DISCRETE-EVENT SYSTEM SIMULATION

FOURTH EDITION

JERRY BANKS · JOHN S. CARSON II
BARRY L. NELSON · DAVID M. NICOL

Prentice Hall International Series in Industrial and Systems Engineering
W. J. Fabrycky and J. H. Mize, Series Editors

Discrete-Event System Simulation

Fourth Edition

Jerry Banks
Independent Consultant

John S. Carson II
Brooks Automation

Barry L. Nelson
Northwestern University

David M. Nicol
University of Illinois, Urbana-Champaign

Prentice Hall
is an imprint of

Contents

1

Introduction to Simulation

A *simulation* is the imitation of the operation of a real-world process or system over time. Whether done by hand or on a computer, simulation involves the generation of an artificial history of a system and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system.

The behavior of a system as it evolves over time is studied by developing a simulation *model*. This model usually takes the form of a set of assumptions concerning the operation of the system. These assumptions are expressed in mathematical, logical, and symbolic relationships between the *entities*, or objects of interest, of the system. Once developed and validated, a model can be used to investigate a wide variety of "what if" questions about the real-world system. Potential changes to the system can first be simulated, in order to predict their impact on system performance. Simulation can also be used to study systems in the design stage, before such systems are built. Thus, simulation modeling can be used both as an analysis tool for predicting the effect of changes to existing systems and as a design tool to predict the performance of new systems under varying sets of circumstances.

In some instances, a model can be developed which is simple enough to be "solved" by mathematical methods. Such solutions might be found by the use of differential calculus, probability theory, algebraic methods, or other mathematical techniques. The solution usually consists of one or more numerical parameters, which are called measures of performance of the system. However, many real-world systems are so complex that models of these systems are virtually impossible to solve mathematically. In these instances, numerical, computer-based simulation can be used to imitate the behavior of the system over time. From the simulation, data are collected as if a real system were being observed. This simulation-generated data is used to estimate the measures of performance of the system.

This book provides an introductory treatment of the concepts and methods of one form of simulation modeling—discrete-event simulation modeling. The first chapter initially discusses when to use simulation, its advantages and disadvantages, and actual areas of its application. Then the concepts of system and model are explored. Finally, an outline is given of the steps in building and using a simulation model of a system.

1.1 WHEN SIMULATION IS THE APPROPRIATE TOOL

The availability of special-purpose simulation languages, of massive computing capabilities at a decreasing cost per operation, and of advances in simulation methodologies have made simulation one of the most widely used and accepted tools in operations research and systems analysis. Circumstances under which simulation is the appropriate tool to nse have been discussed by many authors, from Naylor *et al.* [1966] to Shannon [1998]. Simulation can be used for the following purposes:

1. Simulation enables the study of, and experimentation with, the internal interactions of a complex system or of a subsystem within a complex system.
2. Informational, organizational, and environmental changes can be simulated, and the effect of these alterations on the model's behavior can be observed.
3. The knowledge gained during the designing of a simulation model could be of great value toward suggesting improvement in the system under investigation.
4. Changing simulation inputs and observing the resulting outputs can produce valuable insight into which variables are the most important and into how variables interact.
5. Simulation can be used as a pedagogical device to reinforce analytic solution methodologies.
6. Simulation can be used to experiment with new designs or policies before implementation, so as to prepare for what might happen.
7. Simulation can be used to verify analytic solutions.
8. Simulating different capabilities for a machine can help determine the requirements on it.
9. Simulation models designed for training make learning possible without the cost and disruption of on-the-job instruction.
10. Animation shows a system in simulated operation so that the plan can be visualized.
11. The modern system (factory, wafer fabrication plant, service organization, etc.) is so complex that its internal interactions can be treated only through simulation.

1.2 WHEN SIMULATION IS NOT APPROPRIATE

This section is based on an article by Banks and Gibson [1997], who gave ten rules for evaluating when simulation is not appropriate. The first rule indicates that simulation should not be used when the problem can be solved by common sense. An example is given of an automobile tag facility serving customers who arrive randomly at an average rate of 100/hour and are served at a mean rate of 12/hour. To determine the minimum number of servers needed, simulation is not necessary. Just compute 100/12 = 8.33 indicating that nine or more servers are needed.

The second rule says that simulation should not be used if the problem can be solved analytically. For example, under certain conditions, the average waiting time in the example above can be found from curves that were developed by Hillier and Lieberman [2002].

The next rule says that simulation should not be used if it is easier to perform direct experiments. An example of a fast-food drive-in restaurant is given where it was less expensive to stage a person taking orders using a hand-held terminal and voice communication to determine the effect of adding another order station on customer waiting time.

The fourth rule says not to use simulation if the costs exceed the savings. There are many steps in completing a simulation, as will be discussed in Section 1.11, and these must be done thoroughly. If a simulation study costs $20,000 and the savings might be $10,000, simulation would not be appropriate.

Rules five and six indicate that simulation should not be performed if the resources or time are not available. If the simulation is estimated to cost $20,000 and there is only $10,000 available, the suggestion is not to

venture into a simulation study. Similarly, if a decision in needed in two weeks and a simulation will take a month, the simulation study is not advised.

Simulation takes data, sometimes lots of data. If no data is available, not even estimates, simulation is not advised. The next rule concerns the ability to verify and validate the model. If there is not enough time or if the personnel are not available, simulation is not appropriate.

If managers have unreasonable expectations, if they ask for too much too soon, or if the power of simulation is overestimated, simulation might not be appropriate.

Last, if system behavior is too complex or can't be defined, simulation is not appropriate. Human behavior is sometimes extremely complex to model.

1.3 ADVANTAGES AND DISADVANTAGES OF SIMULATION

Simulation is intuitively appealing to a client because it mimics what happens in a real system or what is perceived for a system that is in the design stage. The output data from a simulation should directly correspond to the outputs that could be recorded from the real system. Additionally, it is possible to develop a simulation model of a system without dubious assumptions (such as the same statistical distribution for every random variable) of mathematically solvable models. For these and other reasons, simulation is frequently the technique of choice in problem solving.

In contrast to optimization models, simulation models are "run" rather than solved. Given a particular set of input and model characteristics, the model is run and the simulated behavior is observed. This process of changing inputs and model characteristics results in a set of scenarios that are evaluated. A good solution, either in the analysis of an existing system or in the design of a new system, is then recommended for implementation.

Simulation has many advantages, but some disadvantages. These are listed by Pegden, Shannon, and Sadowski [1995]. Some advantages are these:

1. New policies, operating procedures, decision rules, information flows, organizational procedures, and so on can be explored without disrupting ongoing operations of the real system.
2. New hardware designs, physical layouts, transportation systems, and so on can be tested without committing resources for their acquisition.
3. Hypotheses about how or why certain phenomena occur can be tested for feasibility.
4. Time can be compressed or expanded to allow for a speed-up or slow-down of the phenomena under investigation.
5. Insight can be obtained about the interaction of variables.
6. Insight can be obtained about the importance of variables to the performance of the system.
7. Bottleneck analysis can be performed to discover where work in process, information, materials, and so on are being delayed excessively.
8. A simulation study can help in understanding how the system operates rather than how individuals think the system operates.
9. "What if" questions can be answered. This is particularly useful in the design of new systems.

Some disadvantages are these:

1. Model building requires special training. It is an art that is learned over time and through experience. Furthermore, if two models are constructed by different competent individuals, they might have similarities, but it is highly unlikely that they will be the same.

2. Simulation results can be difficult to interpret. Most simulation outputs are essentially random variables (they are usually based on random inputs), so it can be hard to distinguish whether an observation is a result of system interrelationships or of randonmess.
3. Simulation modeling and analysis can be time consuming and expensive. Skimping on resources for modeling and analysis could result in a simulation model or analysis that is not sufficient to the task.
4. Simulation is used in some cases when an analytical solution is possible, or even preferable, as was discussed in Section 1.2. This might be particularly true in the simulation of some waiting lines where closed-form queueing models are available.

In defense of simulation, these four disadvantages, respectively, can be offset as follows:

1. Vendors of simulation software have been actively developing packages that contain models that need only input data for their operation. Such models have the generic tag "simulator" or "template."
2. Many simulation software vendors have developed output-analysis capabilities within their packages for performing very thorough analysis.
3. Simulation can be performed faster today than yesterday and will be even faster tomorrow, because of advances in hardware that permit rapid running of scenarios and because of advances in many simulation packages. For example, some simulation software contains constructs for modeling material handling that uses such transporters as fork-lift trucks, conveyors, and automated guided vehicles.
4. Closed-form models are not able to analyze most of the complex systems that are encountered in practice. In many years of consulting practice by two of the authors, not one problem was encountered that could have been solved by a closed-form solution.

1.4 AREAS OF APPLICATION

The applications of simulation are vast. The Winter Simulation Conference (WSC) is an excellent way to learn more about the latest in simulation applications and theory. There are also numerous tutorials at both the beginning and the advanced levels. WSC is sponsored by six technical societies and the National Institute of Standards and Technology (NIST). The technical societies are American Statistical Association (ASA), Association for Computing Machinery/Special Interest Group on Simulation (ACM/SIGSIM), Institute of Electrical and Electronics Engineers: Computer Society (IEEE/CS), Institute of Electrical and Electronics Engineers: Systems, Man and Cybernetics Society (IEEE/SMCS), Institute of Industrial Engineers (IIE), Institute for Operations Research and the Management Sciences: College on Simulation (INFORMS/CS) and The Society for Computer Simulation (SCS). Note that IEEE is represented by two bodies. Information about the upcoming WSC can be obtained from `www.wintersim.org`. WSC programs with full papers are available from `www.informs-cs.org/wscpapers.html`. Some presentations, by area, from a recent WSC are listed next:

Manufacturing Applications
- Dynamic modeling of continuous manufacturing systems, using analogies to electrical systems
- Benchmarking of a stochastic production planning model in a simulation test bed
- Paint line color change reduction in automobile assembly
- Modeling for quality and productivity in steel cord manufacturing
- Shared resource capacity analysis in biotech manufacturing
- Neutral information model for simulating machine shop operations

Semiconductor Manufacturing
- Constant time interval production planning with application to work-in-process control
- Accelerating products under due-date oriented dispatching rules
- Design framework for automated material handling systems in 300-mm wafer fabrication factories

- Making optimal design decisions for next-generation dispensing tools
- Application of cluster tool modeling in a 300-mm wafer fabrication factory
- Resident-entity based simulation of batch chamber tools in 300-mm semiconductor manufacturing

Construction Engineering and Project Management
- Impact of multitasking and merge bias on procurement of complex equipment
- Application of lean concepts and simulation for drainage operations maintenance crews
- Building a virtual shop model for steel fabrication
- Simulation of the residential lumber supply chain

Military Applications
- Frequency-based design for terminating simulations: A peace-enforcement example
- A multibased framework for supporting military-based interactive simulations in 3D environments
- Specifying the behavior of computer-generated forces without programming
- Fidelity and validity: Issues of human behavioral representation
- Assessing technology effects on human performance through trade-space development and evaluation
- Impact of an automatic logistics system on the sortie-generation process
- Research plan development for modeling and simulation of military operations in urban terrain

Logistics, Supply Chain, and Distribution Applications
- Inventory analysis in a server–computer manufacturing environment
- Comparison of bottleneck detection methods for AGV systems
- Semiconductor supply-network simulation
- Analysis of international departure passenger flows in an airport terminal
- Application of discrete simulation techniques to liquid natural gas supply chains
- Online simulation of pedestrian flow in public buildings

Transportation Modes and Traffic
- Simulating aircraft-delay absorption
- Runway schedule determination by simulation optimization
- Simulation of freeway merging and diverging behavior
- Modeling ambulance service of the Austrian Red Cross
- Simulation modeling in support of emergency firefighting in Norfolk
- Modeling ship arrivals in ports
- Optimization of a barge transportation system for petroleum delivery
- Iterative optimization and simulation of barge traffic on an inland waterway

Business Process Simulation
- Agent-based modeling and simulation of store performance for personalized pricing
- Visualization of probabilistic business models
- Modeling and simulation of a telephone call center
- Using simulation to approximate subgradients of convex performance measures in service systems
- Simulation's role in baggage screening at airports
- Human-fatigue risk simulations in continuous operations
- Optimization of a telecommunications billing system
- Segmenting the customer base for maximum returns

Health Care
- Modeling front office and patient care in ambulatory health care practices
- Evaluation of hospital operations between the emergency department and a medical telemetry unit
- Estimating maximum capacity in an emergency room
- Reducing the length of stay in an emergency department
- Simulating six-sigma improvement ideas for a hospital emergency department
- A simulation–integer-linear-programming-based tool for scheduling emergency room staff

Some general trends in simulation applications are as follows: At present, simulation for risk analysis is growing, including in such areas as insurance, options pricing, and portfolio analysis. Another growing area is call-center analysis, which is not amenable to queuing models because of its complexity. Simulation of large-scale systems such as the internet backbone, wireless networks, and supply chains are growing as hardware and software increase their capability to handle extremely large numbers of entities in a reasonable time.

Lastly, simulation models of automated material handling systems (AMHS) are being used as test beds for the development and functional testing of control-system software. Called an emulation, the simulation model is connected in real time to the control-system software or to a software emulator; it is used to provide the same responses to a control system as the real AMHS does (for example, a box blocking or clearing a photo eye, or a command to start picking an order). Software development can begin much earlier during AMHS installation and commissioning, to reduce the time spent in the field on trying to debug control software while attempting to ramp up a new system or continue running an existing one. Models have been driven by control systems at various levels—from high-level supervisory systems, such as warehouse management systems (WMS) or AGV dispatching systems, to such low-level control as programmable logic controllers (PLCs) controlling merges on a conveyor system.

1.5 SYSTEMS AND SYSTEM ENVIRONMENT

To model a system, it is necessary to understand the concept of a system and the system boundary. A *system* is defined as a group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose. An example is a production system manufacturing automobiles. The machines, component parts, and workers operate jointly along an assembly line to produce a high-quality vehicle.

A system is often affected by changes occurring outside the system. Such changes are said to occur in the *system environment* [Gordon, 1978]. In modeling systems, it is necessary to decide on the *boundary* between the system and its environment. This decision may depend on the purpose of the study.

In the case of the factory system, for example, the factors controlling the arrival of orders may be considered to be outside the influence of the factory and therefore part of the environment. However, if the effect of supply on demand is to be considered, there will be a relationship between factory output and arrival of orders, and this relationship must be considered an activity of the system. Similarly, in the case of a bank system, there could be a limit on the maximum interest rate that can be paid. For the study of a single bank, this would be regarded as a constraint imposed by the environment. In a study of the effects of monetary laws on the banking industry, however, the setting of the limit would be an activity of the system. [Gordon, 1978]

1.6 COMPONENTS OF A SYSTEM

In order to understand and analyze a system, a number of terms need to be defined. An *entity* is an object of interest in the system. An *attribute* is a property of an entity. An *activity* represents a time period of specified length. If a bank is being studied, customers might be one of the entities, the balance in their checking accounts might be an attribute, and making deposits might be an activity.

The collection of entities that compose a system for one study might only be a subset of the overall system for another study [Law and Kelton, 2000]. For example, if the aforementioned bank is being studied to determine the number of tellers needed to provide for paying and receiving, the system can be defined as that portion of the bank consisting of the regular tellers and the customers waiting in line. If the purpose of the study is expanded to determine the number of special tellers needed (to prepare cashier's checks, to sell traveler's checks, etc.), the definition of the system must be expanded.

The *state* of a system is defined to be that collection of variables necessary to describe the system at any time, relative to the objectives of the study. In the study of a bank, possible state variables are the number of busy tellers, the number of customers waiting in line or being served, and the arrival time of the next customer. An *event* is defined as an instantaneous occurrence that might change the state of the system. The term *endogenous* is used to describe activities and events occurring within a system, and the term *exogenous* is used to describe activities and events in the environment that affect the system. In the bank study, the arrival of a customer is an exogenous event, and the completion of service of a customer is an endogenous event.

Table 1.1 lists examples of entities, attributes, activities, events, and state variables for several systems. Only a partial listing of the system components is shown. A complete list cannot be developed unless the purpose of the study is known. Depending on the purpose, various aspects of the system will be of interest, and then the listing of components can be completed.

1.7 DISCRETE AND CONTINUOUS SYSTEMS

Systems can be categorized as discrete or continuous. "Few systems in practice are wholly discrete or continuous, but since one type of change predominates for most systems, it will usually be possible to classify a system as being either discrete or continuous" [Law and Kelton, 2000]. A *discrete system* is one in which the state variable(s) change only at a discrete set of points in time. The bank is an example of a discrete system: The state variable, the number of customers in the bank, changes only when a customer arrives or when the service provided a customer is completed. Figure 1.1 shows how the number of customers changes only at discrete points in time.

A *continuous system* is one in which the state variable(s) change continuously over time. An example is the head of water behind a dam. During and for some time after a rain storm, water flows into the lake behind the dam. Water is drawn from the dam for flood control and to make electricity. Evaporation also decreases the water level. Figure 1.2 shows how the state variable *head of water behind the dam* changes for this continuous system.

1.8 MODEL OF A SYSTEM

Sometimes it is of interest to study a system to understand the relationships between its components or to predict how the system will operate under a new policy. To study the system, it is sometimes possible to experiment with the system itself. However, this is not always possible. A new system might not yet exist; it could be in only hypothetical form or at the design stage. Even if the system exists, it might be impractical to experiment with it. For example, it might not be wise or possible to double the unemployment rate to discover the effect of employment on inflation. In the case of a bank, reducing the numbers of tellers to study the effect on the length of waiting lines might infuriate the customers so greatly that they move their accounts to a competitor. Consequently, studies of systems are often accomplished with a model of a system.

We had a consulting job for the simulation of a redesigned port in western Australia. At $200 millions for a loading/unloading berth, it's not advisable to invest that amount only to find that the berth is inadequate for the task.

A *model* is defined as a representation of a system for the purpose of studying the system. For most studies, it is only necessary to consider those aspects of the system that affect the problem under investigation. These aspects are represented in a model of the system; the model, by definition, is a simplification of the system. On the other hand, the model should be sufficiently detailed to permit valid conclusions to be drawn about the real system. Different models of the same system could be required as the purpose of investigation changes.

Table 1.1 Examples of Systems and Components

System	*Entities*	*Attributes*	*Activities*	*Events*	*State Variables*
Banking	Customers	Checking-account balance	Making deposits	Arrival; departure	Number of busy tellers; number of customers waiting
Rapid rail	Riders	Origination; destination	Traveling	Arrival at station; arrival at destination	Number of riders waiting at each station; number of riders in transit
Production	Machines	Speed; capacity; breakdown rate	Welding; stamping	Breakdown	Status of machines (busy, idle, or down)
Communications	Messages	Length; destination	Transmitting	Arrival at destination	Number waiting to be transmitted
Inventory	Warehouse	Capacity	Withdrawing	Demand	Levels of inventory; backlogged demands

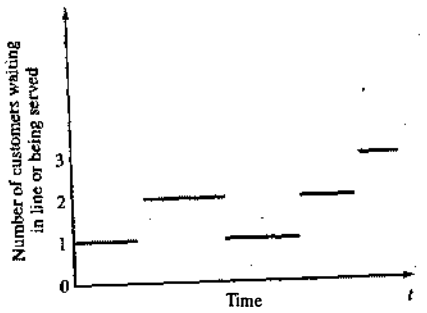


Figure 1.1 Discrete-system state variable.

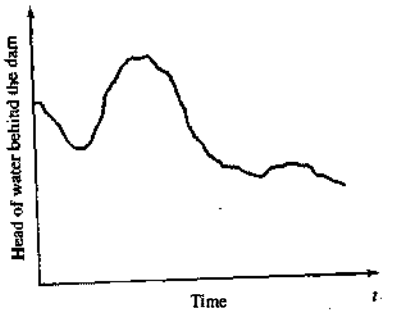


Figure 1.2 Continuous-system state variable.

Just as the components of a system were entities, attributes, and activities, models are represented similarly. However, the model contains only those components that are relevant to the study. The components of a model are discussed more extensively in Chapter 3.

1.9 TYPES OF MODELS

Models can be classified as being mathematical or physical. A mathematical model uses symbolic notation and mathematical equations to represent a system. A simulation model is a particular type of mathematical model of a system.

Simulation models may be further classified as being static or dynamic, deterministic or stochastic, and discrete or continuous. A *static* simulation model, sometimes called a Monte Carlo simulation, represents a system at a particular point in time. *Dynamic* simulation models represent systems as they change over time. The simulation of a bank from 9:00 A.M. to 4:00 P.M. is an example of a dynamic simulation.

Simulation models that contain no random variables are classified as *deterministic*. Deterministic models have a known set of inputs, which will result in a unique set of outputs. Deterministic arrivals would occur at a dentist's office if all patients arrived at the scheduled appointment time. A *stochastic* simulation model has one or more random variables as inputs. Random inputs lead to random outputs. Since the outputs are

random, they can be considered only as estimates of the true characteristics of a model. The simulation of a bank would usually involve random interarrival times and random service times. Thus, in a stochastic simulation, the output measures—the average number of people waiting, the average waiting time of a customer—must be treated as statistical estimates of the true characteristics of the system.

Discrete and continuous systems were defined in Section 1.7. Discrete and continuous models are defined in an analogous manner. However, a discrete simulation model is not always used to model a discrete system, nor is a continuous simulation model always used to model a continuous system. Tanks and pipes are modeled discretely by some software vendors, even though we know that fluid flow is continuous. In addition, simulation models may be mixed, both discrete and continuous. The choice of whether to use a discrete or continuous (or both discrete and continuous) simulation model is a function of the characteristics of the system and the objective of the study. Thus, a communication channel could be modeled discretely if the characteristics and movement of each message were deemed important. Conversely, if the flow of messages in aggregate over the channel were of importance, modeling the system via continuous simulation could be more appropriate. The models considered in this text are discrete, dynamic, and stochastic.

1.10 DISCRETE-EVENT SYSTEM SIMULATION

This is a textbook about discrete-event system simulation. Discrete-event systems simulation is the modeling of systems in which the state variable changes only at a discrete set of points in time. The simulation models are analyzed by numerical methods rather than by analytical methods. *Analytical* methods employ the deductive reasoning of mathematics to "solve" the model. For example, differential calculus can be used to compute the minimum-cost policy for some inventory models. *Numerical* methods employ computational procedures to "solve" mathematical models. In the case of simulation models, which employ numerical methods, models are "run" rather than solved—that is, an artificial history of the system is generated from the model assumptions, and observations are collected to be analyzed and to estimate the true system performance measures. Real-world simulation models are rather large, and the amount of data stored and manipulated is vast, so such runs are usually conducted with the aid of a computer. However, much insight can be obtained by simulating small models manually.

In summary, this textbook is about discrete-event system simulation in which the models of interest are analyzed numerically, usually with the aid of a computer.

1.11 STEPS IN A SIMULATION STUDY

Figure 1.3 shows a set of steps to guide a model builder in a thorough and sound simulation study. Similar figures and discussion of steps can be found in other sources [Shannon, 1975; Gordon, 1978; Law and Kelton, 2000]. The number beside each symbol in Figure 1.3 refers to the more detailed discussion in the text. The steps in a simulation study are as follows:

Problem formulation. Every study should begin with a statement of the problem. If the statement is provided by the policymakers, or those that have the problem, the analyst must ensure that the problem being described is clearly understood. If a problem statement is being developed by the analyst, it is important that the policymakers understand and agree with the formulation. Although not shown in Figure 1.3, there are occasions where the problem must be reformulated as the study progresses. In many instances, policymakers and analysts are aware that there is a problem long before the nature of the problem is known.

Setting of objectives and overall project plan. The objectives indicate the questions to be answered by simulation. At this point, a determination should be made concerning whether simulation is the

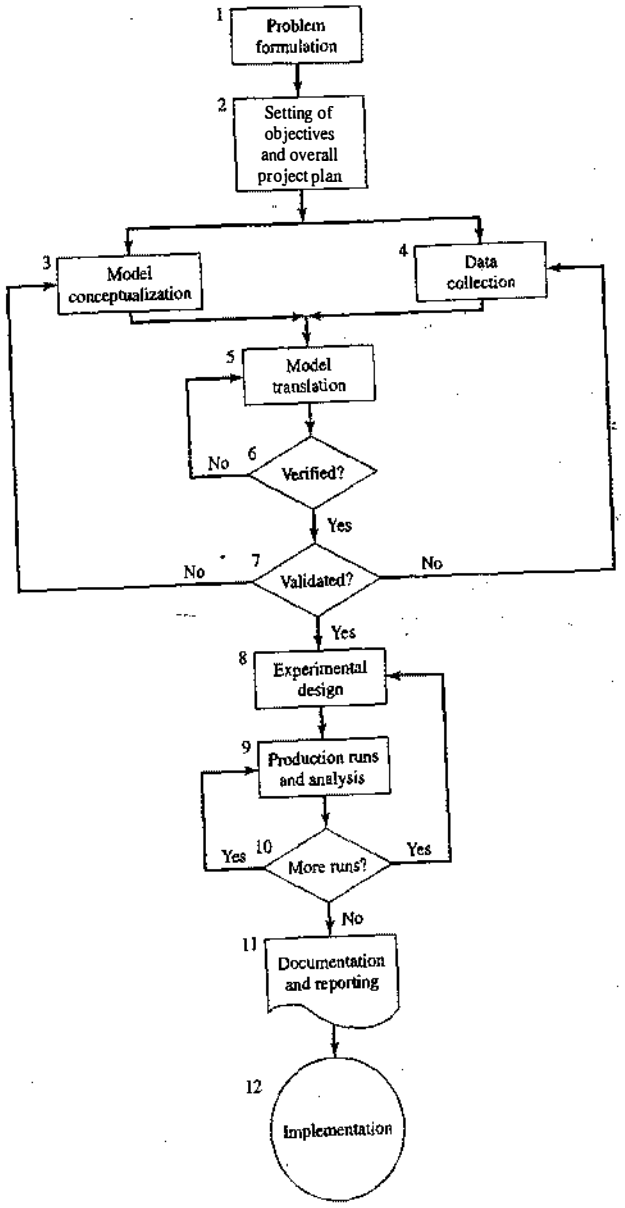


Figure 1.3 Steps in a simulation study.

appropriate methodology for the problem as formulated and objectives as stated. Assuming that it is decided that simulation is appropriate, the overall project plan should include a statement of the alternative systems to be considered and of a method for evaluating the effectiveness of these alternatives. It should also include the plans for the study in terms of the number of people involved, the cost of the study, and the number of days required to accomplish each phase of the work, along with the results expected at the end of each stage.

