believe that the traffic along the arcs consists of Poisson processes. The reader is urged to solve problem 3 at the end of this section to realize that the input process to a service center in a network with feedback is not Poisson in general [BEUT 1978, BURK 1976]. It is for this reason that Jackson's result is remarkable.

The average queue length, $E[N_j]$, and the response time, $E[R_j]$, of node j are given by:

$$E[N_j] = \frac{\rho_j}{1 - \rho_j} \quad \text{and} \quad E[R_j] = \frac{1}{\lambda} \frac{\rho_j}{1 - \rho_j}.$$

From these, the average number of jobs in the system and the average response time are computed to be:

$$E[N] = \sum_{j=0}^{m} \frac{\rho_j}{1 - \rho_j}$$

and

$$\begin{aligned} E[R] &= \frac{1}{\lambda} \sum_{j=0}^{m} \frac{\rho_j}{1 - \rho_j} \\ &= \frac{1/(p_0\mu_0)}{1 - \lambda/(p_0\mu_0)} + \sum_{j=1}^{m} \frac{p_j/(p_0\mu_j)}{1 - \lambda p_j/(\mu_j p_0)} \\ &= \frac{1}{\mu_0 p_0 - \lambda} + \sum_{j=1}^{m} \frac{1}{\left(\frac{p_0\mu_j}{p_j} - \lambda\right)} \\ &= \frac{1}{\frac{\mu_0}{V_0} - \lambda} + \sum_{j=1}^{m} \frac{1}{\frac{\mu_j}{V_j} - \lambda} = \sum_{j=0}^{m} \frac{E[B_j]}{1 - \lambda_j E[B_j]}. \end{aligned} \tag{9.5}$$

where the total service requirement on device j, on the average, is given by $E[B_j] = V_j/\mu_j$. This last formula for the response time is a generalized version of formula (9.3). It also affords an "unfolded" interpretation of the queuing network of Figure 9.6, in the same way as Figure 9.4b is the "unfolded" version of the network in Figure 9.4a. #

Jackson's result applies in even greater generality. Consider an open queuing network with $(m + 1)$ nodes, where the ith node consists of c_i exponential servers each with mean service time of $1/\mu_i$ seconds. External Poisson sources contribute γ_i jobs/second to the average rate of arrival to the ith node so that the total arrival rate $\lambda = \sum_{i=0}^{m} \gamma_i$. If we let $q_i = \gamma_i/\lambda$, $i = 0, 1, \ldots, m$ so that $\sum_{i=0}^{m} q_i = 1$, then a job will first enter the network at node i with probability q_i. Upon service completion at node i, a job next requires service at node j with probability x_{ij} or completes execution with probability $1 - \sum_{j=0}^{m} x_{ij}$.

First, we analyze the behavior of a tagged job through the network. This behavior can be modeled as a discrete-parameter Markov chain as in Example 7.17. Equation (7.52) is applicable here in computing V_i, the average number of visits made by the tagged program to node i. Therefore:

$$V_i = q_i + \sum_{k=0}^{m} x_{ki} V_k, \qquad i = 0, 1, \ldots, m.$$

Now the average job arrival rate to node i is obtained by multiplying V_i by λ, the average job arrival rate to the network. Thus:

$$\lambda_i = \lambda V_i = \lambda q_i + \sum_{k=0}^{m} x_{ki} \lambda V_k.$$

Noting that $\lambda q_i = \gamma_i$ and $\lambda V_i = \lambda_i$, the above expression simplifies to:

$$\lambda_i = \gamma_i + \sum_{k=0}^{m} \lambda_k x_{ki}, \qquad i = 0, 1, \ldots, m. \tag{9.6}$$