Model conceptualization. The construction of a model of a system is probably as much art as science. Pritsker [1998] provides a lengthy discussion of this step. "Although it is not possible to provide a set of instructions that will lead to building successful and appropriate models in every instance, there are some general guidelines that can be followed" [Morris, 1967]. The art of modeling is enhanced by an ability to abstract the essential features of a problem, to select and modify basic assumptions that characterize the system, and then to enrich and elaborate the model until a useful approximation results. Thus, it is best to start with a simple model and build toward greater complexity. However, the model complexity need not exceed that required to accomplish the purposes for which the model is intended. Violation of this principle will only add to model-building and computer expenses. It is not necessary to have a one-to-one mapping between the model and the real system. Only the essence of the real system is needed.

It is advisable to involve the model user in model conceptualization. Involving the model user will both enhance the quality of the resulting model and increase the confidence of the model user in the application of the model. (Chapter 2 describes a number of simulation models. Chapter 6 describes queueing models that can be solved analytically. However, only experience with real systems—versus textbook problems—can "teach" the art of model building.)

Data collection. There is a constant interplay between the construction of the model and the collection of the needed input data [Shannon, 1975]. As the complexity of the model changes, the required data elements can also change. Also, since data collection takes such a large portion of the total time required to perform a simulation, it is necessary to begin it as early as possible, usually together with the early stages of model building.

The objectives of the study dictate, in a large way, the kind of data to be collected. In the study of a bank, for example, if the desire is to learn about the length of waiting lines as the number of tellers change, the types of data needed would be the distributions of interarrival times (at different times of the day), the service-time distributions for the tellers, and historic distributions on the lengths of waiting lines under varying conditions. This last type of data will be used to validate the simulation model. (Chapter 9 discusses data collection and data analysis; Chapter 5 discusses statistical distributions that occur frequently in simulation modeling. See also an excellent discussion by Henderson [2003].)

Model translation. Most real-world systems result in models that require a great deal of information storage and computation, so the model must be entered into a computer-recognizable format. We use the term "program" even though it is possible to accomplish the desired result in many instances with little or no actual coding. The modeler must decide whether to program the model in a simulation language, such as GPSS/H (discussed in Chapter 4), or to use special-purpose simulation software. For manufacturing and material handling, Chapter 4 discusses Arena®, AutoMod™, Extend™, Flexsim, MicroSaint, ProModel®, Quest®, SIMUL8®, and WITNESS™. Simulation languages are powerful and flexible. However, if the problem is amenable to solution with the simulation software, the model development time is greatly reduced. Furthermore, most of the simulation-software packages have added features that enhance their flexibility, although the amount of flexibility varies greatly.

Verified? Verification pertains to the computer program prepared for the simulation model. Is the computer program performing properly? With complex models, it is difficult, if not impossible, to translate a model successfully in its entirety without a good deal of debugging; if the input parameters and logical structure of the model are correctly represented in the computer, verification has been completed. For the most part, common sense is used in completing this step. (Chapter 10 discusses verification of simulation models, and Balci [2003] also discusses this topic.)

Validated? Validation usually is achieved through the calibration of the model, an iterative process of comparing the model against actual system behavior and using the discrepancies between the two, and the insights gained, to improve the model. This process is repeated until model accuracy is judged acceptable. In the example of a bank previously mentioned, data was collected concerning the length of waiting lines under current conditions. Does the simulation model replicate this system measure? This is one means of validation. (Chapter 10 discusses the validation of simulation models, and Balci [2003] also discusses this topic.)

Experimental design. The alternatives that are to be simulated must be determined. Often, the decision concerning which alternatives to simulate will be a function of runs that have been completed and analyzed. For each system design that is simulated, decisions need to be made concerning the length of the initialization period, the length of simulation runs, and the number of replications to be made of each run. (Chapters 11 and 12 discuss issues associated with the experimental design, and Kleijnen [1998] discusses this topic extensively.)

Production runs and analysis. Production runs, and their subsequent analysis, are used to estimate measures of performance for the system designs that are being simulated. (Chapters 11 and 12 discuss the analysis of simulation experiments, and Chapter 4 discusses software to aid in this step, including AutoStat (in AutoMod), OptQuest (in several pieces of simulation software), SimRunner (in ProModel), and WITNESS Optimizer (in WITNESS).

More Runs? Given the analysis of runs that have been completed, the analyst determines whether additional runs are needed and what design those additional experiments should follow.

Documentation and reporting. There are two types of documentation: program and progress. Program documentation is necessary for numerous reasons. If the program is going to be used again by the same or different analysts, it could be necessary to understand how the program operates. This will create confidence in the program, so that model users and policymakers can make decisions based on the analysis. Also, if the program is to be modified by the same or a different analyst, this step can be greatly facilitated by adequate documentation. One experience with an inadequately documented program is usually enough to convince an analyst of the necessity of this important step. Another reason for documenting a program is so that model users can change parameters at will in an effort to learn the relationships between input parameters and output measures of performance or to discover the input parameters that "optimize" some output measure of performance.

Musselman [1998] discusses progress reports that provide the important, written history of a simulation project. Project reports give a chronology of work done and decisions made. This can prove to be of great value in keeping the project on course.

Musselman suggests frequent reports (monthly, at least) so that even those not involved in the day-to-day operation can keep abreast. The awareness of these others can often enhance the successful completion of the project by surfacing misunderstandings early, when the problem can be solved easily. Musselman also suggests maintaining a project log providing a comprehensive record of accomplishments, change requests, key decisions, and other items of importance.

On the reporting side, Musselman suggests frequent deliverables. These may or may not be the results of major accomplishments. His maxim is that "it is better to work with many intermediate milestones than with one absolute deadline." Possibilities prior to the final report include a model specification, prototype demonstrations, animations, training results, intermediate analyses, program documentation, progress reports, and presentations. He suggests that these deliverables should be timed judiciously over the life of the project.

The result of all the analysis should be reported clearly and concisely in a final report. This will enable the model users (now, the decision makers) to review the final formulation, the alternative systems that were addressed, the criterion by which the alternatives were compared, the results of the experiments, and the recommended solution to the problem. Furthermore, if decisions have to be justified at a higher level, the final report should provide a vehicle of certification for the model user/decision maker and add to the credibility of the model and of the model-building process.

Implementation. The success of the implementation phase depends on how well the previous 11 steps have been performed. It is also contingent upon how thoroughly the analyst has involved the ultimate model user during the entire simulation process. If the model user has been thoroughly involved during the model-building process and if the model user understands the nature of the model and its outputs, the likelihood of a vigorous implementation is enhanced [Pritsker, 1995]. Conversely, if the model and its underlying assumptions have not been properly communicated, implementation will probably suffer, regardless of the simulation model's validity.

The simulation-model building process shown in Figure 1.3 can be broken down into four phases. The first phase, consisting of steps 1 (Problem Formulation) and 2 (Setting of Objective and Overall Design), is a period of discovery or orientation. The initial statement of the problem is usually quite "fuzzy," the initial objectives will usually have to be reset, and the original project plan will usually have to be fine-tuned. These recalibrations and clarifications could occur in this phase, or perhaps will occur after or during another phase (i.e., the analyst might have to restart the process).

The second phase is related to model building and data collection and includes steps 3 (Model Conceptualization), 4 (Data Collection), 5 (Model Translation), 6 (Verification), and 7 (Validation). A continuing interplay is required among the steps. Exclusion of the model user during this phase can have dire implications at the time of implementation.

The third phase concerns the running of the model. It involves steps 8 (Experimental Design), 9 (Production Runs and Analysis), and 10 (Additional Runs). This phase must have a thoroughly conceived plan for experimenting with the simulation model. A discrete-event stochastic simulation is in fact a statistical experiment. The output variables are estimates that contain random error, and therefore a proper statistical analysis is required. Such a philosophy is in contrast to that of the analyst who makes a single run and draws an inference from that single data point.

The fourth phase, implementation, involves steps 11 (Documentation and Reporting) and 12 (Implementation). Successful implementation depends on continual involvement of the model user and on the successful completion of every step in the process. Perhaps the most crucial point in the entire process is step 7 (Validation), because an invalid model is going to lead to erroneous results, which, if implemented, could be dangerous, costly, or both.

REFERENCES

BALCI, O. [2003], "Verification, Validation, and Certification of Modeling and Simulation Applications," in *Proceedings of the Winter Simulation Conference*, Ed., S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, New Orleans, LA, Dec. 7–10, pp. 150–158.

BANKS, J., AND R. R. GIBSON [1997], "Don't Simulate When: 10 Rules for Determining when Simulation Is Not Appropriate," *IIE Solutions*, September.

GORDON, G. [1978], *System Simulation*, 2d ed., Prentice-Hall, Englewood Cliffs, NJ.

HENDERSON, S. G. [2003], "Input Model Uncertainty: Why Do We Care and What Should We Do About It? " in *Proceedings of the Winter Simulation Conference*, Ed., S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, New Orleans, LA, Dec. 7–10, pp. 90–100.

HILLIER, F. S., AND G. J. LIEBERMAN [2002], *Introduction to Operations Research*, 7th ed., McGraw-Hill, New York.

KLEIJNEN, J. P. C. [1998], "Experimental Design for Sensitivity Analysis, Optimization, and Validation of Simulation Models," in *Handbook of Simulation*, Ed., Jerry Banks, John Wiley, New York.

LAW, A. M., AND W. D. KELTON [2000], *Simulation Modeling and Analysis*, 3d ed., McGraw-Hill, New York.

MORRIS, W. T. [1967], "On the Art of Modeling," *Management Science*, Vol. 13, No. 12.

MUSSELMAN, K. J. [1998], "Guidelines for Success," in *Handbook of Simulation*, Ed., Jerry Banks, John Wiley, New York.

NAYLOR, T. H., J. L. BALINTFY, D. S. BURDICK, AND K. CHU [1966], *Computer Simulation Techniques*, Wiley, New York.

PEGDEN, C. D., R. E. SHANNON, AND R. P. SADOWSKI [1995], *Introduction to Simulation Using SIMAN*, 2d ed., McGraw-Hill, New York.

PRITSKER, A. A. B. [1995], *Introduction to Simulation and SLAM II*, 4th ed., Wiley & Sons, New York.

PRITSKER, A. A. B. [1998], "Principles of Simulation Modeling," in *Handbook of Simulation*, Ed., Jerry Banks, John Wiley, New York.

SHANNON, R. E. [1975], *Systems Simulation: The Art and Science*, Prentice-Hall, Englewood Cliffs, NJ.

SHANNON, R. E. [1998], "Introduction to the Art and Science of Simulation," in *Proceedings of the Winter Simulation Conference*, Eds., D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Washington, DC, Dec. 13–16, pp. 7–14.

EXERCISES

1. Name entities, attributes, activities, events, and state variables for the following systems:
   (a) University library
   (b) Bank
   (c) Call center
   (d) Hospital blood bank
   (e) Departmental store
   (f) Fire service station
   (g) Airport
   (h) Software organization
2. Consider the simulation process shown in Figure 1.3.
   (a) Reduce the steps by at least two by combining similar activities. Give your rationale.
   (b) Increase the steps by at least two by separating current steps or enlarging on existing steps. Give your rationale.
3. A simulation of a major traffic intersection is to be conducted, with the objective of improving the current traffic flow. Provide three iterations, in increasing order of complexity, of steps 1 and 2 in the simulation process of Figure 1.3.
4. A simulation is to be conducted of cooking a spaghetti dinner to discover at what time a person should start in order to have the meal on the table by 7:00 P.M. Read a recipe for preparing a spaghetti dinner (or ask a friend or relative for the recipe). As best you can, trace what you understand to be needed, in the data-collection phase of the simulation process of Figure 1.3, in order to perform a simulation in which the model includes each step in the recipe. What are the events, activities, and state variables in this system?
5. List down the events and activities applying for master's program in a university.
6. Read an article on the application of simulation related to your major area of study or interest, in the current WSC Proceedings, and prepare a report on how the author accomplishes the steps given in Figure 1.3.

7. Get a copy of a recent WSC Proceedings and report on the different applications discussed in an area of interest to you.
8. Get a copy of a recent WSC Proceedings and report on the most unusual application that you can find.
9. Go to the Winter Simulation Conference website at: `www.wintersim.org` and address the following:
   (a) What advanced tutorials were offered at the previous WSC or are planned at the next WSC?
   (b) Where and when will the next WSC be held?
10. Go to the Winter Simulation Conference website at `www.wintersim.org` and address the following:
   (a) When was the largest (in attendance) WSC, and how many attended?
   (b) In what calendar year, from the beginning of WSC, was there no Conference?
   (c) What was the largest expanse of time, from the beginning of WSC, between occurrences of the Conference?
   (d) Beginning with the 25th WSC, can you discern a pattern for the location of the Conference?
11. Search the web for "Applications of discrete simulation" and prepare a report based on the findings.
12. Search the web for "Manufacturing simulation" and prepare a report based on the findings.
13. Search the web for "Call center simulation" and prepare a report based on the findings.

2

Simulation Examples

This chapter presents several examples of simulations that can be performed by devising a simulation table either manually or with a spreadsheet. The simulation table provides a systematic method for tracking system state over time. These examples provide insight into the methodology of discrete-system simulation and the descriptive statistics used for predicting system performance.

The simulations in this chapter are carried out by following three steps:

1. Determine the characteristics of each of the inputs to the simulation. Quite often, these are modeled as probability distributions, either continuous or discrete.
2. Construct a simulation table. Each simulation table is different, for each is developed for the problem at hand. An example of a simulation table is shown in Table 2.1. In this example, there are p inputs, x_{ij}, $j = 1, 2, \ldots, p$, and one response, y_i, for each of repetitions (or, trials) $i = 1, 2, \ldots, n$. Initialize the table by filling in the data for repetition 1.
3. For each repetition i, generate a value for each of the p inputs, and evaluate the function, calculating a value of the response y_i. The input values may be computed by sampling values from the distributions chosen in step 1. A response typically depends on the inputs and one or more previous responses.

This chapter gives a number of simulation examples in queueing, inventory, reliability, and network analysis. The two queueing examples provide a single-server and two-server system, respectively. (Chapter 6 provides more insight into queueing models.) The first of the inventory examples involves a problem that has a closed-form solution; thus, the simulation solution can be compared to the mathematical solution. The second inventory example pertains to the classic order-level model.

Next, there is an example that introduces the concept of random normal numbers and a model for the simulation of lead-time demand. The examples conclude with the analysis of a network.

Table 2.1 Simulation Table

Repetitions	Inputs x_{i1}	x_{i2}	...	x_{ij}	...	x_{ip}	Response y_i
1							
2							
3							
.							
.							
.							
n							

2.1 SIMULATION OF QUEUEING SYSTEMS

A queueing system is described by its calling population, the nature of the arrivals, the service mechanism, the system capacity, and the queueing discipline. These attributes of a queueing system are described in detail in Chapter 6. A simple single-channel queueing system is portrayed in Figure 2.1.

In the single-channel queue, the calling population is infinite; that is, if a unit leaves the calling population and joins the waiting line or enters service, there is no change in the arrival rate of other units that could need service. Arrivals for service occur one at a time in a random fashion; once they join the waiting line, they are eventually served. In addition, service times are of some random length according to a probability distribution which does not change over time. The system capacity has no limit, meaning that any number of units can wait in line. Finally, units are served in the order of their arrival (often called FIFO: first in, first out) by a single server or channel.

Arrivals and services are defined by the distribution of the time between arrivals and the distribution of service times, respectively. For any simple single- or multichannel queue, the overall effective arrival rate must be less than the total service rate, or the waiting line will grow without bound. When queues grow without bound, they are termed "explosive" or unstable. (In some re-entrant queueing networks in which units return a number of times to the same server before finally exiting from the system, the condition that arrival rate be less than service rate might not guarantee stability. See Harrison and Nguyen [1995] for more explanation. Interestingly, this type of instability was noticed first, not in theory, but in actual manufacturing in semiconductor manufacturing plants.) More complex situations can occur—for example, arrival rates that are greater than service rates for short periods of time, or networks of queues with routing. However, this chapter sticks to the most basic queues.

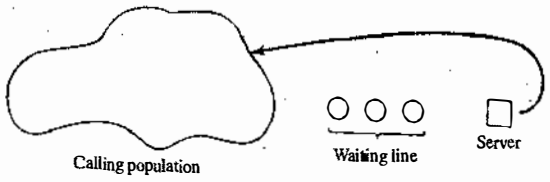


Figure 2.1 Queueing system.

Prior to our introducing several simulations of queueing systems, it is necessary to understand the concepts of system state, events, and simulation clock. (These concepts are studied systematically in Chapter 3.) The state of the system is the number of units in the system and the status of the server, busy or idle. An event is a set of circumstances that causes an instantaneous change in the state of the system. In a single-channel queueing system, there are only two possible events that can affect the state of the system. They are the entry of a unit into the system (the arrival event) and the completion of service on a unit (the departure event). The queueing system includes the server, the unit being serviced (if one is being serviced), and the units in the queue (if any are waiting). The simulation clock is used to track simulated time.

If a unit has just completed service, the simulation proceeds in the manner shown in the flow diagram of Figure 2.2. Note that the server has only two possible states: It is either busy or idle.

The arrival event occurs when a unit enters the system. The flow diagram for the arrival event is shown in Figure 2.3. The unit will find the server either idle or busy; therefore, either the unit begins service immediately, or it enters the queue for the server. The unit follows the course of action shown in Figure 2.4. If the server is busy, the unit enters the queue. If the server is idle and the queue is empty, the unit begins service. It is not possible for the server to be idle while the queue is nonempty.

After the completion of a service, the server either will become idle or will remain busy with the next unit. The relationship of these two outcomes to the status of the queue is shown in Figure 2.5. If the queue is not empty, another unit will enter the server and it will be busy. If the queue is empty, the server will be idle

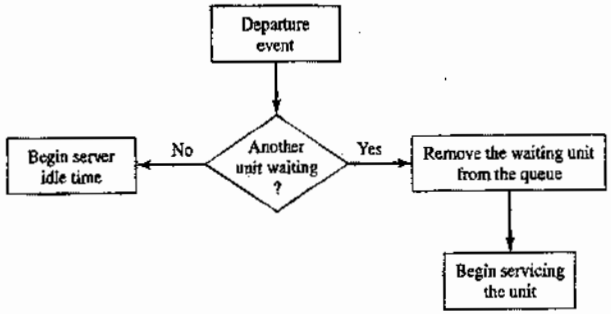


Figure 2.2 *Service just completed* flow diagram.

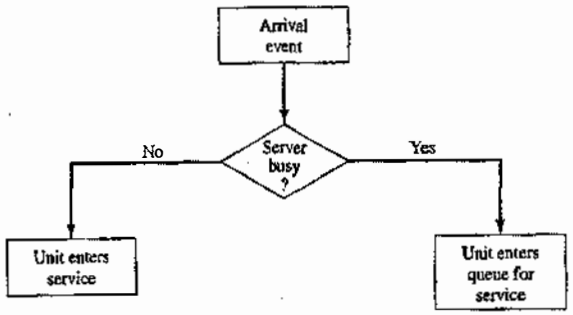


Figure 2.3 *Unit entering system* flow diagram.

		Queue status	
		Not empty	Empty
Server status	Busy	Enter queue	Enter queue
	Idle	Impossible	Enter service

Figure 2.4 Potential unit actions upon arrival.

		Queue status	
		Not empty	Empty
Server outcomes	Busy		Impossible
	Idle	Impossible	

Figure 2.5 Server outcomes after the completion of service.

after a service is completed. These two possibilities are shown as the shaded portions of Figure 2.5. It is impossible for the server to become busy if the queue is empty when a service is completed. Similarly, it is impossible for the server to be idle after a service is completed when the queue is not empty.

Now, how can the events described above occur in simulated time? Simulations of queueing systems generally require the maintenance of an event list for determining what happens next. The event list tracks the future times at which the different types of events occur. Simulations using event lists are described in Chapter 3. This chapter simplifies the simulation by tracking each unit explicitly. Simulation clock times for arrivals and departures are computed in a simulation table customized for each problem. In simulation, events usually occur at random times, the randomness imitating uncertainty in real life. For example, it is not known with certainty when the next customer will arrive at a grocery checkout counter, or how long the bank teller will take to complete a transaction. In these cases, a statistical model of the data is developed either from data collected and analyzed or from subjective estimates and assumptions.

The randomness needed to imitate real life is made possible through the use of "random numbers." Random numbers are distributed uniformly and independently on the interval (0, 1). Random digits are uniformly distributed on the set {0, 1, 2,..., 9}. Random digits can be used to form random numbers by selecting the proper number of digits for each random number and placing a decimal point to the left of the value selected. The proper number of digits is dictated by the accuracy of the data being used for input purposes. If the input distribution has values with two decimal places, two digits are taken from a random digits table (such as Table A.1) and the decimal point is placed to the left to form a random number.

Random numbers also can be generated in simulation packages and in spreadsheets (such as Excel). For example, Excel has a macro function called RAND() that returns a "random" number between 0 and 1. When numbers are generated by using a procedure, they are often referred to as pseudo-random numbers. Because the procedure is fully known, it is always possible to predict the sequence of numbers that will be generated prior to the simulation. The most commonly used methods for generating random numbers are discussed in Chapter 7.

In a single-channel queueing simulation, interarrival times and service times are generated from the distributions of these random variables. The examples that follow show how such times are generated. For simplicity, assume that the times between arrivals were generated by rolling a die five times and recording the up face. Table 2.2 contains a set of five interarrival times generated in this manner. These five interarrival times are used to compute the arrival times of six customers at the queueing system.

Table 2.2 Interarrival and Clock Times

Customer	*Interarrival Time*	*Arrival Time on Clock*
1	—	0
2	2	2
3	4	6
4	1	7
5	2	9
6	6	15

The first customer is assumed to arrive at clock time 0. This starts the clock in operation. The second customer arrives two time units later, at clock time 2. The third customer arrives four time units later, at clock time 6; and so on.

The second time of interest is the service time. Table 2.3 contains service times generated at random from a distribution of service times. The only possible service times are one, two, three, and four time units. Assuming that all four values are equally likely to occur, these values could have been generated by placing the numbers one through four on chips and drawing the chips from a hat with replacement, being sure to record the numbers selected. Now, the interarrival times and service times must be meshed to simulate the single-channel queueing system. As is shown in Table 2.4, the first customer arrives at clock time 0 and immediately begins service, which requires two minutes. Service is completed at clock time 2. The second customer arrives at clock time 2 and is finished at clock time 3. Note that the fourth customer arrived at clock time 7, but service could not begin until clock time 9. This occurred because customer 3 did not finish service until clock time 9.

Table 2.4 was designed specifically for a single-channel queue that serves customers on a first-in–first-out (FIFO) basis. It keeps track of the clock time at which each event occurs. The second column of Table 2.4 records the clock time of each arrival event, while the last column records the clock time of each departure event. The occurrence of the two types of events in chronological order is shown in Table 2.5 and Figure 2.6.

It should be noted that Table 2.5 is ordered by clock time, in which case the events may or may not be ordered by customer number. The chronological ordering of events is the basis of the approach to discrete-event simulation described in Chapter 3.

Figure 2.6 depicts the number of customers in the system at the various clock times. It is a visual image of the event listing of Table 2.5. Customer 1 is in the system from clock time 0 to clock time 2. Customer 2 arrives at clock time 2 and departs at clock time 3. No customers are in the system from clock time 3 to clock

Table 2.3 Service Times

Customer	*Service Time*
1	2
2	1
3	3
4	2
5	1
6	4

Table 2.4 Simulation Table Emphasizing Clock Times

A *Customer Number*	B *Arrival Time (Clock)*	C *Time Service Begins (Clock)*	D *Service Time (Duration)*	E *Time Service Ends (Clock)*
1	0	0	2	2
2	2	2	1	3
3	6	6	3	9
4	7	9	2	11
5	9	11	1	12
6	15	15	4	19

Table 2.5 Chronological Ordering of Events

Event Type	*Customer Number*	*Clock Time*
Arrival	1	0
Departure	1	2
Arrival	2	2
Departure	2	3
Arrival	3	6
Arrival	4	7
Departure	3	9
Arrival	5	9
Departure	4	11
Departure	5	12
Arrival	6	15
Departure	6	19

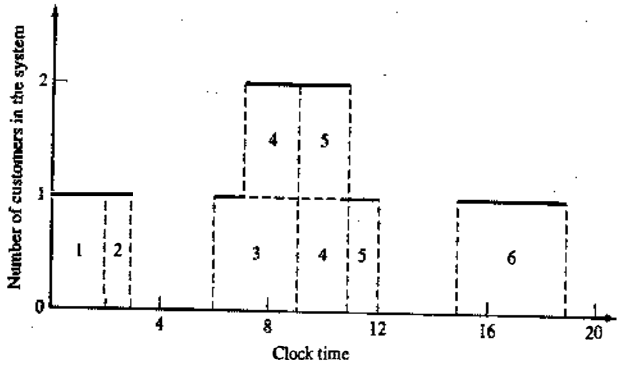


Figure 2.6 Number of customers in the system.

time 6. During some time periods, two customers are in the system, such as at clock time 8, when customers 3 and 4 are both in the system. Also, there are times when events occur simultaneously, such as at clock time 9, when customer 5 arrives and customer 3 departs.

Example 2.1 follows the logic described above while keeping track of a number of attributes of the system. Example 2.2 is concerned with a two-channel queueing system. The flow diagrams for a multichannel queueing system are slightly different from those for a single-channel system. The development and interpretation of these flow diagrams is left as an exercise for the reader.

Example 2.1: Single-Channel Queue

A small grocery store has only one checkout counter. Customers arrive at this checkout counter at random times that are from 1 to 8 minutes apart. Each possible value of interarrival time has the same probability of occurrence, as shown in Table 2.6. The service times vary from 1 to 6 minutes, with the probabilities shown in Table 2.7. The problem is to analyze the system by simulating the arrival and service of 100 customers.

In actuality, 100 customers is too small a sample size to draw any reliable conclusions. The accuracy of the results is enhanced by increasing the sample size, as is discussed in Chapter 11. However, the purpose of the exercise is to demonstrate how simple simulations can be carried out in a table, either manually or with a spreadsheet, not to recommend changes in the grocery store. A second issue, discussed thoroughly in Chapter 11, is that of initial conditions. A simulation of a grocery store that starts with an empty system is not realistic unless the intention is to model the system from startup or to model until steady-state operation is reached. Here, to keep calculations simple, starting conditions and concerns are overlooked.

Table 2.6 Distribution of Time Between Arrivals

Time between Arrivals (Minutes)	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
1	0.125	0.125	001–125
2	0.125	0.250	126–250
3	0.125	0.375	251–375
4	0.125	0.500	376–500
5	0.125	0.625	501–625
6	0.125	0.750	626–750
7	0.125	0.875	751–875
8	0.125	1.000	876–000

Table 2.7 Service-Time Distribution

Service Time (Minutes)	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
1	0.10	0.10	01–10
2	0.20	0.30	11–30
3	0.30	0.60	31–60
4	0.25	0.85	61–85
5	0.10	0.95	86–95
6	0.05	1.00	96–00

A set of uniformly distributed random numbers is needed to generate the arrivals at the checkout counter. Such random numbers have the following properties:

1. The set of random numbers is uniformly distributed between 0 and 1.
2. Successive random numbers are independent.

With tabular simulations, random digits such as those found in Table A.1 in the Appendix can be converted to random numbers. Random digits are converted to random numbers by placing a decimal point appropriately. Since the probabilities in Table 2.6 are accurate to 3 significant digits, three-place random numbers will suffice. It is necessary to list 99 random numbers to generate the times between arrivals. Why only 99 numbers? The first arrival is assumed to occur at time 0, so only 99 more arrivals need to be generated to end up with 100 customers. Similarly, for Table 2.7, two-place random numbers will suffice.

The rightmost two columns of Tables 2.6 and 2.7 are used to generate random arrivals and random service times. The third column in each table contains the cumulative probability for the distribution. The rightmost column contains the random digit assignment. In Table 2.6, the first random digit assignment is 001–125. There are 1000 three-digit values possible (001 through 000). The probability of a time-between-arrival of 1 minute is 0.125, so 125 of the 1000 random digit values are assigned to such an occurrence. Times between arrival for 99 customers are generated by listing 99 three-digit values from Table A.1 and comparing them to the random digit assignment of Table 2.6.

For manual simulations, it is good practice to start at a random position in the random digit table and proceed in a systematic direction, never re-using the same stream of digits in a given problem. If the same pattern is used repeatedly, bias could result from the same pattern's being generated.

The time-between-arrival determination is shown in Table 2.8. Note that the first random digits are 064. To obtain the corresponding time between arrivals, enter the fourth column of Table 2.6 and read 1 minute from the first column of the table. Alternatively, we see that 0.064 is between the cumulative probabilities 0.001 and 0.125, again resulting in 1 minute as the generated time.

Service times for the first 18 and the 100th customers are shown in Table 2.9. These service times were generated via the methodology described above, together with the aid of Table 2.7. (The entire table can be generated by using the Excel spreadsheet for Example 2.1 at www.bcnn.net.) The first customer's service time is 4 minutes, because the random digits 84 fall in the bracket 61–85—or, alternatively, because the derived random number 0.84 falls between the cumulative probabilities 0.61 and 0.85.

Table 2.8 Time-Between-Arrival Determination

Customer	*Random Digits*	*Time between Arrivals (Minutes)*	*Customer*	*Random Digits*	*Time between Arrivals (Minutes)*
1	—	—	11	413	4
2	064	1	12	462	4
3	112	1	13	843	7
4	678	6	14	738	6
5	289	3	15	359	3
6	871	7	16	888	8
7	583	5	17	902	8
8	139	2	18	212	2
9	423	4	⋮	⋮	⋮
10	039	1	100	538	5

Table 2.9 Service Times Generated

Customer	*Random Digits*	*Service Time (Minutes)*	*Customer*	*Random Digits*	*Service Time (Minutes)*
1	84	4	11	94	5
2	18	2	12	32	3
3	87	5	13	79	4
4	81	4	14	92	5
5	06	1	15	46	3
6	91	5	16	21	2
7	79	4	17	73	4
8	09	1	18	55	3
9	64	4	⋮	⋮	⋮
10	38	3	100	26	2

The essence of a manual simulation is the simulation table. These tables are designed for the problem at hand, with columns added to answer the questions posed. The simulation table for the single-channel queue, shown in Table 2.10, is an extension of the type of table already seen in Table 2.4. The first step is to initialize the table by filling in cells for the first customer. The first customer is assumed to arrive at time 0. Service begins immediately and finishes at time 4. The customer was in the system for 4 minutes. After the first customer, subsequent rows in the table are based on the random numbers for interarrival time, service time, and the completion time of the previous customer. For example, the second customer arrives at time 1. But service could not begin until time 4; the server (checkout person) was busy until that time. The second customer waited in the queue for three minutes. The second customer was in the system for 5 minutes. Skip down to the fifth customer. Service ends at time 16, but the sixth customer does not arrive until time 18, at which time service began. The server (checkout person) was idle for two minutes. This process continues for all 100 customers. The rightmost two columns have been added to collect statistical measures of performance, such as each customer's time in system and the server's idle time (if any) since the previous customer departed. In order to compute summary statistics, totals are formed as shown for service times, time customers spend in the system, idle time of the server, and time the customers wait in the queue.

Some of the findings from the simulation in Table 2.10 are as follows:

1. The average waiting time for a customer is 1.74 minutes. This is computed in the following manner:

$$\text{Average waiting time (minutes)} = \frac{\text{total time customers wait in queue (minutes)}}{\text{total numbers of customers}}$$

$$= \frac{174}{100} = 1.74 \text{ minutes}$$

2. The probability that a customer has to wait in the queue is 0.46. This is computed in the following manner:

$$\text{probability(wait)} = \frac{\text{numbers of customers who wait}}{\text{total number of customers}}$$

$$= \frac{46}{100} = 0.46$$

Table 2.10 Simulation Table for Single-Channel Queueing Problem

		Clock		Clock			Clock		
Customer	Interarrival Time (Minutes)	Arrival Time	Service Time (Minutes)	Time Service Begins	Waiting Time in Queue (Minutes)	Time Service Ends	Time Customer Spends in System (Minutes)	Idle Time of Server (Minutes)
1		0	4	0	0	4	4	
2	1	1	2	4	3	6	5	0
3	1	2	5	6	4	11	9	0
4	6	8	4	11	3	15	7	0
5	3	11	1	15	4	16	5	0
6	7	18	5	18	0	23	5	2
7	5	23	4	23	0	27	4	0
8	2	25	1	27	2	28	3	0
9	4	29	4	29	0	33	4	1
10	1	30	3	33	3	36	6	0
11	4	34	5	36	2	41	7	0
12	4	38	3	41	3	44	6	0
13	7	45	4	45	0	49	4	1
14	6	51	5	51	0	56	5	2
15	3	54	3	56	2	59	5	0
16	8	62	2	62	0	64	2	3
17	8	70	4	70	0	74	4	6
18	2	72	3	74	2	77	5	0
19	7	79	1	79	0	80	1	2
20	4	83	2	83	0	85	2	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	5	415	2	416	1	418	3	0
Total	415		317		174		491	101

3. The proportion of idle time of the server is 0.24. This is computed in the following manner:

$$\text{probability of idle server} = \frac{\text{total idle time of server (minutes)}}{\text{total run time of simulation (minutes)}} = \frac{101}{418} = 0.24$$

The probability of the server's being busy is the complement of 0.24, namely, 0.76.

4. The average service time is 3.17 minutes. This is computed in the following manner:

$$\text{Average service time (minutes)} = \frac{\text{total service time (minutes)}}{\text{total number of customers}} = \frac{317}{100} = 3.17 \text{ minutes}$$

This result can be compared with the expected service time by finding the mean of the service-time distribution, using the equation

$$E(S) = \sum_{s=0}^{\infty} sp(s)$$

Applying the expected-value equation to the distribution in Table 2.7 gives

Expected service time =
$1(0.10) + 2(0.20) + 3(0.30) + 4(0.25) + 5(0.10) + 6(0.05) = 3.2$ minutes

The expected service time is slightly higher than the average service time in the simulation. The longer the simulation, the closer the average will be to $E(S)$.

5. The average time between arrivals is 4.19 minutes. This is computed in the following manner:

$$\text{Average time between arrivals (minutes)} = \frac{\text{sum of all times between arrival (minutes)}}{\text{number of arrivals} - 1} = \frac{415}{99} = 4.19 \text{ minutes}$$

One is subtracted from the denominator because the first arrival is assumed to occur at time 0. This result can be compared to the expected time between arrivals by finding the mean of the discrete uniform distribution whose endpoints are $a = 1$ and $b = 8$. The mean is given by

$$E(A) = \frac{a+b}{2} = \frac{1+8}{2} = 4.5 \text{ minutes}$$

The expected time between arrivals is slightly higher than the average. However, as the simulation becomes longer, the average value of the time between arrivals should approach the theoretical mean, $E(A)$.

6. The average waiting time of those who wait is 3.22 minutes. This is computed in the following manner:

$$\text{Average waiting time of those who wait (minutes)} = \frac{\text{total time customers wait in queue (minutes)}}{\text{total number of customers that wait}}$$

$$= \frac{174}{54} = 3.22 \text{ minutes}$$

7. The average time a customer spends in the system is 4.91 minutes. This can be found in two ways. First, the computation can be achieved by the following relationship:

$$\text{Average time customer spends in the system (minutes)} = \frac{\text{total time customers spend in the system (minutes)}}{\text{total number of customers}}$$

$$= \frac{491}{100} = 4.91 \text{ minutes}$$

The second way of computing this same result is to realize that the following relationship must hold:

$$\text{Average time customer spends in the system (minutes)} = \text{average time customer spends waiting in the queue (minutes)} + \text{average time customer spends in service (minutes)}$$

From findings 1 and 4, this results in

$$\text{Average time customer spends in the system} = 1.74 + 3.17 = 4.91 \text{ minutes}$$

A decision maker would be interested in results of this type, but a longer simulation would increase the accuracy of the findings. However, some tentative inferences can be drawn at this point. About half of the customers have to wait; however, the average waiting time is not excessive. The server does not have an undue amount of idle time. More reliable statements about the results would depend on balancing the cost of waiting against the cost of additional servers.

Excel spreadsheets have been constructed for each of the examples in this chapter. The spreadsheets can be found at www.bcnn.net. The spreadsheets have a common format. The first sheet is One-Trial. The second sheet is Experiment. The third sheet is entitled Explain. Here, the logic in the spreadsheet is discussed, and questions pertaining to that logic are asked of the reader. Use the default seed '12345' to reproduce the One-Trial output shown in the examples in the text, and use the appropriate number of trials (or replications) to reproduce the Experiment shown in the text, again using the default seed '12345'.

Exercises relating to the spreadsheets have been prepared also. These are the last set of exercises at the end of this chapter. The first set of exercises is for manual simulation.

The spreadsheets allow for many entities to flow through the system. (In Example 2.1, the entities are customers.) For instance, the spreadsheet for Example 2.1 has 100 customers going through the system, and the number of trials can vary from one to 400. Let's say that 200 trials are selected. Then, 200 trials of the simulation, each of 100 customers, will be conducted.

For Example 2.1, the frequency of waiting time in queue for the first trial of 100 customers is shown in Figure 2.7. (Note: In all histograms in the remainder of this chapter, the upper limit of the bin is indicated on the legend on the x-axis, even if the legend is shown centered within the bin.) As mentioned previously, 46% of them did not have to wait, and 42% waited less than four minutes (but more than zero minutes).

From the Experiment sheet of the Excel spreadsheet, the average waiting time over 50 trials was 1.50 minutes. Figure 2.8 shows a histogram of the 50 average waiting times for the 50 trials. The overall average (1.50 minutes) is just to the right of the two most popular bins.

The exercises ask that you experiment with this spreadsheet. But, you can also experiment on your own to discover the effect of randomness and of the input data. For example, what if you run 400 trials instead of 50?

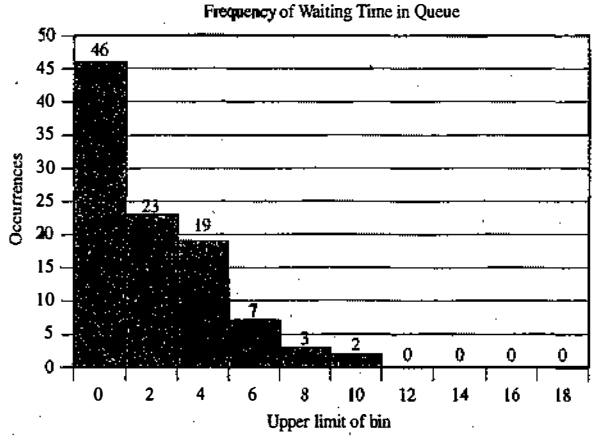


Figure 2.7 Frequency of waiting time in queue.

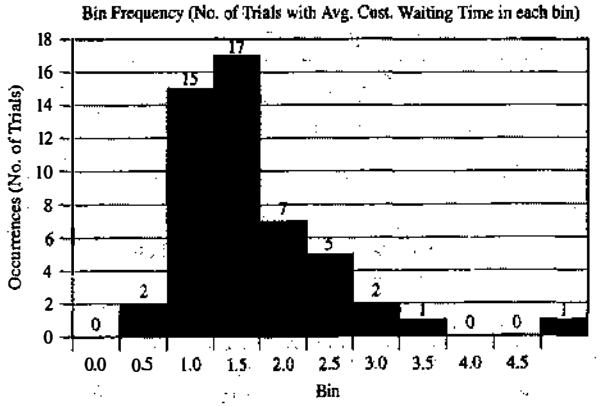


Figure 2.8 Frequency distribution of average waiting times.

Does the shape of the distribution in Figure 2.8 change? What if you run 25 trials instead of 50? How much does the shape change as you generate new trials?

Example 2.2: The Able–Baker Call Center Problem

This example illustrates the simulation procedure when there is more than one service channel. Consider a computer technical support center where personnel take calls and provide service. The time between calls ranges from 1 to 4 minutes, with distribution as shown in Table 2.11. There are two technical support people—Able and Baker. Able is more experienced and can provide service faster than Baker. The distributions of their service times are shown in Tables 2.12 and 2.13. Times are usually a continuous measure. But, in this and other time-based examples in this chapter, we make them discrete for ease of explanation.

The simulation proceeds in a manner similar to Example 2.1, except that it is more complex because of the two servers. A simplifying rule is that Able gets the call if both technical support people are idle. Able has seniority. (The solution would be different if the decision were made at random or by any other rule.)

The problem is to find how well the current arrangement is working. To estimate the system measures of performance, a simulation of the first 100 callers is made. A simulation with more callers would yield more reliable results, but, for purposes of this illustration, a 100-caller simulation has been selected.

Table 2.11 Interarrival Distribution of Calls for Technical Support

Time between Arrivals (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
1	0.25	0.25	01–25
2	0.40	0.65	26–65
3	0.20	0.85	66–85
4	0.15	1.00	86–00

Table 2.12 Service Distribution of Able

Service Time (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
2	0.30	0.30	01–30
3	0.28	0.58	31–58
4	0.25	0.83	59–83
5	0.17	1.00	84–00

Table 2.13 Service Distribution of Baker

Service Time (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
3	0.35	0.35	01–35
4	0.25	0.60	36–60
5	0.20	0.80	61–80
6	0.20	1.00	81–00

The simulation proceeds in accordance with the following set of steps:

Step 1. For Caller k, generate an interarrival time A_k. Add it to the previous arrival time T_{k-1} to get the arrival time of Caller k as $T_k = T_{k-1} + A_k$.

Step 2. If Able is idle, Caller k begins service with Able at the current time T_{now}.

Able's service completion time, $T_{fin,A}$ is given by $T_{fin,A} = T_{now} + T_{svc,A}$ where $T_{svc,A}$ is the service time generated from Able's Service Time Distribution.

Caller k's time in system, T_{sys}, is given by $T_{sys} = T_{fin,A} - T_k$.

Because Able was idle, Caller k's delay, T_{wait}, is given by $T_{wait} = 0$.

If Able is busy, but Baker is idle, Caller k begins service with Baker at the current time T_{now}. Baker's service completion time, $T_{fin,B}$ is given by $T_{fin,B} = T_{now} + T_{svc,B}$ where $T_{svc,B}$ is the service time generated from Baker's Service Time Distribution.

Caller k's time in system, T_{sys}, is given by $T_{sys} = T_{fin,B} - T_k$.

Because Baker was idle, Caller k's delay, T_{wait}, is given by $T_{wait} = 0$.

Step 3. If Able and Baker are both busy, then calculate the time at which the first one becomes available, as follows:

$$T_{beg} = min(T_{fin,A}, T_{fin,B}).$$

Caller k begins service at T_{beg}. When service for Caller k begins, set $T_{now} = T_{beg}$.

Then compute $T_{fin,A}$ or $T_{fin,B}$ as in Step 2.

Caller k's time in system, T_{sys}, is given by $T_{sys} = T_{fin,A} - T_k$ or $T_{sys} = T_{fin,B} - T_k$, as appropriate.

The preceding steps have been implemented in an Excel spreadsheet that is available on the website www.bcnn.net. The reader is strongly encouraged to examine the Excel spreadsheet with particular emphasis on how the cell values are calculated. Also, there are exercises at the end of this chapter that ask the reader to run a variety of experiments using the spreadsheet.

A portion of the output in the Excel spreadsheet is given in Table 2.14 to clarify the steps previously listed. Caller 1 arrives at clock time 0 to get the simulation started. Able is idle, so Caller 1 begins service with Able at clock time 0. The service time, 2 minutes, is generated from information given in Table 2.12 by following the procedure in Example 2.1. Thus, Caller 1 completes service at clock time 2 minutes and was not delayed.

An interarrival time of 2 minutes is generated from Table 2.11 by following the procedure in Example 2.1. So, the arrival of Caller 2 is at clock time 2 minutes. Able is idle at the time, having just completed service on Caller 1, so Caller 2 is served by Able.

Now, skip down to Caller 4, serviced by Able from clock time 8 minutes to clock time 12 minutes. Note that Caller 5 arrives at clock time 9 minutes. Because Able is busy with Caller 4 at that time, but Baker is available, Baker services Caller 5, completing service at clock time 12 minutes.

For the trial under discussion (the simulation of 100 callers), the frequency diagram shown in Figure 2.9 (from the Excel spreadsheet in www.bcnn.net) shows that 62 of 100 (62%) of the callers had no delay, 12% had a delay of one or two minutes, and so on.

The distribution in Figure 2.9 is skewed to the right. Push 'Generate New Trial' repeatedly and notice that this is usually what happens, but not always. 'Generate New Trial' recalculates the spreadsheet for one trial (100 callers). One trial does not provide sufficient information on which to form a conclusion, but this is one of the great advantages to having the spreadsheet—the effect of variability is quite evident.

Table 2.14 Simulation Table for Call-Center Example

Caller Number	Interarrival Time (Minutes)	Clock: Arrival Time	Clock: When Able Available	Clock: When Baker Available	Server Chosen	Service Time (Minutes)	Clock: Time Service Begins	Clock: Able's Service Completion Time	Clock: Baker's Service Completion Time	Caller Delay (Minutes)	Time in System (Minutes)
1		0	0	0	Able	2	0	2		0	2
2	2	2	2	0	Able	2	2	4		0	2
3	4	6	4	0	Able	2	6	8		0	2
4	2	8	8	0	Able	4	8	12		0	4
5	1	9	12	0	Baker	3	9		12	0	3
...	...	...	...	...	...	...	...	...	...	...	...
100	1	219	221	219	Baker	4	219		223	0	4
Total										211	564

Figure 2.9 Frequency of caller delay for first trial.

Figure 2.10 Frequency of caller delay for experiment of 400 trials.

In Table 2.14, it is seen that the total customer delay was 211 minutes, or about 2.1 minutes per caller. It is also seen in Table 2.14 that the total time in system was 564 minutes, or about 5.6 minutes per caller.

In the second sheet of the spreadsheet, we run an experiment with 400 trials (each trial consisting of the simulation of 100 callers) to generate Figure 2.10. It is seen that about 19% of the average delays (78 of 400) are longer than one minute. Only 2.75% (11 of 400) are longer than 2 minutes.

In summary, one server cannot handle all the callers, and three servers would probably be more than are necessary. Adding an additional server would surely reduce the waiting time to nearly zero; however, the cost of waiting would have to be quite high to justify an additional server.

2.2 SIMULATION OF INVENTORY SYSTEMS

An important class of simulation problems involves inventory systems. A simple inventory system is shown in Figure 2.11. This inventory system has a periodic review of length N, at which time the inventory level is checked. An order is made to bring the inventory up to the level M. At the end of the first review period, an order quantity, Q_1, is placed. In this inventory system, the lead time (i.e., the length of time between the placement and receipt of an order) is zero. Demands are not usually known with certainty, so the order

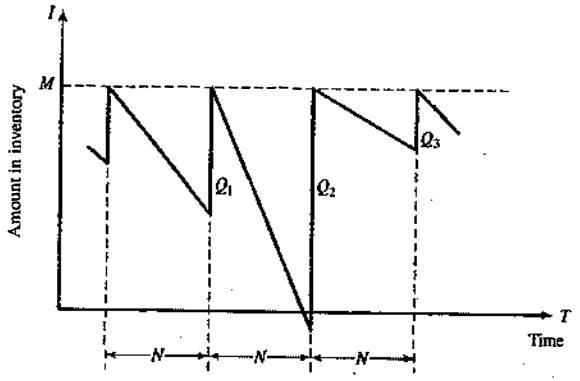


Figure 2.11 Probabilistic order-level inventory system.

quantities are probabilistic. Demand is shown as being uniform over the time period in Figure 2.11. In actuality, demands are not usually uniform and do fluctuate over time. One possibility is that demands all occur at the beginning of the cycle. Another is that the lead time is random of some positive length.

Notice that, in the second cycle, the amount in inventory drops below zero, indicating a shortage. In Figure 2.11, these units are backordered; when the order arrives, the demand for the backordered items is satisfied first. To avoid shortages, a buffer, or safety, stock would need to be carried.

Carrying stock in inventory has an associated cost attributed to the interest paid on the funds borrowed to buy the items (this also could be considered as the loss from not having the funds available for other investment purposes). Other costs can be placed in the carrying or holding cost column: renting of storage space, hiring of guards, and so on. An alternative to carrying high inventory is to make more frequent reviews and, consequently, more frequent purchases or replenishments. This has an associated cost: the ordering cost. Also, there is a cost in being short. Customers could get angry, with a subsequent loss of good will. Larger inventories decrease the possibilities of shortages. These costs must be traded off in order to minimize the total cost of an inventory system.

The total cost (or total profit) of an inventory system is the measure of performance. This can be affected by the policy alternatives. For example, in Figure 2.11, the decision maker can control the maximum inventory level, M, and the length of the cycle, N. What effect does changing N have on the various costs?

In an (M, N) inventory system, the events that may occur are the demand for items in the inventory, the review of the inventory position, and the receipt of an order at the end of each review period. When the lead time is zero, as in Figure 2.11, the last two events occur simultaneously.

In the following example for deciding how many newspapers to buy, only a single time period of specified length is relevant, and only a single procurement is made. Inventory remaining at the end of the single time period is sold for scrap or discarded. A wide variety of real-world problems are of this form, including the stocking of spare parts, perishable items, style goods, and special seasonal items [Hadley and Whitin, 1963].

Example 2.3: The News Dealer's Problem

A classical inventory problem concerns the purchase and sale of newspapers. The newsstand buys the papers for 33 cents each and sells them for 50 cents each. Newspapers not sold at the end of the day are sold as scrap for 5 cents each. Newspapers can be purchased in bundles of 10. Thus, the newsstand can buy 50, 60, and so on. There are three types of newsdays: "good"; "fair"; and "poor"; they have the probabilities 0.35, 0.45, and 0.20, respectively. The distribution of newspapers demanded on each of these days is given in Table 2.15. The problem is to compute the optimal number of papers the newsstand should purchase. This will be accomplished by simulating demands for 20 days and recording profits from sales each day.

Table 2.15 Distribution of Newspapers Demanded Per Day

	Demand Probability Distribution		
Demand	*Good*	*Fair*	*Poor*
40	0.03	0.10	0.44
50	0.05	0.18	0.22
60	0.15	0.40	0.16
70	0.20	0.20	0.12
80	0.35	0.08	0.06
90	0.15	0.04	0.00
100	0.07	0.00	0.00

The profits are given by the following relationship:

$$\text{profit} = \begin{pmatrix}\text{revenue}\\ \text{from sales}\end{pmatrix} - \begin{pmatrix}\text{cost of}\\ \text{newspapers}\end{pmatrix} - \begin{pmatrix}\text{lost profit from}\\ \text{excess demand}\end{pmatrix} + \begin{pmatrix}\text{salvage from sale}\\ \text{of scrap papers}\end{pmatrix}$$

From the problem statement, the revenue from sales is 50 cents for each paper sold. The cost of newspapers is 33 cents for each paper purchased. The lost profit from excess demand is 17 cents for each paper demanded that could not be provided. Such a shortage cost is somewhat controversial, but makes the problem much more interesting. The salvage value of scrap papers is 5 cents each.

Tables 2.16 and 2.17 provide the random digit assignments for the types of newsdays and the demands for those newsdays. To solve this problem by simulation requires setting a policy of buying a certain number of papers each day, then simulating the demands for papers over the 20-day time period to determine the total profit. The policy (number of newspapers purchased) is changed to other values and the simulation repeated until the best value is found.

The simulation table for the decision to purchase 70 newspapers is shown in Table 2.18.