This system of equations has a unique solution if we assume that there is at least one node j such that $\gamma_j > 0$ and that the matrix power series:

$$\sum_{k=0}^{n} X^k$$

converges as n approaches infinity (see Section 7.8). This implies that after some number of visits to various service centers there is a positive probability that a job will depart from the system. Jackson's theorem states that each node behaves like an independent $M/M/c_i$ queue and, therefore, the

steady-state probability of k_i customers at node i, $i = 0, 1, \ldots, m$ is given by the product form:

$$p(k_0, k_1, \ldots, k_m) = p_0(k_0) \cdots p_m(k_m), \tag{9.6a}$$

where $p_i(k_i)$ is the steady-state probability of finding k_i jobs in an $M/M/c_i$ queue with input λ_i and with average service time $1/\mu_i$ for each of the c_i servers. Thus, using equations (8.30) and (8.31), we have:

$$p_i(k_i) = \begin{cases} \dfrac{(\lambda_i/\mu_i)^{k_i}}{k_i!} p_i(0), & 1 \leqslant k_i < c_i \\ \dfrac{(\lambda_i/\mu_i)^{k_i} p_i(0)}{c_i! c_i^{k_i - c_i}}, & k_i \geqslant c_i, \end{cases}$$

where:

$$p_i(0) = \frac{1}{\displaystyle\sum_{k_i=0}^{c_i-1} \frac{(\lambda_i/\mu_i)^{k_i}}{k_i!} + \frac{(\lambda_i/\mu_i)^{c_i}}{c_i!} \frac{1}{1 - \dfrac{\lambda_i}{c_i \mu_i}}}.$$

Problems

1. Consider the open central server queuing model with two I/O channels with a common service rate of 1.2 second^{-1}. The CPU service rate is 2 second^{-1}, the arrival rate is ½ job/second. The branching probabilities are given by $p_0 = 0.1$, $p_1 = 0.3$, and $p_2 = 0.6$. Determine steady-state probabilities, assuming all service times are independent exponentially distributed random variables. Determine the queue-length distributions at each node as well as the average response time from the source to the sink.

2. Consider a variation of the queuing model of Figure 9.4a, where the CPU node consists of two parallel processors with a service rate of μ_0 each. Draw a state diagram for this system and proceed to solve the balance equations. Obtain an expression for the average response time $E[R]$ as a function of μ_0, μ_1, p_0, and λ. Now compare your answer with that obtained using Jackson's result.

*3. [BURK 1976] Consider the $M/M/1$ FCFS queue with feedback as shown in Figure 9.P.1. By Jackson's theorem, it is easy to derive the steady-state pmf of the number of jobs N in the system:

$$P(N = i) = \left(1 - \frac{\lambda}{\mu p}\right)\left(\frac{\lambda}{\mu p}\right)^i, \quad i = 0, 1, \ldots.$$

We wish to show that the actual input process (which is a merger of the external arrival process and the feedback process) is not Poisson, even though the exogenous arrival process and the departure process are both Poisson with rate λ.

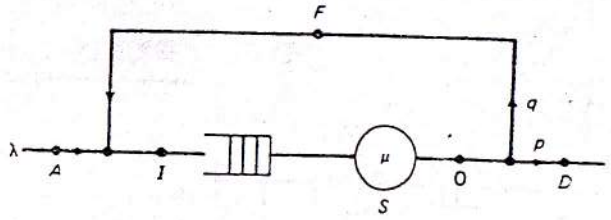


Figure 9.P.1 $M/M/1$ queue with Bernoulli feedback

Proceed by first observing that the complementary distribution function of the terinput times is given by:

$$R_I(t) = e^{-\lambda t} R_Y(t),$$

where Y is the time to the next feedback as measured from the time of the last input to the (queue, server) pair. Next derive the pdf of Y as:

$$f_Y(t) = \mu q e^{-(\mu - \lambda)t},$$

hence show that:

$$R_Y(t) = \frac{\mu q}{\mu - \lambda} e^{-(\mu - \lambda)t} + \frac{\mu p - \lambda}{\mu - \lambda}.$$

From this, conclude that I is hyperexponentially distributed.