On day 1, the demand is for 80 newspapers, but only 70 newspapers are available. The revenue from the sale of 70 newspapers is \$35.00. The lost profit for the excess demand of 10 newspapers is \$1.70. The profit for the first day is computed as follows:

$$\text{Profit} = \$35.00 - \$23.10 - \$1.70 + 0 = \$10.20$$

On the fourth day, the demand is less than the supply. The revenue from sales of 50 newspapers is \$25.00. Twenty newspapers are sold for scrap at \$0.05 each yielding \$1.00. The daily profit is determined as follows:

$$\text{Profit} = \$25.00 - \$23.10 - 0 + \$1.00 = \$2.90$$

Table 2.16 Random Digit Assignment for Type of Newsday

Type of Newsday	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
Good	0.35	0.35	01–35
Fair	0.45	0.80	36–80
Poor	0.20	1.00	81–00

Table 2.17 Random Digit Assignments for Newspapers Demanded

	Cumulative Distribution			*Random Digit Assignment*		
Demand	*Good*	*Fair*	*Poor*	*Good*	*Fair*	*Poor*
40	0.03	0.10	0.44	01–03	01–10	01–44
50	0.08	0.28	0.66	04–08	11–28	45–66
60	0.23	0.68	0.82	09–23	29–68	67–82
70	0.43	0.88	0.94	24–43	69–88	83–94
80	0.78	0.96	1.00	44–78	89–96	95–00
90	0.93	1.00	1.00	79–93	97–00	
100	1.00	1.00	1.00	94–00		

Table 2.18 Simulation Table for Purchase of 70 Newspapers

Day	*Random Digits for Type of Newsday*	*Type of Newsday*	*Random Digits for Demand*	*Demand*	*Revenue from Sales*	*Lost Profit from Excess Demand*	*Salvage from Sale of Scrap*	*Daily Profit*
1	58	Fair	93	80	$35.00	$1.70	–	$10.20
2	17	Good	63	80	35.00	1.70	–	10.20
3	21	Good	31	70	35.00	–	–	11.90
4	45	Fair	19	50	25.00	–	1.00	2.90
5	43	Fair	91	80	35.00	1.70	–	10.20
6	36	Fair	75	70	35.00	–	–	11.90
7	27	Good	84	90	35.00	3.40	–	8.50
8	73	Fair	37	60	30.00	–	0.50	7.40
9	86	Poor	23	40	20.00	–	1.50	−1.60
10	19	Good	02	40	20.00	–	1.50	−1.60
11	93	Poor	53	50	25.00	–	1.00	2.90
12	45	Fair	96	80	35.00	1.70	–	10.20
13	47	Fair	33	60	30.00	–	0.50	7.40
14	30	Good	86	90	35.00	3.40	–	8.50
15	12	Good	16	60	30.00	–	0.50	7.40
16	41	Fair	07	40	20.00	–	1.50	−1.60
17	65	Fair	64	60	30.00	–	0.50	7.40
18	57	Fair	94	80	35.00	1.70	–	10.20
19	18	Good	55	80	35.00	1.70	–	10.20
20	98	Poor	13	40	20.00	–	1.50	−1.60
					$600.00	$17.00	$10.00	$131.00

The profit for the 20-day period is the sum of the daily profits, $131.00. It can also be computed from the totals for the 20 days of the simulation as follows:

$$\text{Total profit} = \$600.00 - \$462.00 + \$17.00 - \$10.00 = \$131.00$$

where the cost of newspapers for 20 days is $(20 \times \$0.33 \times 70) = \$462.00$. In general, because the results of one day are independent of previous days, inventory problems of this type are easier than queueing problems when solved in a spreadsheet such as is shown in www.bcnn.net and discussed shortly.

Figure 2.12 shows the result of 400 trials, each of twenty days, with a policy of purchasing 70 newspapers per day. For these trials, the average total (20-day) profit was $137.61. The minimum 20-day profit was

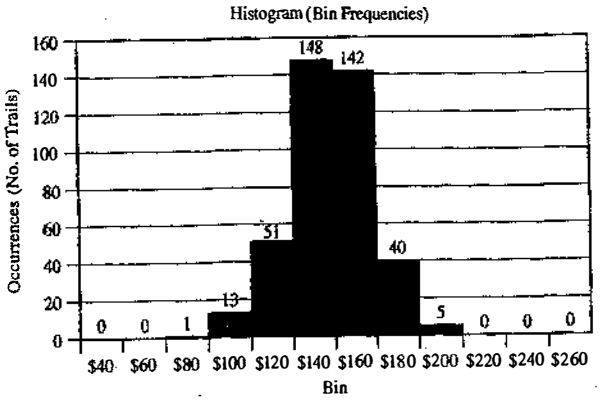


Figure 2.12 Frequency of total (20-day) profits with purchasing of 70 newspapers per day.

$64.70 and the maximum was $186.10. Figure 1.12 shows that only 45 of the 400 trials resulted in a total 20-day profit of more than $160.

The manual solution shown in Table 2.18 had a profit of $131.00. This one 20-day is not far from the average over the 400 trials, $137.61; but the result for one 20-day simulation could have been the minimum value or the maximum value. Such an occurrence demonstrates the usefulness of conducting many trials.

On the One Trial sheet, look at the Daily Profit that results when clicking the button 'Generate New Trial'. The results vary quite a bit both in the histogram called 'Frequency of Daily Profit' (showing what happened on each of the 20 days) and in the total profits for those 20 days. The histograms are almost like snowflakes, in that no two are alike! The first two histograms generated are shown in Figure 2.13.

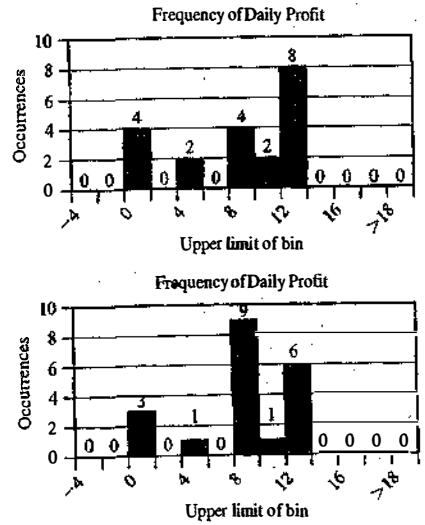


Figure 2.13 First two histograms of daily profit.

Example 2.4: Simulation of an Order-Up-To-Level Inventory System

Consider a situation in which a company sells refrigerators. The system they use for maintaining inventory is to review the situation after a fixed number of days (say N) and make a decision about what is to be done. The policy is to order up to a level (the *order up to* level—say, M), using the following relationship:

Order quantity = (Order-up-to level) − (Ending inventory) + (Shortage quantity)

Let's say that the order-up-to level (M) is 11 and the ending inventory is three. Further, let's say that the review period (N) is five days. Thus, on the fifth day of the cycle, 8 refrigerators will be ordered from the supplier. If there is a shortage of two refrigerators on the fifth day, then 13 refrigerators will be ordered. (There can't be both ending inventory and shortages at the same time.) If there were a shortage of three refrigerators, the first three received would be provided to the customers when the order arrived. That's called "making up backorders." The *lost sales* case occurs when customer demand is lost if the inventory is not available.

The number of refrigerators ordered each day is randomly distributed as shown in Table 2.19. Another source of randomness is the number of days after the order is placed with the supplier before arrival, or *lead time*. The distribution of lead time is shown in Table 2.20. Assume that the orders are placed at the end of the day. If the lead time is zero, the order from the supplier will arrive the next morning, and the refrigerators will be available for distribution that next day. If the lead time is one day, the order from the supplier arrives the second morning after, and will be available for distribution that day.

The simulation has been started with the inventory level at 3 refrigerators and an order for 8 refrigerators to arrive in 2 days' time. The simulation table is shown in Table 2.21.

Following the simulation table for several selected days indicates how the process operates. The order for 8 refrigerators is available on the morning of the third day of the first cycle, raising the inventory level from zero refrigerators to 8 refrigerators. Demands during the remainder of the first cycle reduced the ending inventory level to 2 refrigerators on the fifth day. Thus, an order for 9 refrigerators was placed. The lead time for this order was 2 days. The order for 9 refrigerators was added to inventory on the morning of day 3 of cycle 2.

Table 2.19 Random Digit Assignments for Daily Demand

Demand	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
0	0.10	0.10	01–10
1	0.25	0.35	11–35
2	0.35	0.70	36–70
3	0.21	0.91	71–91
4	0.09	1.00	92–00

Table 2.20 Random Digit Assignments for Lead Time

Lead Time (Days)	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
1	0.6	0.6	1–6
2	0.3	0.9	7–9
3	0.1	1.0	0

Table 2.21 Simulation Tables for {M, N} Inventory System

Day	*Cycle*	*Day within Cycle*	*Beginning Inventory*	*Random Digits for Demand*	*Demand*	*Ending Inventory*	*Shortage Quantity*	*Order Quantity*	*Random Digits for Demand*	*Lead Time (days)*	*Days until Order Arrives*
1	1	1	3	26	1	2	0	—	—	—	1
2	1	2	2	68	2	0	0	—	—	—	—
3	1	3	8	33	1	7	0	—	—	—	—
4	1	4	7	39	2	5	0	—	—	—	—
5	1	5	5	86	3	2	0	9	8	2	2
6	2	1	2	18	1	1	0	—	—	—	1
7	2	2	1	64	2	0	1	—	—	—	—
8	2	3	9	79	3	5	0	—	—	—	—
9	2	4	5	55	2	3	0	—	—	—	—
10	2	5	3	74	3	0	0	11	7	2	2
11	3	1	0	21	1	0	1	—	—	—	1
12	3	2	0	43	2	0	3	—	—	—	—
13	3	3	11	49	2	6	0	—	—	—	—
14	3	4	6	90	3	3	0	—	—	—	—
15	3	5	3	35	1	2	0	9	2	1	1
16	4	1	2	08	0	2	0	—	—	—	—
17	4	2	11	98	4	7	0	—	—	—	—
18	4	3	7	61	2	5	0	—	—	—	—
19	4	4	5	85	3	2	0	—	—	—	—
20	4	5	2	81	3	0	1	12	3	1	1
21	5	1	0	53	2	0	3	—	—	—	—
22	5	2	12	15	1	8	0	—	—	—	—
23	5	3	8	94	4	4	0	—	—	—	—
24	5	4	4	19	1	3	0	—	—	—	—
25	5	5	3	44	2	1	0	10	1	1	1
Total					51	68	9				
Average					2.04	2.72	0.36				

Notice that the beginning inventory on the fifth day of the fourth cycle was 2. An order for 3 refrigerators on that day led to a shortage condition. One refrigerator was backordered on that day. Twelve refrigerators were ordered (11 + 1), and they had a lead time of one day. On the next day, the demand was two, so additional shortages resulted.

At the beginning of the next day, the order had arrived. Three refrigerators were used to make up the backorders and there was a demand for one refrigerator, so the ending inventory was 8.

From five cycles of simulation, the average ending inventory is approximately 2.72 (68/25) units. On 5 of 25 days, a shortage condition existed.

In this example, there cannot be more than one order outstanding from the supplier at any time, but there are situations where lead times are so long that the relationship shown so far needs to be modified to the following:

Order quantity = (Order-up-to level) − (Ending inventory) − (On order) + (Shortage quantity)

This relationship makes sure that extra ordering doesn't occur. To make an estimate of the mean refrigerators in ending inventory by using simulation, many trials would have to be simulated. The Excel spreadsheet solution in www.bcnn.net offers an opportunity to perform such a simulation.

The Excel spreadsheet allows for numerous changes in the input. The policy can be changed (i.e., the values of M and N). The distribution of daily demand and lead time can be changed within the limits of the demand and lead time—that is, demand can be 0, 1, 2, 3, or 4 refrigerators per day and lead times can be 1, 2, or 3 days. Clicking on Generate New Trial will recalculate the spreadsheet. Looking at one figure after another in the One Trial sheet, with the values as given in the problem statement above shows that there is no consistent result.

However, setting the number of trials to 100 in the Experiment sheet and recalculating the spreadsheet produces little variability in the average inventory. It's usually in the range from 2.69 to 3.01, leaving the values the same as in the problem definition above. Nor is there much change in the distribution of the average ending inventory, as shown in Figure 2.14.

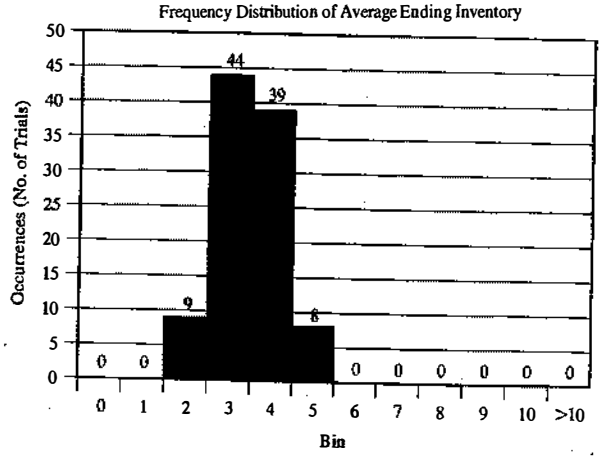


Figure 2.14 Average ending inventory for 100 trials (each 25 days).

2.3 OTHER EXAMPLES OF SIMULATION

This section includes examples of the simulation of a reliability problem, a bombing mission, the generation of the lead-time demand distribution when given the distributions of demand and lead time, and an activity network.

Example 2.5: A Reliability Problem

A milling machine has three different bearings that fail in service. The distribution of the life of each bearing is identical, as shown in Table 2.22. When a bearing fails, the mill stops, a repairperson is called, and a new bearing is installed. The delay time of the repairperson's arriving at the milling machine is also a random variable having the distribution given in Table 2.23. Downtime for the mill is estimated at $10 per minute. The direct on-site cost of the repairperson is $30 per hour. It takes 20 minutes to change one bearing, 30 minutes to change two bearings, and 40 minutes to change three bearings. A proposal has been made to replace all three bearings whenever a bearing fails. Management needs an evaluation of the proposal. The total cost per 10,000 bearing-hours will be used as the measure of performance.

Table 2.24 represents a simulation of 15 bearing changes under the current method of operation. Note that there are instances where more than one bearing fails at the same time. This is unlikely to occur in practice and is due to using a rather coarse grid of 100 hours for bearing life. It will be assumed in this example that the times are never exactly the same and thus no more than one bearing is changed at any breakdown. The cost of the current system is estimated as follows:

Cost of bearing = 45 bearings × $32/bearing = $1,440
Cost of delay time = (110 + 110 + 105) minutes × $10/minute = $3,250
Cost of downtime during repair = 45 bearings × 20 minutes/bearing × $10/minute = $9,000
Cost of repairpersons = 45 bearings × 20 minutes/bearing × $30/60 minutes = $450
Total cost = $1,440 + $3,250 +$9,000 + $450 = $14,140

The total life of the bearings is (22,300 + 18,700 + 18,600) = 59,600 hours. Therefore, the total cost per 10,000 bearing-hours is ($14,140/5.96) = $2,372.

Table 2.25 is a simulation of the proposed method. Note that the random digits are not shown. For the first set of bearings, the earliest failure is at 1,000 hours. All three bearings are replaced at that time, even though the remaining bearings had more life in them. For example, Bearing 1 would have lasted 700 additional hours.

Table 2.22 Bearing-Life Distribution

Bearing Life (Hours)	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
1000	0.10	0.10	01–10
1100	0.13	0.23	11–23
1200	0.25	0.48	24–48
1300	0.13	0.61	49–61
1400	0.09	0.70	62–70
1500	0.12	0.82	71–82
1600	0.02	0.84	83–84
1700	0.06	0.90	85–90
1800	0.05	0.95	91–95
1900	0.05	1.00	96–00

Table 2.23 Delay-Time Distribution

Delay Time (Minutes)	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
5	0.6	0.6	1–6
10	0.3	0.9	7–9
15	0.1	1.0	0

Table 2.24 Bearing Replacement under Current Method

	Bearing 1				*Bearing 2*				*Bearing 3*			
	RD[a]	*Life (Hours)*	*RD*	*Delay (Minutes)*	*RD*	*Life (Hours)*	*RD*	*Delay (Minutes)*	*RD*	*Life (Hours)*	*RD*	*Delay (Minutes)*
1	67	1,400	7	10	71	1,500	8	10	18	1,100	6	5
2	55	1,300	3	5	21	1,100	3	5	17	1,100	2	5
3	98	1,900	1	5	79	1,500	3	5	65	1,400	2	5
4	76	1,500	6	5	88	1,700	1	5	03	1,000	9	10
5	53	1,300	4	5	93	1,800	0	15	54	1,300	8	10
6	69	1,400	8	10	77	1,500	6	5	17	1,100	3	5
7	80	1,500	5	5	08	1,000	9	10	19	1,100	6	5
8	93	1,800	7	10	21	1,100	8	10	09	1,000	7	10
9	35	1,200	0	15	13	1,100	3	5	61	1,300	1	5
10	02	1,000	5	5	03	1,100	2	5	84	1,600	0	15
11	99	1,900	9	10	14	1,000	1	5	11	1,100	5	5
12	65	1,400	4	5	5	1,000	0	15	25	1,200	2	5
13	53	1,300	7	10	29	1,200	2	5	86	1,700	8	10
14	87	1,700	1	5	07	1,000	4	5	65	1,400	3	5
15	90	1,700	2	5	20	1,100	3	5	44	1,200	4	5
Total				110				110				105

[a]RD, random digits.

The cost of the proposed system is estimated as follows:

Cost of bearings = 45 bearings × \$32/bearing = \$1,440
Cost of delay time = 110 minutes × \$10/minute = \$1,100
Cost of downtime during repairs = 15 sets × 40 minutes/set × \$10/minute = \$6,000
Cost of repairpersons = 15 sets × 40 minutes/set × \$30/60 minutes = \$300
Total cost = \$1,440 + \$1,100 + \$6,000 + \$300 = \$8,840

The total life of the bearings is (17,000 × 3) = 51,000 hours. Therefore, the total cost per 10,000 bearing-hours is (\$8,840/5.1) = \$1,733.

The new policy generates a savings of \$634 per 10,000 hours of bearing-life. If the machine runs continuously, the savings are about \$556 per year.

There are two Excel spreadsheet models for Example 2.5 at www.bcnn.net. These are Example 2.5C (the current system) and Example 2.5P (the proposed system). Much flexibility is offered with respect to the inputs on these models. The user can change the distribution of bearing life (making sure that the cumulative

Table 2.25 Bearing Replacement under Proposed Method

	Bearing 1 Life (Hours)	*Bearing 2 Life (Hours)*	*Bearing 3 Life (Hours)*	*First Failure (Hours)*	*Delay (Minutes)*
1	1,700	1,100	1,000	1,000	10
2	1,000	1,800	1,200	1,000	5
3	1,500	1,700	1,300	1,300	5
4	1,300	1,100	1,800	1,100	5
5	1,200	1,100	1,300	1,100	5
6	1,000	1,200	1,200	1,000	10
7	1,500	1,700	1,200	1,200	5
8	1,300	1,700	1,000	1,000	10
9	1,800	1,200	1,100	1,100	15
10	1,300	1,300	1,100	1,100	5
11	1,400	1,300	1,900	1,300	10
12	1,500	1,300	1,400	1,300	5
13	1,500	1,800	1,200	1,200	10
14	1,000	1,900	1,400	1,000	5
15	1,300	1,700	1,700	1,300	5
Total					110

probability is exactly 1.00). The distribution of delay time can be changed (again, making sure that the cumulative probability is exactly 1.00). Also, the parameters of the problem can be changed (bearing cost per unit, and so forth). As in other spreadsheet models, the number of trials can be varied from 1 to 400. Finally, in the Experiment sheet, the endpoints of the bins can be changed for observing the frequency of total cost for 10,000 hours of bearing life.

Example 2.6: Random Normal Numbers

Consider a bomber attempting to destroy an ammunition depot, as shown in Figure 2.15. (This bomber has conventional rather than laser-guided weapons.) If a bomb falls anywhere on the target, a hit is scored; otherwise, the bomb is a miss. The bomber flies in the horizontal direction and carries 10 bombs. The aiming point is (0, 0). The point of impact is assumed to be normally distributed around the aiming point with a standard deviation of 400 meters in the direction of flight and 200 meters in the perpendicular direction. The problem is to simulate the operation and make statements about the number of bombs on target.

Recall that the standardized normal variate, Z, having mean 0 and standard deviation 1, is distributed as

$$Z = \frac{X - \mu}{\sigma}$$

where X is a normal random variable, μ is the mean of the distribution of X, and σ is the standard deviation of X. Then,

$$X = Z\sigma_x$$
$$Y = Z\sigma_Y$$

where (X, Y) are the simulated coordinates of the bomb after it has fallen. With $\sigma_x = 400$ and $\sigma_Y = 200$ we have

$$X = 400Z_i$$
$$Y = 200Z_j$$

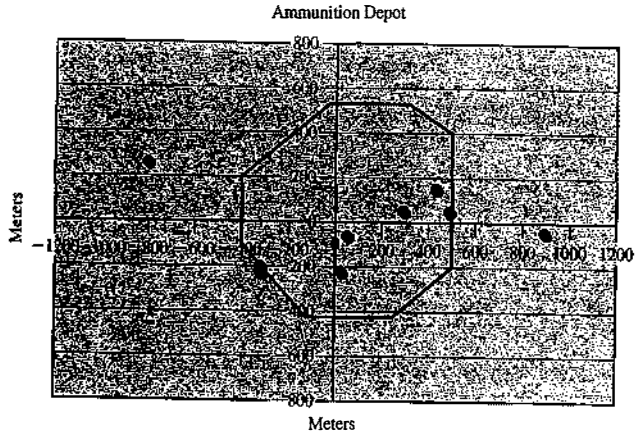


Figure 2.15 Ammunition depot.

The i and j subscripts have been added to indicate that the values of Z should be different. What are these Z values and where can they be found? The values of Z are random normal numbers. These can be generated from uniformly distributed random numbers, as will be discussed in Chapter 7. A small sample of random normal numbers is given in Table A.2. The table of random normal numbers is used in the same way as the table of random numbers: that is, start at a random place in the table and proceed in a systematic direction, avoiding overlap.

An example of one bomber's run will indicate how the simulation is performed. Table 2.26 shows the results of a simulated run. The random normal numbers in Table 2.26 are shown to four-decimal-place accuracy.

The mnemonic RNN_x stands for "random normal number to compute the x coordinate" and corresponds to Z_i. The first random normal number used was 2.2296, generating the x-coordinate 400(2.2296) = 891.8. The random normal number to generate the y-coordinate was −0.1932, resulting in the y-coordinate −38.6. Taken together, (891.8, −38.6) is a miss, for it is off the target. As shown in Table 2.26, there were 5 hits and 5 misses.

Table 2.26 Simulated Bombing Run

Bomb	RNN_x	X Coordinate ($400\ RNN_x$)	RNN_y	Y Coordinate ($200\ RNN_y$)	Result[a]
1	2.2296	891.8	−0.1932	−38.6	Miss
2	−2.0035	−801.4	1.3034	260.7	Miss
3	−3.1432	−1257.3	0.3286	65.7	Miss
4	−0.7968	−318.7	−1.1417	−228.3	Miss
5	1.0741	429.6	0.7612	152.2	Hit
6	0.1265	50.6	−0.3098	−62.0	Hit
7	0.0611	24.5	−1.1066	−221.3	Hit
8	1.2182	487.3	0.2487	49.7	Hit
9	−0.8026	−321.0	−1.0098	−202.0	Miss
10	0.7324	293.0	0.2552	−51.0	Hit

[a]Total: 5 hits, 5 misses

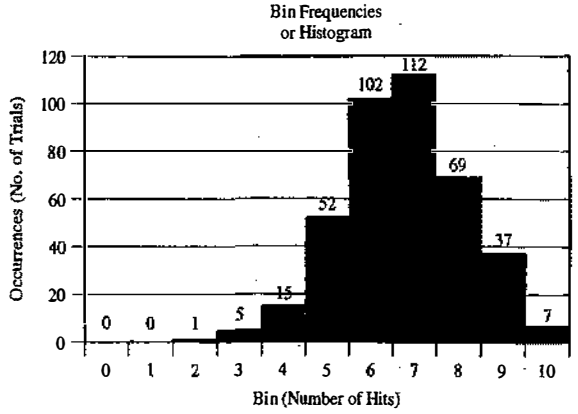


Figure 2.16 Results of 400 trials of the bombing mission.

The spreadsheet for Example 2.6 at www.bcnn.net makes it possible to conduct lots of experiments. Note that the target shown in the One Trial sheet is flexible. It can be changed by editing the (X, Y) coordinates, shown in green, so long as a convex shape is maintained. With the standard deviation in the x-direction set at 400 meters and the standard deviation in the y-direction set at 200 meters, and with the shape of the target unchanged, an experiment was run with 400 trials (each trial being 10 bombs). A result is shown in Figure 2.16.

Notice that the results range from two hits to ten hits. The average is 6.72 hits. If only one mission (trial) is run, a very misleading result could occur, but Figure 2.16 provides useful descriptive information. For instance, 44% [(175/400) × 100%] of the bombing runs there are six or fewer hits. In about 71% of the cases, [(283/400) × 100%] there were six, seven, or eight hits.

Example 2.7: Lead-Time Demand

Lead-time demand occurs in an inventory system when the lead time is not instantaneous. The lead time is the time from placement of an order until the order is received. Assume that lead time is a random variable. During the lead time, demands also occur at random. Lead-time demand is thus a random variable defined as the sum of the demands over the lead time, or $\sum_{i=0}^{T} D_i$, where i is the time period of the lead time, $i = 0, 1, 2, \ldots$; D_i is the demand during the ith time period; and T is the lead time. The distribution of lead-time demand is found by simulating many cycles of lead time and building a histogram based on the results.

A firm sells bulk rolls of newsprint. The daily demand is given by the following probability distribution:

Daily Demand (Rolls)	3	4	5	6
Probability	0.20	0.35	0.30	0.15

The lead time is the number of days from placing an order until the firm receives the order from the supplier. In this instance, lead time is a random variable given by the following distribution:

Lead Time (Days)	1	2	3
Probability	0.36	0.42	0.22

Table 2.27 shows the random digit assignment for demand and Table 2.28 does the same for lead time. The incomplete simulation table is shown in Table 2.29. The random digits for the first cycle were 57. This generates a lead time of 2 days. Thus, two pairs of random digits must be generated for the daily demand. The first of these pairs is 11, which leads to the demand 3. This is followed by the demand 5. The lead-time demand for the first cycle is 8. After many cycles are simulated, a histogram is generated. Click on "Generate New Trial" repeatedly to see the effect of randomness on a 20-cycle trial.

Although the probabilities for each value of lead-time demand can be generated in this case, simulation can also be used to sample from one or more distributions. The resulting distribution of lead-time demand

Table 2.27 Random Digit Assignment for Demand

Daily Demand	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
3	0.20	0.20	01–20
4	0.35	0.55	21–55
5	0.30	0.85	56–85
6	0.15	1.00	86–00

Table 2.28 Random Digit Assignment for Lead Time

Lead Time (Days)	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
1	0.36	0.36	01–36
2	0.42	0.78	37–78
3	0.22	1.00	79–00

Table 2.29 Simulation Table for Lead-Time Demand

Cycle	*Random Digits for Lead Time*	*Lead Time (Days)*	*Random Digits for Demand*	*Demand*	*Lead-Time Demand*
1	57	2	11	3	
			64	5	8
2	33	1	37	4	4
3	46	2	13	3	
			80	5	8
4	91	3	27	4	
			66	5	
			47	4	13
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

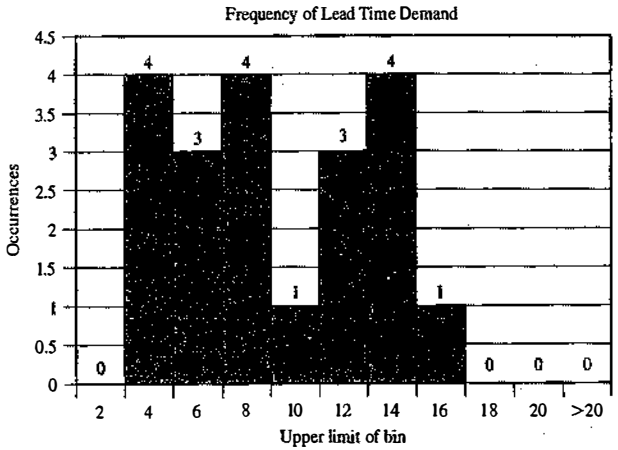


Figure 2.17 Frequency of lead-time demand.