In order to derive the Laplace transform of the interdeparture time D proceed by computing the conditional Laplace transform $L_{D|N_d = i}(s)$, where N_d is the number of jobs left in the system by a departing job. Using the result of problem 5, Section 7.6, show that:

$$P(N_d = i) = P(N = i)$$

and hence the unconditional Laplace transform of D is obtained as:

$$L_D(s) = \frac{\lambda}{s + \lambda}.$$

This verifies that the departure process is Poisson.

9.3 CLOSED QUEUING NETWORKS

One of the implicit assumptions behind the model of Example 9.3 is that immediately upon its arrival a job is scheduled into main memory and is able to compete for active resources such as the CPU and the I/O channels. In practice, the number of main-memory partitions will be limited, which implies the existence of an additional queue, called the **job-scheduler queue** (see Figure 9.7). However, such a network is said to involve multiple resource holding. This is because a job can simultaneously hold main memory and an active device. Such a network cannot be solved by product-form methods

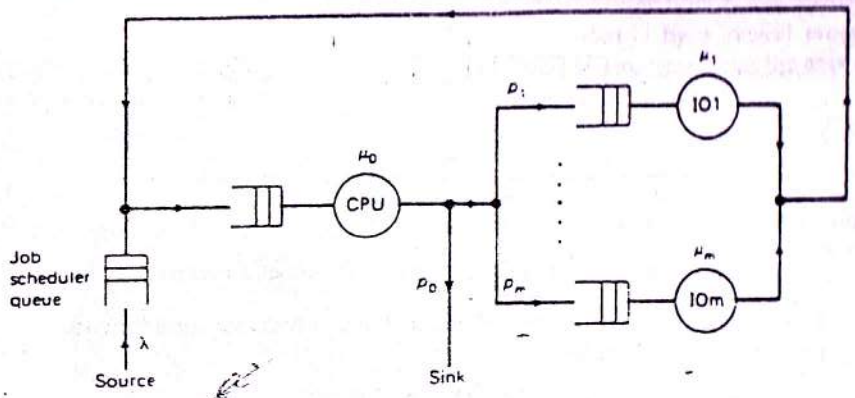


Figure 9.7 An open central server network with a blocking queue

[CHAN 1978b]. Nevertheless, an approximate solution to such a network is provided by the methods of the last section. If the external arrival rate λ is low, then the probability that a job has to wait in the scheduler queue will be low, and we expect the solution of Example 9.3 to be quite good. Thus, the model of Example 9.3 is a light-load approximation to the model of Figure 9.7. Let us now take the other extreme and assume a large value of λ, so that the probability that there is at least one customer in the job scheduler queue is very high.

We may then assume that the departure of a job from the active set immediately triggers the scheduling of an already waiting job into main memory. Thus, the closed network of Figure 9.8 will be a "good" approximation to the system of Figure 9.7, under heavy-load conditions. Each job circulating in this closed network is said to be an active job and must be allo-

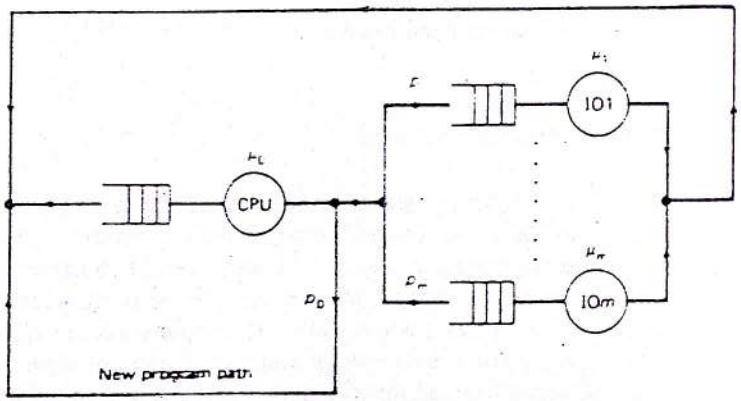


Figure 9.8 The (closed) central server model

cated a partition of main memory. The total number of active jobs is called **the degree (or level) of multiprogramming.**