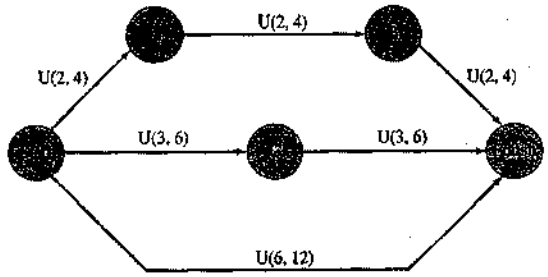


Figure 2.18 Activity network.

might be like that in Figure 2.17. This result was obtained from simulating 20 cycles of lead-time demand, using the Excel spreadsheet found at www.bcnn.net.

Suppose you have a project that requires the completion of a number of activities. Some activities must be carried out sequentially; others can be done in parallel. The project can be represented by a network of activities, as shown in Figure 2.18. There are three paths through the network, each path representing a sequence of activities that must be completed in order. The activities on two different paths can be carried out in parallel.

In the activity network in Figure 2.18, the arcs represent activities and the nodes represent the start or end of an activity. The time to complete all activities on a path is the sum of the activity times along the path. To complete the entire project, all activities must be completed; therefore, project completion time is the maximum over all path completion times.

The topmost path is along the path Start → A → B → Finish. The middle path is along the path Start → C → Finish. The bottom path is given by Start → Finish.

Example 2.8: Project Simulation

Here's a concrete example using Figure 2.18. Let's say that three friends wanted to cook bacon, eggs, and toast for breakfast for some weekend visitors. Each friend was going to prepare one of the three items. The activities might be as follows:

Top path:	Start	→	A	Crack eggs
	A	→	B	Scramble eggs
	B	→	Finish	Cook eggs
Middle path:	Start	→	C	Make toast
	C	→	Finish	Butter toast
Bottom path:	Start	→	Finish	Fry the bacon

Let's say that the times to accomplish each of the activities in preparing this breakfast are variable and can be represented by a uniform distribution between a lower and upper limit, as shown in Figure 2.18. You want to know something about the preparation time so that you can tell the visitors to be at the breakfast table on time. Perhaps you want to estimate the probability of preparing breakfast within a specified amount of time.

The activity times are shown on the arcs of the activity network. For example, the activity time from Start → A (i.e., crack the eggs) is assumed to be uniformly distributed between 2 and 4 minutes. That means that all times between 2 and 4 are equally likely to occur. The expected value, or mean time, for this activity is the midpoint, three minutes.

Applying that logic, the expected value along the topmost path is nine minutes, which is determined by adding the three expected values (3 + 3 + 3). The shortest possible completion time, which is determined by adding the smallest values, is six minutes (2 + 2 + 2). The largest possible time along the top path is twelve minutes (4 + 4 + 4).

Similarly, the expected value through the middle path is nine minutes, while the smallest and largest times are six and twelve minutes, respectively. The bottom path has the same expected value and the same extreme values.

The time that the project will be completed is the maximum time through any of the paths. (Thinking again about the preparation of this breakfast, the time that everything will be ready is when the eggs, the toast and the bacon are ready.) But, since activity times are assumed to have some random variability, the times through the paths are not constant.

Pritsker [1995] showed how such a project could be analyzed with a simulation of independent replications of the activity times. For a uniform distribution, a simulated activity time is given by

Simulated Activity Time = Lower limit + (Upper limit − Lower limit) * Random number

With a table of random numbers, the time for each simulated activity can be computed manually. For example, for activity Start → A, if the random number is 0.7943, the simulated activity time is 2 + (4 −2) * 0.7943 = 3.5886, or 3.59 minutes.

The Experiment worksheet in the Excel workbook for Example 2.8 found at www.bcnn.net allows from 1 to 400 trials and computes the average, median, minimum and maximum values. With 400 trials and using the default seed, the results are as follows:

Mean 10.12 minutes
Minimum 6.85 minutes
Maximum 12.00 minutes

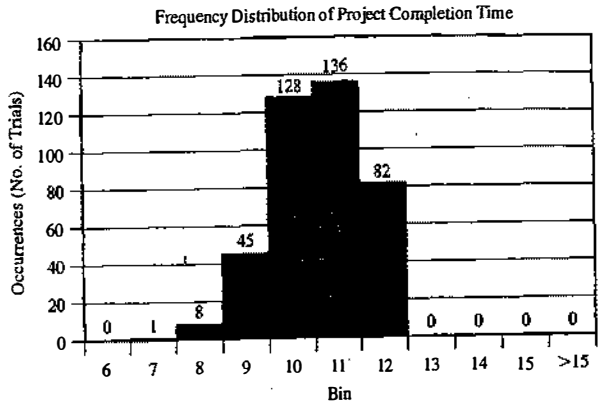


Figure 2.19 Bin frequencies for completion times.

The so-called critical path is the path that takes the longest time; that is, its time is the project completion time. For each of the 400 trials, the experiment determines which path was critical, with these results:

Top path 30.00% of the trials
Middle path 31.25%
Bottom path 38.75%

We conclude that the chance of the bacon being the last item ready is 38.75%. Is this a fluke? Why aren't the paths each represented about 1/3 of the time? The answer to this question is left for a later exercise.

Lastly, the project completion times were placed in a frequency chart. These differ each time that the spreadsheet is recalculated, but, in any large number of trials, the basic shape of the frequency chart (or histogram) will remain roughly the same. Starting from the default seed, the resulting frequency chart for project completion time is shown in Figure 2.19. Inferences that could be drawn include the following:

13.5% of the time (54 of 400), the breakfast will be ready in 9 minutes or less.
20.5% of the time (82 of 400), it will take from 11 to 12 minutes.

2.4 SUMMARY

This chapter introduced simulation concepts by means of examples, to illustrate general areas of application and to motivate the remaining chapters. In the next chapter, we give a more systematic presentation of the basic concepts.

Ad-hoc simulation tables were used in completing each example. Events in the tables were generated by using uniformly distributed random numbers and, in one case, random normal numbers. The examples illustrate the need for working out the characteristics of the input data, generating random variables from the input models, and analyzing the resulting response. The queueing examples, especially the two-channel queue, illustrate some of the complex dependencies that can occur—in this example, between subsequent customers visiting the queue. Because of these complexities, the ad-hoc simulation table approach fails, or becomes unbearably complex, even with relatively simple networks of queues. For this and other reasons, a more

systematic methodology, such as the event scheduling approach described in Chapter 3, is needed. These subjects are treated in more detail in the remaining chapters of the text.

Examples are drawn principally from queueing and inventory systems, because a large number of simulations concern problems in these areas. Additional examples are given in the areas of reliability, static simulation, the generation of a random sample from an unknown distribution, and project management. All of the examples are modeled using Excel. These are available at www.bcnn.net.

REFERENCES

HADLEY G., AND T. M. WHITIN [1963], *Analysis of Inventory Systems*, Prentice-Hall, Englewood Cliffs, NJ.

HARRISON, J.M., AND V. NGUYEN [1995], "Some Badly Behaved Closed Queueing Networks", *Proceedings of IMA Workshop on Stochastic Networks*, eds. F. Kelly and R. J. Williams.

PRITSKER, A. A. B. [1995], *Introduction to Simulation and SLAM II*, 4th ed., John Wiley, New York.

SEILA, A., V. CERIC, AND P. TADIKAMALLA [2003], *Applied Simulation Modeling*, Duxbury, Belmont, CA.

EXERCISES

Manual Exercises

Most of these exercises could also be solved in Excel. For hints on implementation in Excel, study the spreadsheet solutions at www.bcnn.net and read the "Explain" worksheet, as appropriate for the problem being solved. Another suggestion for solving these exercises in Excel is to take advantage of the VB functions in the spreadsheet solutions in www.bcnn.net. Start with a supplied example, delete the existing inputs and simulation table in the One Trial worksheet, and use what remains. On the Experiment worksheet, change the response in the cell just below the word "Link."

1. The daily demand for a product is found to follow the distribution as

Demand	*Probability*
10	0.25
11	0.35
12	0.30
13	0.10

Determine the total demand for the next 10 days.

2. A baker is trying to figure out how many dozens of bagels to bake each day. The probability distribution of the number of bagel customers is as follows:

Number of Customers/Day	8	10	12	14
Probability	0.35	0.30	0.25	0.10

Customers order 1, 2, 3, or 4 dozen bagels according to the following probability distribution.

Number of Dozen Ordered/Customer	1	2	3	4
Probability	0.4	0.3	0.2	0.1

Bagels sell for $8.40 per dozen. They cost $5.80 per dozen to make. All bagels not sold at the end of the day are sold at half-price to a local grocery store. Based on 5 days of simulation, how many dozen (to the nearest 5 dozen) bagels should be baked each day?

3. Develop and interpret flow diagrams analogous to Figures 2.2 and 2.3 for a queueing system with i channels.

4. There is only one telephone in a public booth of a railway station. The following tables indicate the distributions of callers' arrival time and duration of the calls.

Time between arrivals (Minutes)	5	6	7
Probability	0.20	0.70	0.10

Call duration (Minutes)	2	3	4	5
Probability	0.15	0.6	0.15	0.1

Simulate for 100 arrivals of the current system. It is proposed to add another telephone to the booth. Justify the proposal based on the waiting time of callers.

5. The random variables A, B, and C are distributed as

$A \sim N(\mu = 110, \sigma^2 = 110)$

$B \sim N(\mu = 300, \sigma^2 = 230)$

$C \sim N(\mu = 50, \sigma^2 = 60)$

Simulate 50 values of random variable

$$M = \frac{2A+B}{C}$$

Prepare a histogram of the resulting values, using class intervals of width equal to 3.

6. Given A, B, and C, which are uncorrelated random variables. Variable A is normally distributed with $\mu = 100$ and $\sigma^2 = 400$. Variable B is discrete uniformly distributed with probability distribution given by $p(b) = 1/5$ with $b = 0, 1, 2, 3$ and 4. Variable C is distributed in accordance with the following table.

Value of C	*Probability*
10	.10
20	.25
30	.50
40	.15

Use simulation to estimate the mean of a new variable D, that is defined as

$$D = (A - 25B)/(2C)$$

Use a sample of size 10.

7. Estimate, by simulation, the average number of lost sales per week for an inventory system that functions as follows:

   (a) Whenever the inventory level falls to or below 10 units, an order is placed. Only one order can be outstanding at a time.
   (b) The size of each order is equal to $20 - I$, where I is the inventory level when the order is placed.
   (c) If a demand occurs during a period when the inventory level is zero, the sale is lost.
   (d) Daily demand is normally distributed, with a mean of 5 units and a standard deviation of 1.5 units. (Round off demands to the closest integer during the simulation, and, if a negative value results, give it a demand of zero.)
   (e) Lead time is distributed uniformly between zero and 5 days—integers only.
   (f) The simulation will start with 18 units in inventory.
   (g) For simplicity, assume that orders are placed at the close of the business day and received after the lead time has occurred. Thus, if lead time is one day, the order is available for distribution on the morning of the second day of business following the placement of the order.
   (h) Let the simulation run for 5 weeks.

8. AGV is used to carry components between two assembly stations, namely A and B. Three types of components (C1, C2, and C3) from station A are assembled at station B. The interarrival time of C1, C2, and C3 are

Component	*Interarrival Time (Minutes)*
C1	7 ± 2
C2	3 ± 1
C3	8 ± 3

   The AGV can take only three components at a time. It takes 90 seconds to travel (to and fro) and 30 seconds to unload at station B. The AGV waits at station A till it gets the full load of three components. Simulate the system for 1 hour and determine the average waiting times of the three components.

9. The random variables $K1$, $K2$, and $K3$ are distributed as

   $K1 \sim 20 \pm 10$

   $K2 \sim 12 \pm 10$

   $K3 \sim 5 \pm 4$

   Simulate 100 values of random variable

$$M = \frac{2K1 + K2}{K3}$$

   and compute the average value.

10. Consider the assembly of two steel plates, each plate having a hole drilled in its center. The plates are to be joined by a pin. The plates are aligned for assembly relative to the bottom left corner (0,0). The hole placement is centered at (3, 2) on each plate. The standard deviation in each direction is 0.0045. The hole diameter is normally distributed, with a mean of 0.3 and a standard deviation of 0.005. The pin diameter is also distributed normally, with a mean of 0.29 and a standard deviation of 0.004. What fraction of pins will go through the assembled plates? Base your answer on a simulation of 50 observations.

   Hint: Clearance = $\text{Min}(h_1, h_2) - [(x_1 - x_2)^2 + (y_1 - y_2)^2]^{.5} - p$, where

   h_i = hole diameter, i = plate 1, 2
   p = pin diameter
   x_i = distance to center of plate hole, horizontal direction, i = 1, 2
   y_i = distance to center of plate hole, vertical direction, i = 1, 2

11. In the exercise above, the pin will wobble if it is too loose. Wobbling occurs if $\text{Min}(h_1, h_2) - p \geq 0.006$. What fraction of the assemblies wobble? (Conditional probability—i.e., given that the pins go through)

12. Three points are chosen at random on the circumference of a circle. Estimate the probability that they all lie on the same semicircle, by Monte Carlo sampling methods. Perform 5 replications.

13. Two theorems from statistics are as follows:

   Theorem 1 Let $Z_1, Z_2, \ldots, Z_k$ be normally and independently distributed random variables with mean $\mu = 0$ and variance $\sigma^2 = 1$. Then the random variable

$$\chi^2 = Z_1^2 + Z_2^2 + \cdots + Z_k^2$$

   is said to follow the chi-squared distribution with k degrees of freedom, abbreviated χ_k^2.

   Theorem 2 Let $Z \sim N(0, 1)$ and V be a chi-square random variable with k degrees of freedom. If Z and V are independent, then the random variable

$$T = \frac{Z}{\sqrt{V/k}}$$

   is said to follow the t distribution with k degrees of freedom, abbreviated t_k.

   Generate a t-distributed random variate with 3 degrees of freedom. Use the following random values in the order shown:

random digits	random normal numbers
6729	1.06
1837	−0.72
2572	0.28
8134	−0.18
5251	−0.63

14. Students arrive at the university library counter with interarrival times distributed as

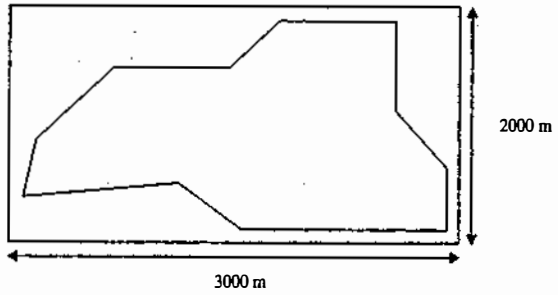
Time between arrivals (Minutes)	5	6	7	8
Probability	0.1	0.4	0.3	0.2

   The time for a transaction at the counter is distributed as

Transaction time (Minutes)	2	3	4	5
Probability	0.15	0.5	0.2	0.15

If more than two students are in the queue, an arriving student goes away without joining the queue. Simulate the system and determine the balking rate of the students.

15. The sketch of a city golf link is given. Using simulation determine the area of the golf link.



16. In Example 2.2, assume that the average delay with another server, Charlie, is virtually zero. But another server costs $20/hour. If the average caller delay costs $200 per hour, at what point is it advisable to add another server?

17. In Example 2.7, the only way to have a lead-time demand of 3 or 4 is to have lead time of one day and a demand of 3 or 4. The probability of that happening is 0.36(0.20 + 0.35) = 0.198. Using this logic, what is the probability of having lead-time demand equal 11 or 12? (Use computation, not simulation.)

18. In Example 2.3, what is the probability of having demand equal to 50 papers, immaterial of the type of the day? (Use computation, not simulation.)

Spreadsheet Exercises

19. In Example 2.1, assume that the interarrival times are distributed as

Interarrival time (Minutes)	2	4	6	8	10
Probability	0.15	0.2	0.3	0.2	0.15

Run the experiment for 50 trials. Is there any difference between the bin frequencies shown and those of Figure 2.8?

20. In Example 2.1, assume that the service time is uniformly distributed between 1 and 6 minutes (consider only integers). Run the experiment for 100 trials. Analyze the impact of this change, over the waiting time statistics.

21. Run the experiment in Example 2.1 for 25, 50, 100, 200, and 400 trials. (All trials are with different sets of random numbers.) What are the differences in the minimum and maximum values of the average waiting times? If there are differences, how do you explain them?

22. Re-do Example 2.2 with 10 experiments of 150 trials each. What is the probability that a caller has an average delay of 4 minutes or more?

23. In Example 2.2, conduct an experiment of 400 trials. Explain the large spread between the minimum average delay and the maximum average delay.

24. In Example 2.2, run 10 experiments with 50 trials each and 10 experiments with 400 trials each. Explain the differences in output that you observe, if any, between the two experiments.

25. In Example 2.2, modify the spreadsheet so that the number of calls taken by Able and Baker can be displayed. What fraction did each take in an experiment of 400 trials?

26. In Example 2.3, determine the best policy for ordering newspapers assuming zero opportunity cost. Compare the result with that obtained assuming positive opportunity cost.

27. In Example 2.3, assume that due to competition the newsstand can bargain and buy papers for 30 cents each. Verify whether there will be any change in the ordering policy obtained earlier. (All other costs remain unchanged.)

28. In Example 2.3, analyze the effect of change in probability of newsdays to 0.25, 0.5, and 0.25 for good, fair, and poor types, respectively.

29. At what bearing cost in Example 2.5 is the total cost per 10,000 hours virtually the same for the current and proposed systems? Base your estimate on 10 experiments of the current system and of the proposed system, each experiment consisting of 400 trials.

30. Change the cell widths on the experiment in the Excel spreadsheet for Example 2.5 (current or proposed) to a width of $50 beginning below the minimum value (for example, if the minimum value is $1528.46, let the first cell begin at $1500). What is the advantage of doing this?

31. Using the spreadsheet for Example 2.5 (proposed), run 10 experiments of 40 trials each, and record the range (maximum value–minimum value) of the results. Next, compute the average range. Then, do the same as before, except use 400 trials in each experiment. If there is a difference, how do you explain it?

32. In Example 2.5 (proposed), assume the following delay-time distribution:

Delay time (Minutes)	4	8	12	16
Probability	0.25	0.25	0.25	0.25

Re-run the model and check the impact of this change in the final decision.

33. Set $\sigma_x = 400$ metres and $\sigma_y = 400$ metres in the spreadsheet for Example 2.6. Leave the target intact. Conduct a simulation with 200 trials. Determine the average number of hits.

34. Set $\sigma_x = 2\ \sigma_y$ metres in the spreadsheet for Example 2.6. Leave the target intact. What is the value of σ_x if the average number of hits is to be about 6.0, based on one experiment of 400 trials.

35. In Example 2.7, suppose you wanted a better estimate of the distribution of lead-time demand: Would you (1) increase the number of cycles on the "One Trial" worksheet? or (2) increase the number of trials on the "Experiment" worksheet? Explain your answer.

36. In Example 2.7, let the demand probabilities be equal for the possible values 3, 4, 5, and 6. Run an experiment with 400 trials. Compare the average lead-time demand from using the original input data and from using the new input data.

37. In Example 2.7, let the lead-time probabilities be equal for the possible values 1, 2, and 3. Run an experiment with 400 trials. Compare the average lead-time demand from using the original input data and from using the new input data.

38. In Example 2.8, recalculate the spreadsheet 20 times, each with 400 trials. Record the number of times that the middle path was the longest. What is your best guess of the true mean of the fraction of time that the middle path is taken?

39. In the above exercise, what is the smallest value and the largest value encountered for the number of times that the middle path was selected? What if you had conducted one simulation, gotten the smallest (or largest) value, and reported that as the result? How could you argue yourself out of that jam?

40. In Example 2.8, suppose the third pathway (the bottom one in Figure 2.18) is changed so that it consists of six U(1, 2) activities. Modify the spreadsheet to accommodate this. Which is the most frequently occurring path now? What insight does this exercise provide?

41. Using Excel, generate 1000 values in a column, using the formula = −10 * LN(RAND()).

(a) Compute descriptive statistics about the data in that column, including the minimum value, the maximum value, the mean, the median, and the standard deviation.
(b) Tabulate the values into 10 bins, each of width equal to five: the first bin beginning at 0, and the eleventh bin for overflow (if any).
(c) Does the histogram resemble any distribution with which you are familiar? If so, what is its name?

Hint: Use FREQUENCY in Excel to form bins.

42. Using Excel, generate 12 columns, each with 250 values, using the formula
= RAND().

In cell M1, place the formula
= SUM(A1:L1)-6 and copy it to the 249 cells below M1 in column M.

(a) Compute descriptive statistics about the data in that column, including minimum value, maximum value, mean, median, and standard deviation.
(b) Tabulate the values with 9 bins: the first bin will include all values less than or equal to −3.5; the next six bins are of width one; the last bin will include all values greater than 3.5.
(c) Does the histogram resemble any distribution with which you are familiar? If so, what is its name?

Hint 1: Use FREQUENCY in Excel to form bins.

Hint 2: The values in Column M can be used instead of those in Table A.2.

43. Using Excel, generate 1000 random numbers in the range (0–1000) and form a frequency table with 10 class intervals.

44. Using Excel, generate 100 random numbers equally distributed between 23 and 87.

45. Consider Example 2.5. If the proposed system is modified as follows: whenever a bearing fails, two bearings are replaced

(a) The one that has failed and
(b) another one, out of the other two remaining bearings with longest operational time.

Using Excel, simulate the system and compare the cost with previous policy of changing all the three bearings.

46. Consider the simulation of a management game. There are three players A, B, and C. Each player has two strategies which they play with equal probabilities. The players select strategies independently. The following table gives the payoff.

	Payoff		
Strategies	*A*	*B*	*C*
A1-B1-C1	10	−5	5
A1-B1-C2	0	8	2
A1-B2-C1	9	3	−2
A1-B2-C2	−4	5	9
A2-B1-C1	6	1	3
A2-B1-C2	0	0	10
A2-B2-C1	6	10	−6
A2-B2-C2	0	4	6

Simulate 100 plays and determine the payoffs.

3

General Principles

This chapter develops a common framework for the modeling of complex systems by using discrete-event simulation. It covers the basic building blocks of all discrete-event simulation models: entities and attributes, activities, and events. In discrete-event simulation, a system is modeled in terms of its state at each point in time; of the entities that pass through the system and the entities that represent system resources; and of the activities and events that cause the system state to change. Discrete-event models are appropriate for those systems for which changes in system state occur only at discrete points in time.

The simulation languages and software (collectively called simulation packages) described in Chapter 4 are fundamentally packages for discrete-event simulation. A few of the packages also include the capability to model continuous variables in a purely continuous simulation or a mixed discrete–continuous model. The discussion in this chapter focuses on the discrete-event concepts and methodologies. The discussion in Chapter 4 focuses more on the capabilities of the individual packages and on some of their higher-level constructs.

This chapter introduces and explains the fundamental concepts and methodologies underlying all discrete-event simulation packages. These concepts and methodologies are not tied to any particular package. Many of the packages use different terminology from that used here, and most have a number of higher-level constructs designed to make modeling simpler and more straightforward for their application domain. For example, this chapter discusses the fundamental abstract concept of an entity, but Chapter 4 discusses more concrete realizations of entities, such as machines, conveyors, and vehicles that are built into some of the packages to facilitate modeling in the manufacturing, material handling, or other domains.

Applications of simulation in specific contexts are discussed in Part Five of this text. Topics covered include the simulation of manufacturing and material handling systems in Chapter 13, the simulation of computer systems in Chapter 14, and the simulation of communications systems in Chapter 15.

Section 3.1 covers the general principles and concepts of discrete-event simulation, the event scheduling/time advance algorithm, and the three prevalent world views: event scheduling, process interaction,

and activity scanning. Section 3.2 introduces some of the notions of list processing, one of the more important methodologies used in discrete-event simulation software. Chapter 4 covers the implementation of the concepts in a number of the more widely used simulation packages.

3.1 CONCEPTS IN DISCRETE-EVENT SIMULATION

The concept of a system and a model of a system were discussed briefly in Chapter 1. This chapter deals exclusively with dynamic, stochastic systems (i.e., involving time and containing random elements) that change in a discrete manner. This section expands on these concepts and proposes a framework for the development of a discrete-event model of a system. The major concepts are briefly defined and then illustrated by examples:

System A collection of entities (e.g., people and machines) that interact together over time to accomplish one or more goals.

Model An abstract representation of a system, usually containing structural, logical, or mathematical relationships that describe a system in terms of state, entities and their attributes, sets, processes, events, activities, and delays.

System state A collection of variables that contain all the information necessary to describe the system at any time.

Entity Any object or component in the system that requires explicit representation in the model (e.g., a server, a customer, a machine).

Attributes The properties of a given entity (e.g., the priority of a waiting customer, the routing of a job through a job shop).

List A collection of (permanently or temporarily) associated entities, ordered in some logical fashion (such as all customers currently in a waiting line, ordered by "first come, first served," or by priority).

Event An instantaneous occurrence that changes the state of a system (such as an arrival of a new customer).

Event notice A record of an event to occur at the current or some future time, along with any associated data necessary to execute the event; at a minimum, the record includes the event type and the event time.

Event list A list of event notices for future events, ordered by time of occurrence; also known as the future event list (FEL).

Activity A duration of time of specified length (e.g., a service time or interarrival time), which is known when it begins (although it may be defined in terms of a statistical distribution).

Delay A duration of time of unspecified indefinite length, which is not known until it ends (e.g., a customer's delay in a last-in–first-out waiting line which, when it begins, depends on future arrivals).

Clock A variable representing simulated time, called CLOCK in the examples to follow.

Different simulation packages use different terminology for the same or similar concepts—for example, lists are sometimes called sets, queues, or chains. Sets or lists are used to hold both entities and event notices. The entities on a list are always ordered by some rule, such as first-in–first-out or last-in–first-out, or are ranked by some entity attribute, such as priority or due date. The future event list is always ranked by the event time recorded in the event notice. Section 3.2 discusses a number of methods for handling lists and introduces some of the methodologies for efficient processing of ordered sets or lists.

An activity typically represents a service time, an interarrival time, or any other processing time whose duration has been characterized and defined by the modeler. An activity's duration may be specified in a number of ways:

1. Deterministic—for example, always exactly 5 minutes;
2. Statistical—for example, as a random draw from among 2, 5, 7 with equal probabilities;
3. A function depending on system variables and/or entity attributes—for example, loading time for an iron ore ship as a function of the ship's allowed cargo weight and the loading rate in tons per hour.

However it is characterized, the duration of an activity is computable from its specification at the instant it begins. Its duration is not affected by the occurrence of other events (unless, as is allowed by some simulation packages, the model contains logic to cancel an event). To keep track of activities and their expected completion time, at the simulated instant that an activity duration begins, an event notice is created having an event time equal to the activity's completion time. For example, if the current simulated time is CLOCK = 100 minutes and an inspection time of exactly 5 minutes is just beginning, then an event notice is created that specifies the type of event (an end-of-inspection event), and the event time (100 + 5 = 105 minutes).

In contrast to an activity, a delay's duration is not specified by the modeler ahead of time, but rather is determined by system conditions. Quite often, a delay's duration is measured and is one of the desired outputs of a model run. Typically, a delay ends when some set of logical conditions becomes true or one or more other events occur. For example, a customer's delay in a waiting line may be dependent on the number and duration of service of other customers ahead in line as well as the availability of servers and equipment.

A delay is sometimes called a *conditional wait*, an activity an *unconditional wait*. The completion of an activity is an event, often called a *primary event*, that is managed by placing an event notice on the FEL. In contrast, delays are managed by placing the associated entity on another list, perhaps representing a waiting line, until such time as system conditions permit the processing of the entity. The completion of a delay is sometimes called a conditional or secondary event, but such events are not represented by event notices, nor do they appear on the FEL.

The systems considered here are dynamic, that is, changing over time. Therefore, system state, entity attributes and the number of active entities, the contents of sets, and the activities and delays currently in progress are all functions of time and are constantly changing over time. Time itself is represented by a variable called CLOCK.

Example 3.1: Call Center, Revisited

Consider the Able–Baker call center system of Example 2.2. A discrete-event model has the following components:

System state

$L_Q(t)$, the number of callers waiting to be served at time t;
$L_A(t)$, 0 or 1 to indicate Able as being idle or busy at time t;
$L_B(t)$, 0 or 1 to indicate Baker as being idle or busy at time t.

Entities Neither the callers nor the servers need to be explicitly represented, except in terms of the state variables, unless certain caller averages are desired (compare Examples 3.4 and 3.5).

Events

Arrival event;
Service completion by Able;
Service completion by Baker.

Activities

Interarrival time, defined in Table 2.11;
Service time by Able, defined in Table 2.12;
Service time by Baker, defined in Table 2.13.

Delay A caller's wait in queue until Able or Baker becomes free.

Clock	*System state*	*Entities and attributes*	*Set 1*	*Set 2*	...	*Future event list, FEL*	*Cumulative statistics and counters*
t	$(x, y, z, \ldots)$					$(3, t_1)$ – Type 3 event to occur at time t_1 $(1, t_2)$ – Type 1 event to occur at time t_2	

Figure 3.1 Prototype system snapshot at simulation time t.

The definition of the model components provides a static description of the model. In addition, a description of the dynamic relationships and interactions between the components is also needed. Some questions that need answers include:

1. How does each event affect system state, entity attributes, and set contents?
2. How are activities defined (i.e., deterministic, probabilistic, or some other mathematical equation)? What event marks the beginning or end of each activity? Can the activity begin regardless of system state, or is its beginning conditioned on the system being in a certain state? (For example, a machining "activity" cannot begin unless the machine is idle, not broken, and not in maintenance.)
3. Which events trigger the beginning (and end) of each type of delay? Under what conditions does a delay begin or end?
4. What is the system state at time 0? What events should be generated at time 0 to "prime" the model—that is, to get the simulation started?

A discrete-event simulation is the modeling over time of a system all of whose state changes occur at discrete points in time—those points when an event occurs. A discrete-event simulation (hereafter called a simulation) proceeds by producing a sequence of system snapshots (or system images) that represent the evolution of the system through time. A given snapshot at a given time (CLOCK = t) includes not only the system state at time t, but also a list (the FEL) of all activities currently in progress and when each such activity will end, the status of all entities and current membership of all sets, plus the current values of cumulative statistics and counters that will be used to calculate summary statistics at the end of the simulation. A prototype system snapshot is shown in Figure 3.1. (Not all models will contain every element exhibited in Figure 3.1. Further illustrations are provided in the examples in this chapter.)

3.1.1 The Event Scheduling/Time Advance Algorithm

The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order is based on the future event list (FEL). This list contains all event notices for events that have been scheduled to occur at a future time. Scheduling a future event means that, at the instant an activity begins, its duration is computed or drawn as a sample from a statistical distribution; and that the end-activity event, together with its event time, is placed on the future event list. In the real world, most future events are not

scheduled but merely happen—such as random breakdowns or random arrivals. In the model, such random events are represented by the end of some activity, which in turn is represented by a statistical distribution.

At any given time t, the FEL contains all previously scheduled future events and their associated event times (called $t_1, t_2, \ldots$ in Figure 3.1). The FEL is ordered by event time, meaning that the events are arranged chronologically—that is, the event times satisfy

$$t < t_1 \le t_2 \le t_3 \le \cdots t_n.$$

Time t is the value of CLOCK, the current value of simulated time. The event associated with time t_1 is called the imminent event; that is, it is the next event that will occur. After the system snapshot at simulation time CLOCK = t has been updated, the CLOCK is advanced to simulation time CLOCK = t_1, the imminent event notice is removed from the FEL, and the event is executed. Execution of the imminent event means that a new system snapshot for time t_1 is created, one based on the old snapshot at time t and the nature of the imminent event. At time t_1, new future events may or might not be generated, but if any are, they are scheduled by creating event notices and putting them into their proper position on the FEL. After the new system snapshot for time t_1 has been updated, the clock is advanced to the time of the new imminent event and that event is executed. This process repeats until the simulation is over. The sequence of actions that a simulator (or simulation language) must perform to advance the clock and build a new system snapshot is called the *event-scheduling/time-advance algorithm*, whose steps are listed in Figure 3.2 (and explained thereafter).

The length and contents of the FEL are constantly changing as the simulation progresses, and thus its efficient management in a computerized simulation will have a major impact on the efficiency of the computer program representing the model. The management of a list is called *list processing*. The major list processing operations performed on a FEL are removal of the imminent event, addition of a new event to the list, and occasionally removal of some event (called cancellation of an event). As the imminent event is usually at the top of the list, its removal is as efficient as possible. Addition of a new event (and cancellation of an old event) requires a search of the list. The efficiency of this search depends on the logical organization of the list and on how the search is conducted. In addition to the FEL, all the sets in a model are maintained in some logical order, and the operations of addition and removal of entities from the set also require efficient list-processing techniques. A brief introduction to list processing in simulation is given in Section 3.2.

The removal and addition of events from the FEL is illustrated in Figure 3.2. Event 3 with event time t_1 represents, say, a service completion event at server 3. Since it is the imminent event at time t, it is removed from the FEL in step 1 (Figure 3.2) of the event-scheduling/time-advance algorithm. When event 4 (say, an arrival event) with event time t^* is generated at step 4, one possible way to determine its correct position on the FEL is to conduct a top-down search:

If $t^* < t_2$,	place event 4 at the top of the FEL.
If $t_2 \le t^* < t_3$,	place event 4 second on the list.
If $t_3 \le t^* < t_4$,	place event 4 third on the list.
	⋮
If $t_n \le t^*$,	place event 4 last on the list.

(In Figure 3.2, it was assumed that t^* was between t_2 and t_3.) Another way is to conduct a bottom-up search. The least efficient way to maintain the FEL is to leave it as an unordered list (additions placed arbitrarily at the top or bottom), which would require at step 1 of Figure 3.2 a complete search of the list for the imminent event before each clock advance. (The imminent event is the event on the FEL with the lowest event time.)

The system snapshot at time 0 is defined by the initial conditions and the generation of the so-called exogenous events. The specified initial conditions define the system state at time 0. For example, in Figure 3.2, if $t = 0$,

Old system snapshot at time t

CLOCK	System state	...	Future event list	...
t	(5, 1, 6)		$(3, t_1)$ – Type 3 event to occur at time t_1 $(1, t_2)$ – Type 1 event to occur at time t_2 $(1, t_3)$ – Type 1 event to occur at time t_3 . . . $(2, t_n)$ –Type 2 event to occur at time t_n	

Event-scheduling/time-advance algorithm

Step 1. Remove the event notice for the imminent event (event 3, time t_1) from FEL.
Step 2. Advance CLOCK to imminent event time (i.e., advance CLOCK from t to t_1).
Step 3. Execute imminent event: update system state, change entity attributes, and set membership as needed.
Step 4. Generate future events (if necessary) and place their event notices on FEL, ranked by event time. (Example: Event 4 to occur at time t^*, where $t_2 < t^* < t_3$.)
Step 5. Update cumulative statistics and counters.

New system snapshot at time t_1

CLOCK	System state	...	Future event list	...
t_1	(5, 1, 5)		$(1, t_2)$ – Type 1 event to occur at time t_2 $(4, t^*)$ – Type 4 event to occur at time t^* $(1, t_3)$ – Type 1 event to occur at time t_3 . . . $(2, t_n)$ – Type 2 event to occur at time t_n	

Figure 3.2 Advancing simulation time and updating system image.

then the state (5, 1, 6) might represent the initial number of customers at three different points in the system. An exogenous event is a happening "outside the system" that impinges on the system. An important example is an arrival to a queueing system. At time 0, the first arrival event is generated and is scheduled on the FEL (meaning that its event notice is placed on the FEL). The interarrival time is an example of an activity. When the clock eventually is advanced to the time of this first arrival, a second arrival event is generated. First, an interarrival time is generated, a^*; it is added to the current time, CLOCK = t; the resulting (future) event time, $t + a^* = t^*$, is used to position the new arrival event notice on the FEL. This method of generating an external arrival stream is called bootstrapping; it provides one example of how future events are generated in step 4 of the event-scheduling/time-advance algorithm. Bootstrapping is illustrated in Figure 3.3. The first three interarrival times generated are 3.7, 0.4, and 3.3 time units. The end of an interarrival interval is an example of a primary event.

A second example of how future events are generated (step 4 of Figure 3.2) is provided by a service completion event in a queueing simulation. When one customer completes service, at current time CLOCK = t, if the next customer is present, then a new service time, s^*, will be generated for the next customer. The next service completion event will be scheduled to occur at future time $t^* = t + s^*$, by placing onto the FEL a new event notice, of type *service completion*, with event time t^*. In addition, a service-completion event will be

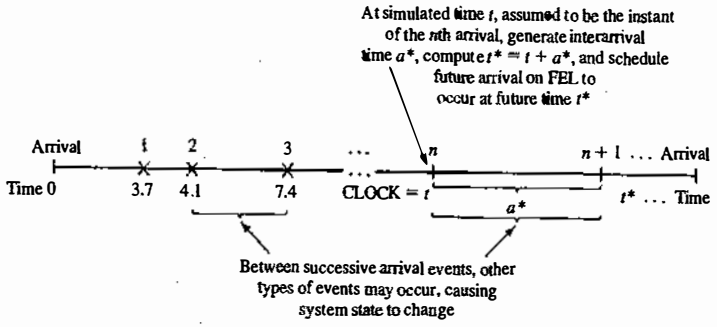


Figure 3.3 Generation of an external arrival stream by bootstrapping.

generated and scheduled at the time of an arrival event provided that, upon arrival, there is at least one idle server in the server group. A service time is an example of an activity. Beginning service is a conditional event, because its occurrence is triggered only on the condition that a customer is present and a server is free. Service completion is an example of a primary event. Note that a conditional event, such as beginning service, is triggered by a primary event' occurring and by certain conditions prevailing in the system. Only primary events appear on the FEL.

A third important example is the alternate generation of runtimes and downtimes for a machine subject to breakdowns. At time 0, the first runtime will be generated, and an end-of-runtime event will be scheduled. Whenever an end-of-runtime event occurs, a downtime will be generated, and an end-of-downtime event will be scheduled on the FEL. When the CLOCK is eventually advanced to the time of this end-of-downtime event, a runtime is generated, and an end-of-runtime event is scheduled on the FEL. In this way, runtimes and downtimes continually alternate throughout the simulation. A runtime and a downtime are examples of activities, and *end of runtime* and *end of downtime* are primary events.

Every simulation must have a stopping event, here called E, which defines how long the simulation will run. There are generally two ways to stop a simulation:

1. At time 0, schedule a stop simulation event at a specified future time T_E. Thus, before simulating, it is known that the simulation will run over the time interval $[0, T_E]$. Example: Simulate a job shop for $T_E = 40$ hours.
2. Run length T_E is determined by the simulation itself. Generally, T_E is the time of occurrence of some specified event E. Examples: T_E is the time of the 100th service completion at a certain service center. T_E is the time of breakdown of a complex system. T_E is the time of disengagement or total kill (whichever occurs first) in a combat simulation. T_E is the time at which a distribution center ships the last carton in a day's orders.

In case 2, T_E is not known ahead of time. Indeed, it could be one of the statistics of primary interest to be produced by the simulation.

3.1.2 World Views

When using a simulation package or even when doing a manual simulation, a modeler adopts a world view or orientation for developing a model. The most prevalent world views are the event-scheduling world view, as discussed in the previous section, the process-interaction world view, and the activity-scanning world view.

Even if a particular package does not directly support one or more of the world views, understanding the different approaches could suggest alternative ways to model a given system.

To summarize the previous discussion, when using the event-scheduling approach, a simulation analyst concentrates on events and their effect on system state. This world view will be illustrated by the manual simulations of Section 3.1.3 and the Java simulation in Chapter 4.

When using a package that supports the process-interaction approach, a simulation analyst thinks in terms of processes. The analyst defines the simulation model in terms of entities or objects and their life cycle as they flow through the system, demanding resources and queueing to wait for resources. More precisely, a process is the life cycle of one entity. This life cycle consists of various events and activities. Some activities might require the use of one or more resources whose capacities are limited. These and other constraints cause processes to interact, the simplest example being an entity forced to wait in a queue (on a list) because the resource it needs is busy with another entity. The process interaction approach is popular because it has intuitive appeal and because the simulation packages that implement it allow an analyst to describe the process flow in terms of high-level block or network constructs, while the interaction among processes is handled automatically.

In more precise terms, a process is a time-sequenced list of events, activities and delays, including demands for resources, that define the life cycle of one entity as it moves through a system. An example of a "customer process" is shown in Figure 3.4. In this figure, we see the interaction between two customer processes as customer $n + 1$ is delayed until the previous customer's "end service event" occurs. Usually, many processes are active simultaneously in a model, and the interaction among processes could be quite complex.

Underlying the implementation of the process interaction approach in a simulation package, but usually hidden from a modeler's view, events are being scheduled on a future event list and entities are being placed onto lists whenever they face delays, causing one process to temporarily suspend its execution while other processes proceed. It is important that the modeler have a basic understanding of the concepts and, for the simulation package being used, a detailed understanding of the built-in but hidden rules of operation. Schriber and Brunner [2003] provide understanding in this area.

Both the event-scheduling and the process-interaction approach use a variable time advance—that is, when all events and system state changes have occurred at one instant of simulated time, the simulation clock is advanced to the time of the next imminent event on the FEL. The activity-scanning approach, in contrast, uses a fixed time increment and a rule-based approach to decide whether any activities can begin at each point in simulated time.

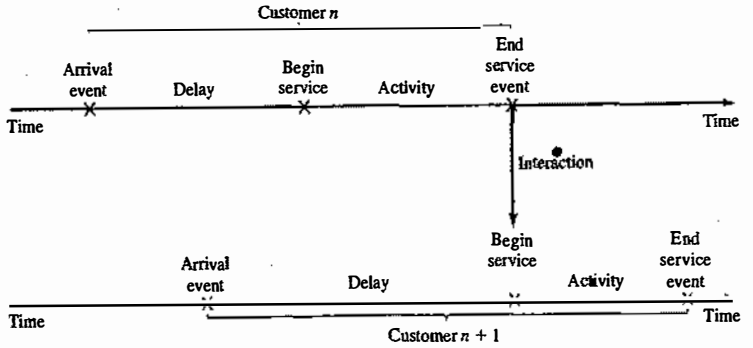


Figure 3.4 Two interacting customer processes in a single-server queue.

With the activity-scanning approach, a modeler concentrates on the activities of a model and those conditions, simple or complex, that allow an activity to begin. At each clock advance, the conditions for each activity are checked, and, if the conditions are true, then the corresponding activity begins. Proponents claim that the activity-scanning approach is simple in concept and leads to modular models that are more easily maintained, understood, and modified by other analysts at later times. They admit, however, that the repeated scanning to discover whether an activity can begin results in slow runtime on computers. Thus, the pure activity-scanning approach has been modified (and made conceptually somewhat more complex) by what is called the three-phase approach, which combines some of the features of event scheduling with activity scanning to allow for variable time advance and the avoidance of scanning when it is not necessary, but keeps the main advantages of the activity-scanning approach.

In the three-phase approach, events are considered to be activities of duration zero time units. With this definition, activities are divided into two categories, which are called B and C.

B activities activities bound to occur; all primary events and unconditional activities.
C activities activities or events that are conditional upon certain conditions being true.

The B-type activities and events can be scheduled ahead of time, just as in the event-scheduling approach. This allows variable time advance. The FEL contains only B-type events. Scanning to learn whether any C-type activities can begin or C-type events occur happens only at the end of each time advance, after all B-type events have completed. In summary, with the three-phase approach, the simulation proceeds with repeated execution of the 3 phases until it is completed:

Phase A Remove the imminent event from the FEL and advance the clock to its event time. Remove from the FEL any other events that have the same event time.
Phase B Execute all B-type events that were removed from the FEL. (This could free a number of resources or otherwise change system state.)
Phase C Scan the conditions that trigger each C-type activity and activate any whose conditions are smet. Rescan until no additional C-type activities can begin and no events occur.

The three-phase approach improves the execution efficiency of the activity-scanning method. In addition, proponents claim that the activity scanning and three-phase approaches are particularly good at handling complex resource problems in which various combinations of resources are needed to accomplish different tasks. These approaches guarantee that all resources being freed at a given simulated time will all be freed before any available resources are reallocated to new tasks.

Example 3.2: Call Center, Back Again

The events and activities were identified in Example 3.1. Under the three-phase approach, the conditions for beginning each activity in Phase C are as follows:

Activity	Condition
Service time by Able	A caller is in queue and Able is idle
Service time by Baker	A caller is in queue, Baker is idle and Able is busy

Using the process-interaction approach, we view the model from the viewpoint of a caller and its "life cycle." Considering a life cycle as beginning upon arrival, a customer process is pictured in Figure 3.4.

In summary, as will be illustrated in Chapter 4, the process-interaction approach has been adopted by the simulation packages most popular in the USA. On the other hand, a number of activity-scanning packages are popular in the UK and Europe. Some of the packages allow portions of a model to be event-scheduling based, if that orientation is convenient, mixed with the process-interaction approach. Finally, some of the packages are based on a flow chart, block diagram or network structure, which upon closer examination turns out to be a specific implementation of the process-interaction concept.

3.1.3 Manual Simulation Using Event Scheduling

In the conducting of an event-scheduling simulation, a simulation table is used to record the successive system snapshots as time advances.

Example 3.3: Single-Channel Queue

Reconsider the grocery store with one checkout counter that was simulated in Example 2.1 by an ad hoc method. The system consists of those customers in the waiting line plus the one (if any) checking out. A stopping time of 60 minutes is set for this example. The model has the following components:

System state $(LQ(t), LS(t))$, where $LQ(t)$ is the number of customers in the waiting line, and $LS(t)$ is the number being served (0 or 1) at time t.
Entities The server and customers are not explicitly modeled, except in terms of the state variables.
Events
Arrival (A);
Departure (D);
Stopping event (E), scheduled to occur at time 60.
Event notices
(A, t), representing an arrival event to occur at future time t;
(D, t), representing a customer departure at future time t;
(E, 60), representing the simulation stop event at future time 60.
Activities
Interarrival time, defined in Table 2.6;
Service time, defined in Table 2.7
Delay Customer time spent in waiting line.

The event notices are written as (event type, event time). In this model, the FEL will always contain either two or three event notices. The effect of the arrival and departure events was first shown in Figures 2.2 and 2.3 and is shown in more detail in Figures 3.5 and 3.6.

The simulation table for the checkout counter is given in Table 3.1. The reader should cover all system snapshots except one, starting with the first, and attempt to construct the next snapshot from the previous one and the event logic in Figures 3.5 and 3.6. The interarrival times and service times will be identical to those used in Table 2.10:

Interarrival Times	1	1	6	3	7	5	2	4	1…
Service Times	4	2	5	4	1	5	4	1	4…

Initial conditions are that the first customer arrive at time 0 and begin service. This is reflected in Table 3.1 by the system snapshot at time zero (CLOCK = 0), with $LQ(0) = 0$, $LS(0) = 1$, and both a departure event and arrival event on the FEL. Also, the simulation is scheduled to stop at time 60. Only two statistics, server utilization and maximum queue length, will be collected. Server utilization is defined by total server busy time (B) divided by total time (T_E). Total busy time, B, and maximum queue length, MQ, will be accumulated as the simulation progresses. A column headed "comments" is included to aid the reader. (a^* and s^* are the generated interarrival and service times, respectively.)

As soon as the system snapshot at time CLOCK = 0 is complete, the simulation begins. At time 0, the imminent event is (A, 1). The CLOCK is advanced to time 1, and (A, 1) is removed from the FEL.

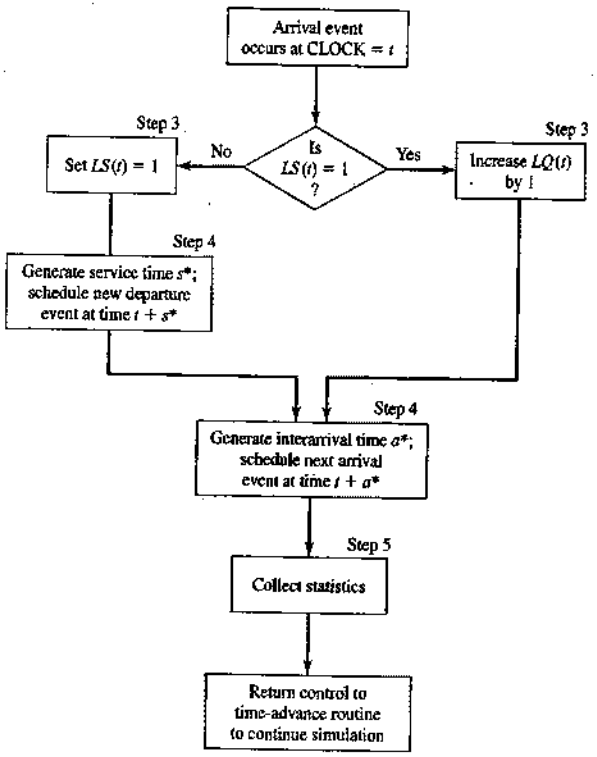


Figure 3.5 Execution of the arrival event.

Because $LS(t) = 1$ for $0 \le t \le 1$ (i.e., the server was busy for 1 minute), the cumulative busy time is increased from $B = 0$ to $B = 1$. By the event logic in Figure 3.6, set $LS(1) = 1$ (the server becomes busy). The FEL is left with three future events, $(A, 2)$, $(D, 4)$, and $(E, 60)$. The simulation CLOCK is next advanced to time 2, and an arrival event is executed. The interpretation of the remainder of Table 3.1 is left to the reader.

The simulation in Table 3.1 covers the time interval [0,23]. At simulated time 23, the system empties, but the next arrival also occurs at time 23. The server was busy for 21 of the 23 time units simulated, and the maximum queue length was two. This simulation is, of course, too short to draw any reliable conclusions. Exercise 1 asks the reader to continue the simulation and to compare the results with those in Example 2.1. Note that the simulation table gives the system state at all times, not just the listed times. For example, from time 11 to time 15, there is one customer in service and one in the waiting line.

When an event-scheduling algorithm is computerized, only one snapshot (the current one or partially updated one) is kept in computer memory. With the idea of implementing event scheduling in Java or some other general-purpose language, the following rule should be followed. A new snapshot can be derived only from the previous snapshot, newly generated random variables, and the event logic (Figures 3.5 and 3.6). Past snapshots should be ignored for advancing of the clock. The current snapshot must contain all information necessary to continue the simulation.

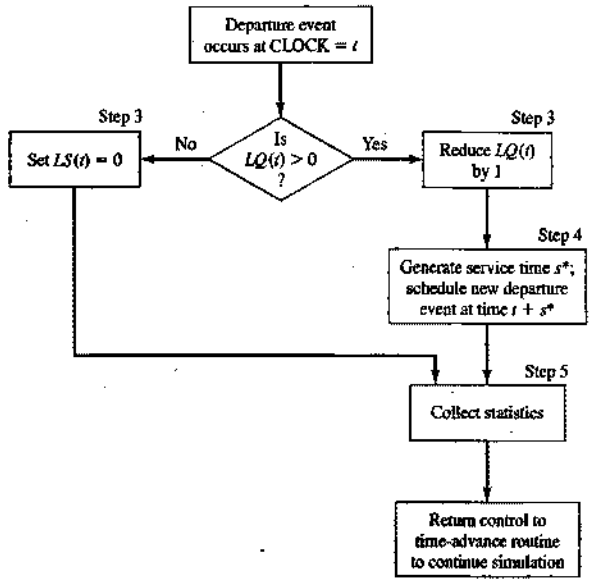


Figure 3.6 Execution of the departure event.