Example 9.4

Let us return to the cyclic queuing model studied in the last chapter (Example 8.6), which is shown in Figure 9.9. Here, we choose to represent the state of the system by a pair, (k_0, k_1) where k_i denotes the number of jobs at node i $(i = 0, 1)$. Recall that $k_0 + k_1 = n$, the degree of multiprogramming. Unlike the two-node open queuing network (Figure 9.4a), the state space in this case is finite. The dot pattern on the $(k_0 - k_1)$ plane of Figure 9.10 represents the infinite-state space of the open network (Figure 9.4a) while the dot pattern on the line $k_0 + k_1 = n$ is the finite-state space of the cyclic (closed) queuing network being studied here.

The state diagram for the cyclic queuing model is shown in Figure 9.11. The balance equations are given by:

$$(\mu_1 + \mu_0 p_1) p(k_0, k_1) = \mu_0 p_1 p(k_0 + 1,\ k_1 - 1) + \mu_1 p(k_0 - 1,\ k_1 + 1),\quad k_0,\ k_1 > 0$$

$$\mu_1 p(0, n) = \mu_0 p_1 p(1,\ n - 1),$$

$$\mu_0 p_1 p(n,\ 0) = \mu_1 p(n - 1,\ 1).$$

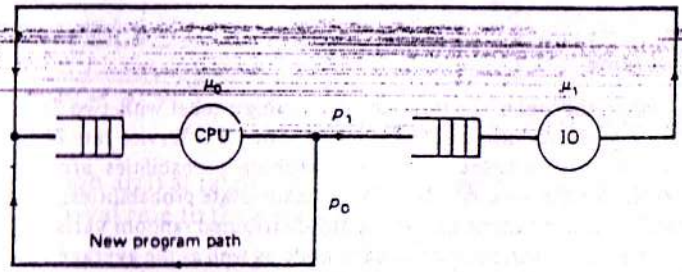


Figure 9.9 The closed cyclic queuing model

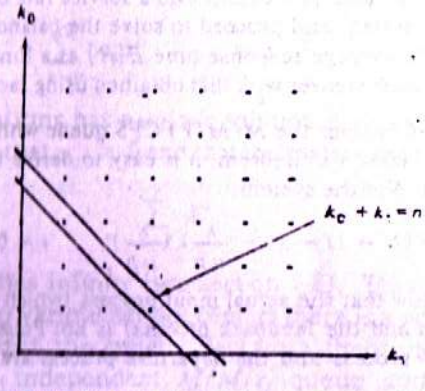


Figure 9.10 State spaces for two-stage networks

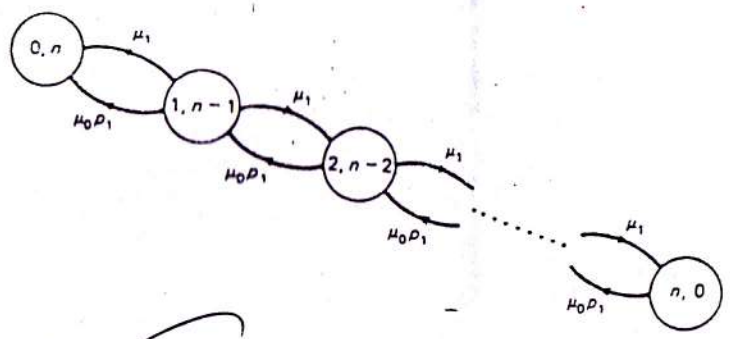


Figure 9.11 The state diagram for the closed cyclic queuing model

If we let $\rho_0 = a/\mu_0$ and $\rho_1 = ap_1/\mu_1$, where a is an arbitrary constant, then it can be verified by direct substitution that the steady-state probability $p(k_0, k_1)$ has the following product form:

$$p(k_0, k_1) = \frac{1}{C(n)} \rho_0^{k_0} \rho_1^{k_1}.$$

The normalizing constant $C(n)$ is chosen so that:

$$\sum_{\substack{k_0 + k_1 = n \\ k_0, k_1 \geq 0}} p(k_0, k_1) = 1.$$

The choice of the constant a is quite arbitrary in that the value of $p(k_0, k_1)$ will not change with a, although the intermediate values ρ_0, ρ_1, and $C(n)$ will depend upon a. If we define $\lambda_0 = a$ and $\lambda_1 = ap_1$, we may interpret the vector (λ_0, λ_1) as the relative throughputs of the corresponding nodes. Then $\rho_0 = (\lambda_0/\mu_0)$ and $\rho_1 = (\lambda_1/\mu_1)$ are interpreted as relative utilizations. Two popular choices of the constant a are $a = 1$ and $a = \mu_0$. Choosing $a = \mu_0$ we have $\rho_0 = 1$ and $\rho_1 = \mu_0 p_1/\mu_1$. Also

$$p(k_0, k_1) = \frac{1}{C(n)} \rho_1^{k_1}.$$

Using the normalization condition, we get:

$$1 = \frac{1}{C(n)} \sum_{k_1=0}^{n} \rho_1^{k_1} = \frac{1}{C(n)} \frac{1 - \rho_1^{n+1}}{1 - \rho_1}$$

or

$$C(n) = \begin{cases} \dfrac{1 - \rho_1^{n+1}}{1 - \rho_1}, & \rho_1 \neq 1 \\ n + 1, & \rho_1 = 1. \end{cases}$$

Now the CPU utilization U_0 may be expressed as:

$$U_0 = 1 - p(0, n) = 1 - \frac{\rho_1^n}{C(n)},$$

$$U_0 = \begin{cases} \dfrac{\rho_1 - \rho_1^{n+1}}{1 - \rho_1^{n+1}}, & \rho_1 \neq 1 \\ \dfrac{n}{n+1}, & \rho_1 = 1. \end{cases}$$

This agrees with the solution obtained in the last chapter [equation (8.38)]. The average throughput is given by:

$$E[T] = \mu_0 U_0 p_0.$$

#

Example 9.5

Consider the (closed) central server network shown in Figure 9.8. The state of the network is given by an $(m + 1)$-tuple, $(k_0, k_1, \ldots, k_m)$ where $k_i \geq 0$ is the number of jobs at server i (including any in service). Since the number of jobs in a closed network is fixed, we must further impose the constraint $\sum_{i=0}^{m} k_i = n$ on every state. Thus the state space of the network is finite, the number of states being equal to the number of partitions of n objects among $m + 1$ cells. You were asked to compute this number in problem 5, Section 1.12:

$$\binom{n+m}{m} = \frac{(n+m)!}{n!m!}. \qquad (9.7)$$

If we assume that service times at all servers are exponentially distributed, the stochastic process modeling the behavior of the network is a finite-state continuous-parameter Markov chain, which can be shown to be irreducible and recurrent non-null (assuming that $0 < p_i < 1$, $i = 0, 1, \ldots, m$). In principle, therefore, we can write down the steady-state balance equations and obtain the unique steady-state probabilities. However, the number of equations in this system will be equal to the number of states given by expression (9.7). This is a formidable number of states, even for relatively small values of n and m. Fortunately, Gordon and Newell [GORD 1967] have shown that such Markovian closed networks possess relatively simple product-form solutions.

In order to use their technique, we first analyze the behavior of a tagged program, ignoring all queues in the network. The movement of a tagged program through the network can be modeled by a discrete-parameter Markov chain with $m + 1$ states. The transition probability matrix X of this Markov chain is given by:

$$X = \begin{bmatrix} p_0 & p_1 & \cdots & p_m \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$