Table 3.1 Simulation Table for Checkout Counter (Example 3.3)

	System State				*Cumulative Statistics*	
Clock	*LQ(t)*	*LS(t)*	*Future Event List*	*Comment*	*B*	*MQ*
0	0	1	(A, 1) (D, 4) (E, 60)	First A occurs ($a^* = 1$) Schedule next A ($s^* = 4$) Schedule first D	0	0
1	1	1	(A, 2) (D, 4) (E, 60)	Second A occurs: (A, 1) ($a^* = 1$) Schedule next A (Customer delayed)	1	1
2	2	1	(D, 4) (A, 8) (E, 60)	Third A occurs: (A, 2) ($a^* = 6$) Schedule next A (Two customers delayed)	2	2
4	1	1	(D, 6) (A, 8) (E, 60)	First D occurs: (D, 4) ($s^* = 2$) Schedule next D (Customer delayed)	4	2
6	0	1	(A, 8) (D, 11) (E, 60)	Second D occurs: (D, 6) ($s^* = 5$) Schedule next D	6	2

(*continued overleaf*)

Table 3.1 (*continued*)

Clock	System State LQ(t)	System State LS(t)	Future Event List	Comment	Cumulative Statistics B	Cumulative Statistics MQ
8	1	1	(D, 11) (A, 11) (E, 60)	Fourth A occurs: (A, 8) (a^* = 3 Schedule next A (Customer delayed)	8	2
11	1	1	(D, 15) (A, 18) (E, 60)	Fifth A occurs: (A, 11) (a^* = 7) Schedule next A Third D occurs: (D, 11) (s^* = 4) Schedule next D Customer delayed	11	2
15	0	1	(D, 16) (A, 18) (E, 60)	Fourth D occurs: (D, 15) (s^* = 1) Schedule next D	15	2
16	0	0	(A, 18) (E, 60)	Fifth D occurs: (D, 16)	16	2
18	0	1	(D, 23) (A, 23) (E, 60)	Sixth A occurs (a^* = 5) Schedule next A (s^* = 5) Schedule next D	16	2
23	0	1	(A, 25) (D, 27) (E, 60)	Seventh A occurs: (A, 23) (a^* = 2) Schedule next Arrival Sixth D occurs: (D, 23)	21	2

Example 3.4: The Checkout-Counter Simulation, Continued

Suppose that, in the simulation of the checkout counter in Example 3.3, the simulation analyst desires to estimate mean response time and mean proportion of customers who spend 5 or more minutes in the system. A response time is the length of time a customer spends in the system. In order to estimate these customer averages, it is necessary to expand the model in Example 3.3 to represent the individual customers explicitly. In addition, to be able to compute an individual customer's response time when that customer departs, it will be necessary to know that customer's arrival time. Therefore, a customer entity with arrival time as an attribute will be added to the list of model components in Example 3.3. These customer entities will be stored in a list to be called "CHECKOUT LINE"; they will be called *C*1, *C*2, *C*3, ... Finally, the event notices on the FEL will be expanded to indicate which customer is affected. For example, (*D*, 4, *C*1) means that customer *C*1 will depart at time 4. The additional model components are the following:

Entities (*Ci*, *t*), representing customer *Ci* who arrived at time *t*

Event notices

(*A*, *t*, *Ci*), the arrival of customer *Ci* at future time *t*

(*D*, *t*, *Cj*), the departure of customer *Cj* at future time *t*

Set "CHECKOUT LINE," the set of all customers currently at the checkout counter (being served or waiting to be served), ordered by time of arrival

Three new cumulative statistics will be collected: *S*, the sum of customer response times for all customers who have departed by the current time; *F*, the total number of customers who spend 5 or more minutes at the checkout counter; and N_D, the total number of departures up to the current simulation time. These three

Table 3.2 Simulation Table for Example 3.4

Clock	System State LQ(t)	System State LS(t)	CHECKOUT LINE	Future Event List	Cumulative Statistics S	Cumulative Statistics N_D	Cumulative Statistics F
0	0	1	(C1,0)	(A,1,C2) (D,4, C1) (E,60)	0	0	0
1	1	1	(C1,0) (C2,1)	(A,2,C3) (D,4,C1) (E,60)	0	0	0
2	2	1	(C1,0) (C2,1) (C3,2)	(D,4,C1) (A,8,C4) (E,60)	0	0	0
4	1	1	(C2,1) (C3,2)	(D,6,C2) (A,8,C4) (E, 60)	4	1	0
6	0	1	(C3,2)	(A,8,C4) (D,11,C3) (E,60)	9	2	1
8	1	1	(C3,2) (C4,8)	(D,11,C3) (A,11,C5) (E,60)	9	2	1
11	1	1	(C4,8) (C5,11)	(D,15,C4) (A,18,C6) (E,60)	18	3	2
15	0	1	(C5, 11)	(D,16,C5) (A,18,C6) (E,60)	25	4	3
16	0	0		(A,18,C6) (E,60)	30	5	4
18	0	1	(C6,18)	(D,23,C6) (A,23,C7) (E,60)	30	5	4
23	0	1	(C7,23)	(A,25,C8) (D,27,C7) (E,60)	35	6	5

cumulative statistics will be updated whenever the departure event occurs; the logic for collecting these statistics would be incorporated into step 5 of the departure event in Figure 3.6.

The simulation table for Example 3.4 is shown in Table 3.2. The same data for interarrival and service times will be used again; so Table 3.2 essentially repeats Table 3.1, except that the new components are included (and the comment column has been deleted). These new components are needed for the computation of the cumulative statistics *S*, *F*, and N_D. For example, at time 4, a departure event occurs for customer *C*1. The customer entity *C*1 is removed from the list called "CHECKOUT LINE"; the attribute "time of arrival" is noted to be 0, so the response time for this customer was 4 minutes. Hence, *S* is incremented by 4 minutes. N_D is incremented by one customer, but *F* is not incremented, for the time in system was less than five minutes. Similarly, at time 23, when the departure event (*D*, 23, *C*6) is being executed, the response time for customer *C*6 is computed by

$$\begin{aligned}\text{Response time} &= \text{CLOCK TIME} - \text{attribute "time of arrival"}\\ &= 23 - 18\\ &= 5 \text{ minutes}\end{aligned}$$

Then *S* is incremented by 5 minutes, and *F* and N_D by one customer.

For a simulation run length of 23 minutes, the average response time was $S/N_D = 35/6 = 5.83$ minutes, and the observed proportion of customers who spent 5 or more minutes in the system was $F/N_D = 0.83$. Again, this simulation was far too short to regard these estimates as having any degree of accuracy. The purpose of Example 3.4, however, was to illustrate the notion that, in many simulation models, the information desired from the simulation (such as the statistics S/N_D and F/N_D) to some extent dictates the structure of the model.

Example 3.5: The Dump-Truck Problem

Six dump trucks are used to haul coal from the entrance of a small mine to the railroad. Figure 3.7 provides a schematic of the dump-truck operation. Each truck is loaded by one of two loaders. After a loading, the truck immediately moves to the scale, to be weighed as soon as possible. Both the loaders and the scale have a first-come–first-served waiting line (or queue) for trucks. Travel time from a loader to the scale is considered negligible. After being weighed, a truck begins a travel time (during which time the truck unloads) and then afterward returns to the loader queue. The distributions of loading time, weighing time, and travel time are given in Tables 3.3, 3.4, and 3.5, respectively, together with the random digit assignment for generating

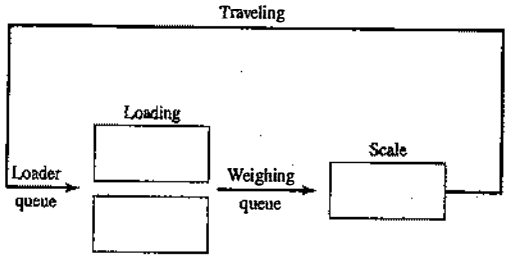


Figure 3.7 Dump truck problem.

Table 3.3 Distribution of Loading Time for the Dump Trucks

Loading Time	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
5	0.30	0.30	1–3
10	0.50	0.80	4–8
15	0.20	1.00	9–0

Table 3.4 Distribution of Weighing Time for the Dump Trucks

Weighing Time	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
12	0.70	0.70	1–7
16	0.30	1.00	8–0

Table 3.5 Distribution of Travel Time for the Dump Trucks

Travel Time	*Probability*	*Cumulative Probability*	*Random Digit Assignment*
40	0.40	0.40	1–4
60	0.30	0.70	5–7
80	0.20	0.90	8–9
100	0.10	1.00	0

these variables by using random digits from Table A.1. The purpose of the simulation is to estimate the loader and scale utilizations (percentage of time busy). The model has the following components:

System state $[LQ(t), L(t), WQ(t), W(t)]$, where
$LQ(t)$ = number of trucks in loader queue
$L(t)$ = number of trucks (0, 1, or 2) being loaded
$WQ(t)$ = number of trucks in weigh queue
$W(t)$ = number of trucks (0 or 1) being weighed, all at simulation time t

Event notices
(ALQ, t, DTi), dump truck i arrives at loader queue (ALQ) at time t
(EL, t, DTi), dump truck i ends loading (EL) at time t
(EW, t, DTi), dump truck i ends weighing (EW) at time t

Entities The six dump trucks ($DT1$, ..., $DT6$)

Lists
Loader queue, all trucks waiting to begin loading, ordered on a first-come–first-served basis
Weigh queue, all trucks waiting to be weighed, ordered on a first-come–first-served basis

Activities Loading time, weighing time, and travel time

Delays Delay at loader queue, and delay at scale

The simulation table is given in Table 3.6. It has been assumed that five of the trucks are at the loaders and one is at the scale at time 0. The activity times are taken from the following list as needed:

Loading Time	10	5	5	10	15	10	10
Weighing Times	12	12	12	16	12	16	
Travel Times	60	100	40	40	80		

When an end-loading (EL) event occurs, say for truck j at time t, other events might be triggered. If the scale is idle $[W(t) = 0]$, truck j begins weighing and an end-weighing event (EW) is scheduled on the FEL; otherwise, truck j joins the weigh queue. If, at this time, there is another truck waiting for a loader, it will be removed from the loader queue and will begin loading by the scheduling of an *end-loading* event (EL) on the FEL. Both this logic for the occurrence of the end-loading event and the appropriate logic for the other two events, should be incorporated into an event diagram, as in Figures 3.5 and 3.6 of Example 3.3. The construction of these event-logic diagrams is left as an exercise for the reader (Exercise 2).

As an aid to the reader, in Table 3.6, whenever a new event is scheduled, its event time is written as "t + (activity time)." For example, at time 0, the imminent event is an EL event with event time 5. The clock is advanced to time $t = 5$, dump truck 3 joins the weigh queue (because the scale is occupied), and truck 4 begins to load. Thus, an EL event is scheduled for truck 4 at future time 10, computed by (present time) + (loading time) = 5 + 5 = 10.

In order to estimate the loader and scale utilizations, two cumulative statistics are maintained:

B_L = total busy time of both loaders from time to time t
B_S = total busy time of the scale from 0 to time t

Both loaders are busy from time 0 to time 20, so $B_L = 40$ at time $t = 20$—but, from time 20 to time 24, only one loader is busy; thus, B_L increases by only 4 minutes over the time interval [20, 24]. Similarly, from time 25 to time 36, both loaders are idle ($L(25) = 0$), so B_L does not change. For the relatively short simulation in Table 3.6, the utilizations are estimated as follows:

$$\text{Average loader utilization} = \frac{49/2}{76} = 0.32$$

$$\text{Average scale utilization} = \frac{76}{76} = 1.00$$

Table 3.6 Simulation Table for Dump-Truck Operation (Example 3.5)

Clock	System State				Lists		Future Event	Cumulative Statistics	
					Loader	Weigh			
t	*LQ(t)*	*L(t)*	*WQ(t)*	*W(t)*	*Queue*	*Queue*	*List*	B_L	B_S
0	3	2	0	1	DT4 DT5 DT6		(EL, 5, DT3) (EL, 10, DT2) (EW, 12, DT1)	0	0
5	2	2	1	1	DT5 DT6	DT3	(EL, 10, DT2) (EL, 5 + 5, DT4) (EW, 12, DT1)	10	5
10	1	2	2	1	DT6	DT3 DT2	(EL, 10, DT4) (EW, 12, DT1) (EL, 10 + 10, DT5)	20	10
10	0	2	3	1		DT3 DT2 DT4	(EW, 12, DT1) (EL, 20, DT5) (EL, 10 + 15, DT6)	20	10
12	0	2	2	1		DT2 DT4	(EL, 20, DT5) (EW, 12 + 12, DT3) (EL, 25, DT6) (ALQ, 12 + 60, DT1)	24	12
20	0	1	3	1		DT2 DT4 DT5	(EW, 24, DT3) (EL, 25, DT6) (ALQ, 72, DT1)	40	20
24	0	1	2	1		DT4 DT5	(EL, 25, DT6) (EW, 24 + 12, DT2) (ALQ, 72, DT1) (ALQ, 24 + 100, DT3)	44	24
25	0	0	3	1		DT4 DT5 DT6	(EW, 36, DT2) (ALQ, 72, DT1) (ALQ, 124, DT3)	45	25
36	0	0	2	1		DT5 DT6	(EW, 36 + 16, DT4) (ALQ, 72, DT1) (ALQ, 36 + 40, DT2) (ALQ, 124, DT3)	45	36
52	0	0	1	1		DT6	(EW, 52 + 12, DT5) (ALQ, 72, DT1) (ALQ, 76, DT2) (ALQ, 52 + 40, DT4) (ALQ, 124, DT3)	45	52
64	0	0	0	1			(ALQ, 72, DT1) (ALQ, 76, DT2) (EW, 64 + 16, DT6) (ALQ, 92, DT4) (ALQ, 124, DT3) (ALQ, 64 + 80, DT5)	45	64

(*continued overleaf*)

Table 3.6 (*continued*)

Clock	System State				Lists		Future Event	Cumulative Statistics	
					Loader	Weigh			
t	*LQ(t)*	*L(t)*	*WQ(t)*	*W(t)*	*Queue*	*Queue*	*List*	B_L	B_S
72	0	1	0	1			(ALQ, 76, DT2) (EW, 80, DT6) (EL, 72 + 10, DT1) (ALQ, 92, DT4) (ALQ, 124, DT3) (ALQ, 144, DT5)	45	72
76	0	2	0	1			(EW, 80, DT6) (EL, 82, DT1) (EL, 76 + 10, DT2) (ALQ, 92, DT4) (ALQ, 124, DT3) (ALQ, 144, DT5)	49	76

These estimates cannot be regarded as accurate estimates of the long-run "steady-state" utilizations of the loader and scale; a considerably longer simulation would be needed to reduce the effect of the assumed conditions at time 0 (five of the six trucks at the loaders) and to realize accurate estimates. On the other hand, if the analyst were interested in the so-called transient behavior of the system over a short period of time (say 1 or 2 hours), given the specified initial conditions, then the results in Table 3.6 can be considered representative (or constituting one sample) of that transient behavior. Additional samples can be obtained by conducting additional simulations, each one having the same initial conditions but using a different stream of random digits to generate the activity times.

Table 3.6, the simulation table for the dump-truck operation, could have been simplified somewhat by not explicitly modeling the dump trucks as entities—that is, the event notices could be written as (*EL*, *t*), and so on, and the state variables used to keep track merely of the number of trucks in each part of the system, not which trucks were involved. With this representation, the same utilization statistics could be collected. On the other hand, if mean "system" response time, or proportion of trucks spending more than 30 minutes in the "system," were being estimated, where "system" refers to the loader queue and loaders and the weigh queue and scale, then dump truck entities (*DT*), together with an attribute equal to arrival time at the loader queue, would be indispensable. Whenever a truck left the scale, that truck's response time could be computed as current simulation time (*t*) minus the arrival-time attribute. This new response time would be used to update the cumulative statistics: S = total response time of all trucks that have been through the "system" and F = number of truck response times that have been greater than 30 minutes. This example again illustrates the notion that, to some extent, the complexity of the model depends on the performance measures being estimated.

Example 3.6: The Dump-Truck Problem Revisited

The events and activities were identified in Example 3.5. Under the activity-scanning approach, the conditions for beginning each activity are as follows:

Activity	Condition
Loading time	Truck is at front of loader queue, and at least one loader is idle.
Weighing time	Truck is at front of weigh queue, and weigh scale is idle.
Travel time	Truck has just completed a weighing.

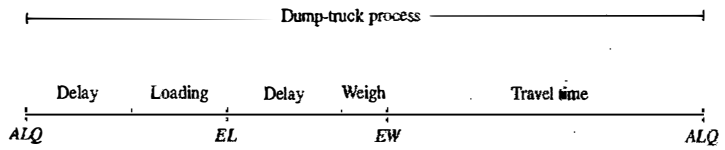


Figure 3.8 The dump-truck process.

Using the process-interaction approach, we view the model from the viewpoint of one dump truck and its "life cycle." Considering a life cycle as beginning at the loader queue, we can picture a dump-truck process as in Figure 3.8.

3.2 LIST PROCESSING

List processing deals with methods for handling lists of entities and the future event list. Simulation packages provide, both explicitly for an analyst's use and covertly in the simulation mechanism behind the language, facilities for an analyst or the model itself to use lists and to perform the basic operations on lists.

Section 3.2.1 describes the basic properties and operations performed on lists. Section 3.2.2 discusses the use of arrays for processing lists and the use of array indices to create linked lists, arrays being a simpler mechanism for describing the basic operations than the more general dynamically allocated linked lists discussed in Section 3.2.3. Finally, Section 3.2.4 briefly introduces some of the more advanced techniques for managing lists.

The purpose of this discussion of list processing is not to prepare the reader to implement lists and their processing in a general-purpose language such as FORTRAN, C++, or Java, but rather to increase the reader's understanding of lists and of their underlying concepts and operations.

3.2.1 Lists: Basic Properties and Operations

As has previously been discussed, lists are a set of ordered or ranked records. In simulation, each record represents one entity or one event notice.

Lists are ranked, so they have a *top* or *head* (the first item on the list); some way to traverse the list (to find the second, third, etc. items on the list); and a *bottom* or *tail* (the last item on the list). A head pointer is a variable that points to or indicates the record at the top of the list. Some implementations of lists also have a tail pointer that points to the bottom item on the list.

For purposes of discussion, an entity, along with its attributes or an event notice, will be referred to as a *record.* An entity identifier and its attributes are *fields* in the entity record; the event type, event time, and any other event-related data are fields in the event-notice record. Each record on a list will also have a field that holds a "next pointer" that points to the next record on the list, providing a way to traverse the list. Some lists also require a "previous pointer," to allow for traversing the list from bottom to top.

For either type of list, the main activities in list processing are adding a record to a list and removing a record from a list. More specifically, the main operations on a list are the following:

1. removing a record from the top of the list;
2. removing a record from any location on the list;
3. adding an entity record to the top or bottom of the list;
4. adding a record at an arbitrary position in the list, one specified by the ranking rule.

The first and third operations, removing or adding a record to the top or bottom of the list, can be carried out in minimal time by adjusting two record pointers and the head or tail pointer, the other two operations require at least a partial search through the list. Making these two operations efficient is the goal of list-processing techniques.

In the event-scheduling approach, when time is advanced and the imminent event is due to be executed, the removal operation takes place first—namely, the event at the top of the FEL is removed from the FEL. If an arbitrary event is being canceled, or an entity is removed from a list based on some of its attributes (say, for example, its priority and due date) to begin an activity, then the second removal operation is performed. When an entity joins the back of a first-in–first-out queue implemented as a list, then the third operation, adding an entity to the bottom of a list, is performed. Finally, if a queue has the ranking rule *earliest due date first*, then, upon arrival at the queue, an entity must be added to the list not at the top or bottom, but at the position determined by the due-date ranking rule.

For simulation on a computer, whether via a general-purpose language (such as FORTRAN, C++, or Java) or via a simulation package, each entity record and event notice is stored in a physical location in computer memory. There are two basic possibilities: (a) All records are stored in arrays. Arrays hold successive records in contiguous locations in computer memory. They therefore can be referenced by an array index that can be thought of as a row number in a matrix. (b) All entities and event notices are represented by structures (as in C) or classes (as in Java), allocated from RAM memory as needed, and tracked by pointers to a record or structure.

Most simulation packages use dynamically allocated records and pointers to keep track of lists of items, as arrays are conceptually simpler, so the concept of linked lists is first explained through arrays and array indices in Section 3.2.2 and then applied to dynamically allocated records and pointers in Section 3.2.3.

3.2.2 Using Arrays for List Processing

The array method of list storage is typical of FORTRAN, but it may be used in other procedural languages. Most versions of FORTRAN do not have actual record-type data structures, but a record may be implemented as a row in a 2-dimensional array or as a number of parallel arrays. For convenience, we use the notation R(i) to refer to the i-th record in the array, however it may be stored in the language being used. Most modern simulation packages do not use arrays for list storage, but rather use dynamically allocated records—that is, records that are created upon first being needed and destroyed when they are no longer needed.

Arrays are advantageous in that any specified record, say the i-th, can be retrieved quickly without searching, merely by referencing R(i). Arrays are disadvantaged when items are added to the middle of a list or the list must be rearranged. In addition, arrays typically have a fixed size, determined at compile time or upon initial allocation when a program first begins to execute. In simulation, the maximum number of records for any list could be difficult (or impossible) to predict, and the current number of them in a list may vary widely over the course of the simulation run. Worse yet, most simulations require more than one list; if they are kept in separate arrays, each would have to be dimensioned to the largest the list would ever be, potentially using excessive amounts of computer memory.

In the use of arrays for storing lists, there are two basic methods for keeping track of the ranking of records in a list. One method is to store the first record in R(1), the second in R(2), and so on, and the last in R(*tailptr*), where *tailptr* is used to refer to the last item in the list. Although simple in concept and easy to understand, this method will be extremely inefficient for all except the shortest lists, those of less than five or so records, for adding an item, for example, in position 41 in a list of 100 items, will require that the last 60 records be physically moved down one array position to make space for the new record. Even if the list were a first-in–first-out list, removing the top item from the list would be inefficient, as all remaining items would have to be physically moved up one position in the array. The physical rearrangement method of managing lists will not be discussed further. What is needed is a method to track and rearrange the logical ordering of items in a list without having to move the records physically in computer memory.

In the second method, a variable called a head pointer, with name *headptr*, points to the record at the top of the list. For example, if the record in position R(11) were the record at the top of the list, then *headptr* would have the value 11. In addition, each record has a field that stores the index or pointer of the next record in the list. For convenience, let R(i, next) represent the next index field.

Example 3.7: A List for the Dump Trucks at the Weigh Queue

In Example 3.5, the dump-truck problem, suppose that a waiting line of three dump trucks occurred at the weigh queue, specifically, DT3, DT2, and DT4, in that order, at exactly CLOCK time 10 in Table 3.6. Suppose further that the model is tracking one attribute of each dump truck: its arrival time at the weigh queue, updated each time it arrives. Finally, suppose that the entities are stored in records in an array dimensioned from 1 to 6, one record for each dump truck. Each entity is represented by a record with 3 fields: the first is an entity identifier; the second is the arrival time at the weigh queue; the last is a pointer field to "point to" the next record, if any, in the list representing the weigh queue, as follows:

[DTi, arrival time at weigh queue, next index]

Before its first arrival at the weigh queue, and before being added to the weigh queue list, a dump truck's second and third fields are meaningless. At time 0, the records would be initialized as follows:

R(1) = [DT1, 0.0, 0]
R(2) = [DT2, 0.0, 0]
⋮
R(6) = [DT6, 0.0, 0]

Then, at CLOCK time 10 in the simulation in Table 3.6, the list of entities in the weigh queue would be defined by

headptr = 3
R(1) = [DT1, 0.0, 0]
R(2) = [DT2, 10.0, 4]
R(3) = [DT3, 5.0, 2]
R(4) = [DT4, 10.0, 0]
R(5) = [DT5, 0.0, 0]
R(6) = [DT6, 0.0, 0]
tailptr = 4

To traverse the list, start with the head pointer, go to that record, retrieve that record's next pointer, and proceed, to create the list in its logical order—for example,

headptr = 3
R(3) = [DT3, 5.0, 2]
R(2) = [DT2, 10.0, 4]
R(4) = [DT4, 10.0, 0]

The zero entry for next pointer in R(4), as well as tailptr = 4, indicates that DT4 is at the end of the list.

Using next pointers for a first-in-first-out list, such as the weigh queue in this example, makes the operations of adding and removing entity records, as dump trucks join and leave the weigh queue, particularly simple. At CLOCK time 12, dump truck DT3 begins weighing and thus leaves the weigh queue. To remove the DT3 entity record from the top of the list, update the head pointer by setting it equal to the next pointer value of the record at the top of the list:

headptr = R(headptr, next)

In this example, we get

headptr = R(3, next) = 2

meaning that dump truck DT2 in R(2) is now at the top of the list.

Similarly, at CLOCK time 20, dump truck DT5 arrives at the weigh queue and joins the rear of the queue. To add the DT5 entity record to the bottom of the list, the following steps are taken:

R(tailptr, next) = 5 (update the next pointer field of the previously last item)
tailptr = 5 (update the value of the tail pointer)

This approach becomes slightly more complex when a list is a ranked list, such as the future event list, or an entity list ranked by an entity attribute. For ranked lists, to add or remove an item anywhere except to the head or tail of the list, searching is usually required. See Example 3.8.

Note that, in the dump-truck problem, the loader queue could also be implemented as a list using the same six records and the same array, because each dump-truck entity will be on at most one of the two lists and, while loading, weighing or traveling, a dump truck will be on neither list.

3.2.3 Using Dynamic Allocation and Linked Lists

In procedural languages, such as C++ and Java, and in most simulation languages, entity records are dynamically created when an entity is created and event notice records are dynamically created whenever an event is scheduled on the future event list. The languages themselves, or the operating systems on which they are running, maintain a linked list of free chunks of computer memory and allocate a chunk of desired size upon request to running programs. (Another use of linked lists!) When an entity "dies," that is, exits from the simulated system, and also after an event occurs and the event notice is no longer needed, the corresponding records are freed, making that chunk of computer memory available for later reuse; the language or operating system adds the chunk to the list of free memory.

In this text, we are not concerned with the details of allocating and freeing computer memory, so we will assume that the necessary operations occur as needed. With dynamic allocation, a record is referenced by a pointer instead of by an array index. When a record is allocated in C++ or Java, the allocation routine returns a pointer to the allocated record, which must be stored in a variable or a field of another record for later use. A pointer to a record can be thought of as the physical or logical address in computer memory of the record.

In our example, we will use a notation for records identical to that in the previous section (3.2.2):

Entities: [ID, attributes, next pointer]
Event notices: [event type, event time, other data, next pointer]

but we will not reference them by the array notation R(i) as before, because it would be misleading. If for some reason we wanted the third item on the list, we would have to traverse the list, counting items until we reached the third record. Unlike in arrays, there is no way to retrieve the i-th record in a linked list directly, because the actual records could be stored at any arbitrary location in computer memory and are not stored contiguously, as arrays are.

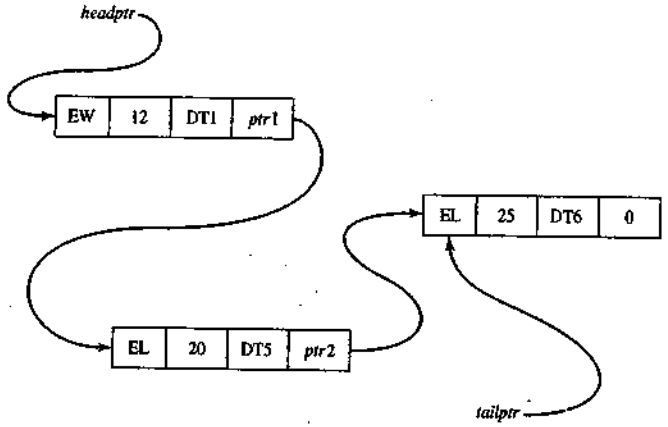


Figure 3.9 The dump-truck future event list as a linked list.

Example 3.8: The Future Event List and the Dump-Truck Problem

Beginning from Table 3.6, event notices in the dump-truck problem of Example 3.5 are expanded to include a pointer to the next event notice on the future event list and can be represented by

[event type, event time, DT*i*, nextptr],

—for example,

[EL, 10, DT3, nextptr]

where EL is the end-loading event to occur at future time 10 for dump truck DT3, and the field *nextptr* points to the next record on the FEL. Keep in mind that the records may be stored anywhere in computer memory, and in particular are not necessarily stored contiguously. Figure 3.9 represents the future event list at CLOCK time 10, taken from Table 3.6. The fourth field in each record is a pointer value to the next record in the future event list.

The C++ and Java languages, and other general-purpose languages, use different notation for referencing data from pointer variables. For discussion purposes, if R is a pointer to a record, then

R→eventtype, R→eventtime, R→next

are the event type, the event time and the next record for the event notice that R points to. For example, if R is set equal to the head pointer for the FEL at CLOCK time 10, then

R→eventtype = EW
R→eventtime = 12
R→next is the pointer for the second event notice on the FEL,

so that

R→next→eventtype = EL
R→next→eventtime = 20
R→next→next is the pointer to the third event notice on the FEL

If one of the pointer fields is zero (or null), then that record is the last item in the list, and the pointer variable *tailptr* points to that last record, as depicted in Figure 3.9.

What we have described are called singly-linked lists, because there is a one-way linkage from the head of the list to its tail. The tail pointer is kept mostly for convenience and efficiency, especially for lists for which items are added at the bottom of the list. Of course, a tail pointer is not strictly necessary, because the last item can always be found by traversing the list, but it does make some operations more efficient.

For some purposes, it is desirable to traverse or search a list by starting at the tail in addition to the head. For such purposes, a doubly-linked list can be used. Records on a doubly-linked list have two pointer fields, one for the next record and one for the previous record. Good references that discuss arrays, singly- and doubly-linked lists, and searching and traversing of lists are Cormen, *et al.* [2001] and Sedgewick [1998].

3.2.4 Advanced Techniques

Many of the modern simulation packages use techniques and representations of lists that are more efficient than searching through a doubly-linked list. Most of these topics are too advanced for this text. The purpose of this section is to introduce some of the more advanced ideas briefly.

One idea to speed up processing or doubly-linked lists is to use a middle pointer in addition to a head and tail pointer. With special techniques, the *mid* pointer will always point to the approximate middle of the list. Then, when a new record is being added to the list, the algorithm first examines the middle record to decide whether to begin searching at the head of the list or the middle of the list. Theoretically, except for some overhead due to maintenance of the mid pointer, this technique should cut search times in half. A few advanced techniques use one or more mid pointers, depending on the length of the list.

Some advanced algorithms use list representations other than a doubly-linked list, such as heaps or trees. These topics are beyond the scope of this text. Good references are Cormen, *et al.* [2001] and Sedgewick [1998].

3.3 SUMMARY

This chapter introduced the major concepts and building blocks in simulation, the most important being entities and attributes, events, and activities. The three major world views—event scheduling, process interaction, activity scanning—were discussed. Finally, to gain an understanding of one of the most important underlying methodologies, Section 3.2 introduced some of the basic notions of list processing.

The next chapter will provide a survey of some of the most widely used and popular simulation packages, most of which either use exclusively or allow the process-interaction approach to simulation.

REFERENCES

CORMEN, T. H., C. E. LEISEROON, AND R. L. RIVEST [2001], *Introduction to Algorithms*, 2nd ed., McGraw–Hill, New York.

SCHRIBER, T. J., AND D. T. BRUNNER [2003], "Inside Simulation Software: How it Works and why it Matters," *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., New Orleans., LA, Dec. 7–10, pp. 113–123.

SEDGEWICK, R. [1998], *Algorithms in C++*, 3d ed., Addison–Wesley, Reading, MA.

EXERCISES

Instructions to the reader: For most exercises, the reader should first construct a model by explicitly defining the following:

1. system state;
2. system entities and their attributes;
3. sets, and the entities that may be put into the sets;
4. events and activities; event notices;
5. variables needed to collect cumulative statistics.

Second, the reader should either (1) develop the event logic (as in Figures 3.5 and 3.6 for Example 3.3) in preparation for using the event-scheduling approach, or (2) develop the system processes (as in Figure 3.4) in preparation for using the process-interaction approach.

Most problems contain activities that are uniformly distributed over an interval $[a, b]$. When conducting a manual simulation, assume that $a, a+1, a+2, \ldots, b$ are the only possible values; that is, the activity time is a *discrete* random variable. The discreteness assumption will simplify the manual simulation.

1. (a) Using the event-scheduling approach, continue the (manual) checkout-counter simulation in Example 3.3, Table 3.1. Use the same interarrival and service times that were previously generated and used in Example 2.1. When the last interarrival time is used, continue the simulation until time 60 using the data in Tables 2.8 and 2.9.

   (b) Do exercise 1(a) again, adding the model components necessary to estimate mean response time and proportion of customers who spend 5 or more minutes in the system. (*Hint:* See Example 3.4, Table 3.2.)

   (c) Comment on the relative merits of manual versus computerized simulations.

2. Give the detailed flow chart for the simulation of a single-server queueing system.

3. In the dump-truck problem of Example 3.5, it is desired to estimate mean response time and the proportion of response times which are greater than 30 minutes. A response time for a truck begins when that truck arrives at the loader queue and ends when the truck finishes weighing. Add the model components and cumulative statistics needed to estimate these two measures of system performance. Simulate for 8 hours.

4. Consider a single-server queueing system with arrival and service details as:
   Interarrival times 3 2 6 2 4 5
   Service times 2 5 5 8 4 5
   Prepare a table similar to Table 3.2 for the given data. Stop simulation when the clock reaches 20.

5. Continue the table that is prepared in Exercise 4 till C5 leaves the system.

6. The data for Table 3.2 are changed to the following:
   Interarrival times 4 5 2 8 3 7
   Service times 5 3 4 6 2 7
   Prepare a table in the manner of Table 3.2 with a stopping event at time 25.

7. Redo Example 2.2 (the Able–Baker call center problem) by a manual simulation, using the event-scheduling approach.

8. Redo Example 2.4 (the (M, N) inventory system) by a manual simulation, using the event-scheduling approach.

9. Redo Example 2.5 (the bearing-replacement problem) by a manual simulation, using the event-scheduling approach.

10. Redo Example 3.5 with the following data:
    Loading times 5 10 5 5 10 5 10 10
    Weighing times 12 12 12 16 16 12 12 16
    Travel times 80 80 100 40 100 40 60 40

4

Simulation Software

In this chapter, we first discuss the history of simulation software. Simulation software has a history that is just reaching middle age. We base this history on our collective experience, articles written by Professor Richard Nance, and panel discussions at the annual Winter Simulation Conference.

Next, we discuss features and attributes of simulation software. If you were about to purchase simulation software, what would concern you? Would it be the cost, the ease of learning, the ease of use, or would it be the power to model the kind of system with which you are concerned? Or would it be the animation capabilities? Following the discussion of features, we discuss other issues and concerns related to the selection of simulation software.

Software used to develop simulation models can be divided into three categories. First, there are the general-purpose programming languages, such as C, C++, and Java. Second, there are simulation programming languages, examples being GPSS/H™, SIMAN V®and SLAM II®. Third, there are the simulation environments. This category includes many products that are distinguished one way or another (by, for example, cost, application area, or type of animation), but have common characteristics, such as a graphical user interface and an environment that supports all (or most) aspects of a simulation study. Many simulation environments contain a simulation programming language, but some take a graphical approach similar to process-flow diagramming.

In the first category, we discuss simulation in Java. Java is a general-purpose programming language that was not specifically designed for use in simulation. Java was chosen since it is widely used and widely available. Today very few people writing discrete-event simulation models are using programming languages alone; however, in certain application areas, some people are using packages based on Java or on another general-purpose language. Understanding how to develop a model in a general-purpose language helps to understand how the basic concepts and algorithms discussed in Chapter 3 are implemented.

In the second category, we discuss GPSS/H, a highly structured process-interaction simulation language. GPSS was designed for relatively easy simulation of queuing systems, such as in job shops, but it

has been used to simulate systems of great complexity. It was first introduced by IBM; today, there are various implementations of GPSS, GPSS/H being one of the most widely used.

In the third category, we have selected a number of simulation software packages for discussion. There are many simulation packages currently available; we have selected a few that have survived and thriven for a number of years, to represent different approaches for model-building.

One of the important components of a simulation environment is the output analyzer, which is used to conduct experimentation and assist with analyses. To illustrate the range of desirable characteristics, we look at four tools incorporated into some of the simulation environments. Typically these statistical analysis tools compute summary statistics, confidence intervals, and other statistical measures. Some support warmup determination, design of experiments, and sensitivity analyses. Many packages now offer optimization techniques based on genetic algorithms, evolutionary strategies, tabu search, scatter search, and other recently developed heuristic methods. In addition to the support for statistical analysis and optimization, the simulation environments offer data management, scenario definition, and run management. Data management offers support for managing all the input and output data associated with the analyses.

4.1 HISTORY OF SIMULATION SOFTWARE

Our discussion of the history of simulation software is based on Nance [1995], who breaks the years from 1955 through 1986 into five periods. Additional historical information is taken from a panel discussion at the 1992 Winter Simulation Conference entitled "Perspectives of the Founding Fathers" [Wilson, 1992], during which eight early users of simulation presented their historical perspectives. We add a sixth and most recent period:

1955–60	The Period of Search
1961–65	The Advent
1966–70	The Formative Period
1971–78	The Expansion Period
1979–86	The Period of Consolidation and Regeneration
1987–?	The Period of Integrated Environments

The following subsections provide a brief presentation of this history. As indicated in [Nance, 1995], there were at least 137 simulation programming languages reported as of 1981, and many more since then. This brief history is far from all-inclusive. The languages and packages we mention have stood the test of time by surviving to the present day or were the historical forerunner of a package in present use.

4.1.1 The Period of Search (1955–60)

In the early years, simulation was conducted in FORTRAN or other general-purpose programming language, without the support of simulation-specific routines. In the first period (1955–1960), much effort was expended on the search for unifying concepts and the development of reusable routines to facilitate simulation. The General Simulation Program of K.D. Tocher and D.G. Owen [Tocher, 1960] is considered the first "language effort." Tocher identified and developed routines that could be reused in subsequent simulation projects.

4.1.2 The Advent (1961–65)

The forerunners of the simulation programming languages (SPLs) in use today appeared in the period 1961–65. As Harold Hixson said in [Wilson, 1992], "in the beginning there were FORTRAN, ALGOL, and GPSS"— that is, there were the FORTRAN-based packages (such as SIMSCRIPT and GASP), the ALGOL descendent SIMULA, and GPSS.

The first process interaction SPL, GPSS was developed by Geoffrey Gordon at IBM and appeared in about 1961. Gordon developed GPSS (General Purpose Simulation System) for quick simulations of communications and computer systems, but its ease of use quickly spread its popularity to other application areas. GPSS is based on a block-diagram representation (similar to a process-flow diagram) and is suited for queuing models of all kinds. As reported by Reitman in [Wilson, 1992], as early as 1965 GPSS was connected to an interactive display terminal that could interrupt and display intermediate results, a foreshadow of the interactive simulations of today, but far too expensive at the time to gain widespread use.

Harry Markowitz (later to receive a Nobel Prize for his work in portfolio theory) provided the major conceptual guidance for SIMSCRIPT, first appearing in 1963. The RAND Corporation developed the language under sponsorship of the U.S. Air Force. SIMSCRIPT originally was heavily influenced by FORTRAN, but, in later versions, its developers broke from its FORTRAN base and created its own SPL. The initial versions were based on event scheduling.

Phillip J. Kiviat of the Applied Research Laboratory of the United States Steel Corporation began the development of GASP (General Activity Simulation Program) in 1961. Originally, it was based on the general-purpose programming language ALGOL, but later a decision was made to base it on FORTRAN. GASP, like GPSS, used flowchart symbols familiar to engineers. It was not a language proper, but rather a collection of FORTRAN routines to facilitate simulation in FORTRAN.

Numerous other SPLs were developed during this time period. Notably, they included SIMULA, an extension of ALGOL, developed in Norway and widely used in Europe, and The Control and Simulation Language (CSL), which took an activity-scanning approach.

4.1.3 The Formative Period (1966–70)

During this period, concepts were reviewed and refined to promote a more consistent representation of each language's worldview. The major SPLs matured and gained wider usage.

Rapid hardware advancements and user demands forced some languages, notably GPSS, to undergo major revisions. GPSS/360, with its extensions to earlier versions of GPSS, emerged for the IBM 360 computer. Its popularity motivated at least six other hardware vendors and other groups to produce their own implementation of GPSS or a look-alike.

SIMSCRIPT II represented a major advancement in SPLs. In its free-form English-like language and "forgiving" compiler, an attempt was made to give the user major consideration in the language design.

ECSL, a descendent of CSL, was developed and became popular in the UK. In Europe, SIMULA added the concept of classes and inheritance, thus becoming a precursor of the modern object-oriented programming languages.

4.1.4 The Expansion Period (1971–78)

Major advances in GPSS during this period came from outside IBM. Julian Reitman of Norden Systems headed the development of GPSS/NORDEN, a pioneering effort that offered an interactive, visual online environment. James O. Henriksen of Wolverine Software developed GPSS/H, released in 1977 for IBM mainframes, later for mini-computers and the PC. It was notable for being compiled and reportedly 5 to 30 times faster than standard GPSS. With the addition of new features, including an interactive debugger, it has become the principal version of GPSS in use today.

Alan Pritsker at Purdue made major changes to GASP, with GASP IV appearing in 1974. It incorporated state events in addition to time events, thus adding support for the activity-scanning worldview in addition to the event-scheduling worldview.

Efforts were made during this period to attempt to simplify the modeling process. Using SIMULA, an attempt was made to develop a system definition from a high-level user perspective that could be translated automatically into an executable model. Similar efforts included interactive program generators, the "Programming by Questionnaire," and natural-language interfaces, together with automatic mappings to the language of choice. As did earlier over-optimistic beliefs in automatic programming, these efforts ran into severe limitations in the generality of what could be modeled—that is, they ran into the unavoidable complexity of real-world systems. Nevertheless, efforts to simplify simulation modeling continue, with the most success seen in simulation systems designed for application to narrow domains.

4.1.5 Consolidation and Regeneration (1979–86)

The fifth period saw the beginnings of SPLs written for, or adapted to, desktop computers and the microcomputer. During this period, the predominant SPLs extended their implementation to many computers and microprocessors while maintaining their basic structure.

Two major descendants of GASP appeared: SLAM II and SIMAN. The Simulation Language for Alternative Modeling (SLAM), produced by Pritsker and Associates, Inc., sought to provide multiple modeling perspectives and combined modeling capabilities [Pritsker and Pegden, 1979]—that is, it had an event-scheduling perspective based on GASP, a network worldview (a variant of the process-interaction perspective), and a continuous component. With SLAM, you could select one worldview or use a mix of all three.

SIMAN (SIMulation ANalysis) possessed a general modeling capability found in SPLs such as GASP IV, but also had a block-diagram component similar in some respects to that in SLAM and GPSS. C. Dennis Pegden developed SIMAN as a one-person faculty project over a period of about two years; he later founded Systems Modeling Corporation to market SIMAN. SIMAN was the first major simulation language executable on the IBM PC and designed to run under MS-DOS constraints. Similar to GASP, SIMAN allowed an event-scheduling approach by programming in FORTRAN with a supplied collection of FORTRAN routines, a block-diagram approach (another variant of the process-interaction worldview) analogous in some ways to that of GPSS and SLAM, and a continuous component.

4.1.6 Integrated Environments (1987–Present)

The most recent period is notable by the growth of SPLs on the personal computer and the emergence of simulation environments with graphical user interfaces, animation, and other visualization tools. Many of these environments also contain input-data analyzers and output analyzers. Some packages attempt to simplify the modeling process by the use of process-flow or block diagramming and of "fill-in-the-blank" windows that avoid the need to learn programming syntax. Animation ranges from schematic-like representations to 2-D and 3-D scale drawings.

Recent advancements have been made in web-based simulation. Much discussion has taken place concerning a role for simulation in supply-chain management. The combination of simulation and emulation shows promise.

Information about various software packages is given in Section 4.7, including the websites of the vendors. A view of current developments in simulation software is available from these websites.

4.2 SELECTION OF SIMULATION SOFTWARE

This chapter includes a brief introduction to a number of simulation-software packages. Every two years, OR/MS Today publishes a simulation-software survey [Swain, 2003]. The 2003 issue had 48 products, including simulation support packages such as input-data analyzers.

Table 4.1 Model-Building Features

Feature	*Description*
Modeling worldview	Process interaction, event perspectives, and continuous modeling, depending on needs
Input-data analysis capability	Estimate empirical or statistical distributions from raw data
Graphical model-building	Process-flow, block-diagram, or network approach
Conditional routing	Route entities based on prescribed conditions or attributes
Simulation programming	Capability to add procedural logic through a high-level powerful simulation language
Syntax	Easily understood, consistent, unambiguous, English-like
Input flexibility	Accepts data from external files, data bases, spreadsheets, or interactively
Modeling conciseness	Powerful actions, blocks, or nodes
Randomness	Random-variate generators for all common distributions, e.g., exponential triangular uniform normal
Specialized components and templates	Material handling: vehicles, conveyors, bridge cranes, AS/RS, etc. Handling of liquids and bulk materials Communication systems Computer systems Call centers Others
User-built custom objects	Reusable objects, templates and submodels
Continuous flow	Tanks and pipes or bulk conveyors
Interface with general-programming language	Link code in C, C++, Java, or other general-programming language

Table 4.2 Runtime Environment

Feature	*Description*
Execution speed	Many runs needed for scenarios and replications. Impacts development as well as experimentation
Model size; number of variables and attributes	Should be no built-in limits
Interactive debugger	Monitor the simulation in detail as it progresses. Ability to break, trap, run until, step; to display status, attributes and variables; etc.
Model status and statistics	Display at any time during simulation
Runtime license	Ability to change parameters and run a model (but not to change logic or bnild a new model)

Table 4.3 Animation and Layout Features

Feature	*Description*
Type of animation	True to scale or iconic (such as process-flow diagram)
Import drawing and object files	From CAD (vector formats) drawings or icons (bit-mapped or raster graphics)
Dimension	2-D, 2-D with perspective, 3-D
Movement	Motion of entities or status indicators
Quality of motion	Smooth or jerky
Libraries of common objects	Extensive predrawn graphics
Navigation	Panning, zooming, rotation
Views	User defined, named
Display step	Control of animation speed
Selectable objects	Dynamic status and statistics displayed upon selection
Hardware requirements	Standard or special video card, RAM reqnirements

There are many features that are relevant when selecting simulation software [Banks, 1996]. Some of these features are shown, along with a brief description, in Tables 4.1 to 4.5. We offer the following advice when evaluating and selecting simulation software:

1. Do not focus on a single issue, such as ease of use. Consider the accuracy and level of detail obtainable, ease of learning, vendor support, and applicability to your applications.
2. Execution speed is important. Do not think exclusively in terms of experimental runs that take place at night and over the weekend. Speed affects development time. During debugging, an analyst might have to wait for the model to run up to the point in simulated time where an error occurs many times before the error is identified.
3. Beware of advertising claims and demonstrations. Many advertisements exploit positive features of the software only. Similarly, the demonstrations solve the test problem very well, but perhaps not your problem.
4. Ask the vendor to solve a small version of your problem.
5. Beware of "checklists" with "yes" and "no" as the entries. For example, many packages claim to have a conveyor entity. However, implementations have considerable variation and level of fidelity.

Table 4.4 Output Features

Feature	*Description*
Scenario manager	Create user-defined scenarios to simulate
Run manager	Make all runs (scenarios and replications) and save results for future analyses
Warmup capability	For steady-state analysis
Independent replications	Using a different set of random numbers
Optimization	Genetic algorithms, tabu search, etc.
Standardized reports	Summary reports including averages, counts, minimum and maximum, etc.
Customized reports	Tailored presentations for managers
Statistical analysis	Confidence intervals, designed experiments, etc.
Business graphics	Bar charts, pie charts, time lines, etc.
Costing module	Activity-based costing included
File export	Input to spreadsheet or database for custom processing and analysis
Database maintenance	Store output in an organized manner

Table 4.5 Vendor Support and Product Documentation

Feature	*Description*
Training	Regularly scheduled classes of high quality
Documentation	Quality, completeness, online
Help system	General or context-sensitive
Tutorials	For learning the package or specific features
Support	Telephone, e-mail, web
Upgrades, maintenance	Regularity of new versions and maintenance releases that address customer needs
Track record	Stability, history, customer relations

Implementation and capability are what is important. As a second example, most packages offer a runtime license, but these vary considerably in price and features.

6. Simulation users ask whether the simulation model can link to and use code or routines written in external languages such as C, C++, or Java. This is a good feature, especially when the external routines already exist and are suitable for the purpose at hand. However, the more important question is whether the simulation package and language are sufficiently powerful to avoid having to write logic in any external language.
7. There may be a significant trade-off between the graphical model-building environments and ones based on a simulation language. Graphical model-building removes the learning curve due to language syntax, but it does not remove the need for procedural logic in most real-world models and the debugging to get it right. Beware of "no programming required" unless either the package is a near-perfect fit to your problem domain or programming (customized procedural logic) is possible with the supplied blocks, nodes, or process-flow diagram—in which case "no programming required" refers to syntax only and not the development of procedural logic.

4.3 AN EXAMPLE SIMULATION

Example 4.1: The Checkout Counter: A Typical Single-Server Queue

The system, a grocery checkout counter, is modeled as a single-server queue. The simulation will run until 1000 customers have been served. In addition, assume that the interarrival times of customers are exponentially distributed with a mean of 4.5 minutes, and that the service times are (approximately) normally distributed with a mean of 3.2 minutes and a standard deviation of 0.6 minute. (The approximation is that service times are always positive.) When the cashier is busy, a queue forms with no customers turned away. This example was simulated manually in Examples 3.3 and 3.4 by using the event-scheduling point of view. The model contains two events: the arrival and departure events. Figures 3.5 and 3.6 provide the event logic.

The following three sections illustrate the simulation of this single-server queue in Java, GPSS/H, and SSF. Although this example is much simpler than models that arise in the study of complex systems, its simulation contains the essential components of all discrete-event simulations.

4.4 SIMULATION IN JAVA

Java is a widely used programming language that has been used extensively in simulation. It does not, however, provide any facilities directly aimed at aiding the simulation analyst, who therefore must program all details of the event-scheduling/time-advance algorithm, the statistics-gathering capability, the generation of samples from specified probability distributions, and the report generator. However, the runtime library does provide a random-number generator. Unlike with FORTRAN or C, the object-orientedness of Java does support modular construction of large models. For the most part, the special-purpose simulation languages hide the details of event scheduling, whereas in Java all the details must be explicitly programmed. However, to a certain extent, simulation libraries such as SSF (Cowie 1999) alleviate the development burden by providing access to standardized simulation functionality and by hiding low-level scheduling minutiae.

There are many online resources for learning Java; we assume a prior working knowledge of the language. Any discrete-event simulation model written in Java contains the components discussed in Section 4.3: system state, entities and attributes, sets, events, activities and delays, and the components listed shortly. To facilitate development and debugging, it is best to organize the Java model in a modular fashion by using methods. The following components are common to almost all models written in Java:

Clock A variable defining simulated time
Initialization method A method to define the system state at time 0
Min-time event method A method that identifies the imminent event, that is, the element of the future event list (`FutureEventList`) that has the smallest time-stamp
Event methods For each event type, a method to update system state (and cumulative statistics) when that event occurs
Random-variate generators Methods to generate samples from desired probability distributions
Main program To maintain overall control of the event-scheduling algorithm
Report generator A method that computes summary statistics from cumulative statistics and prints a report at the end of the simulation

The overall structure of a Java simulation program is shown in Figure 4.1. This flowchart is an expansion of the event-scheduling/time-advance algorithm outlined in Figure 3.2. (The steps mentioned in Figure 4.1 refer to the five steps in Figure 3.2.)

The simulation begins by setting the simulation `Clock` to zero, initializing cumulative statistics to zero, generating any initial events (there will always beat least one) and placing them on the `FutureEventList`,

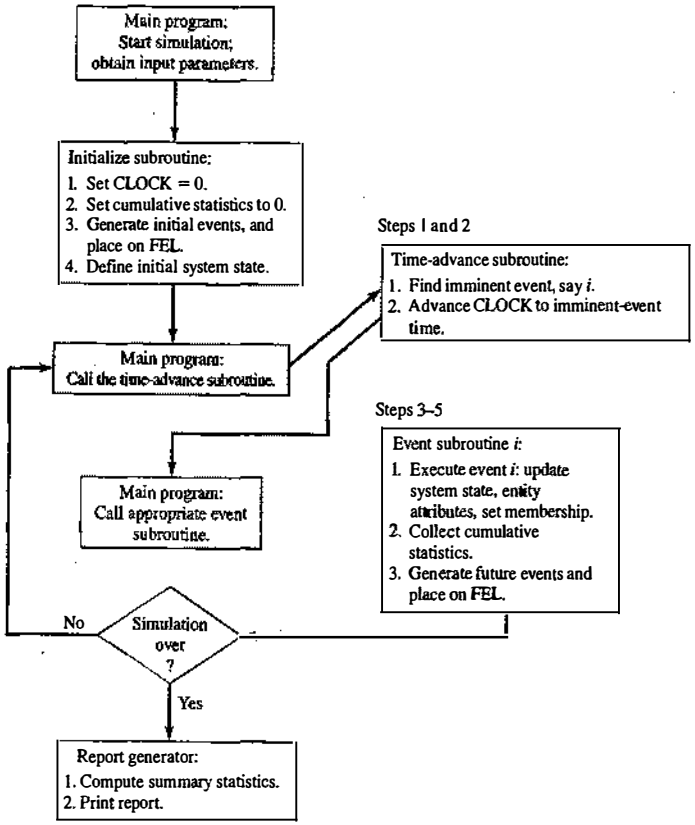


Figure 4.1 Overall structure of an event-scheduling simulation program.

and defining the system state at time 0. The simulation program then cycles, repeatedly passing the current least-time event to the appropriate event methods until the simulation is over. At each step, after finding the imminent event but before calling the event method, the simulation `Clock` is advanced to the time of the imminent event. (Recall that, during the simulated time between the occurrence of two successive events, the system state and entity attributes do not change in value. Indeed, this is the definition of discrete-event simulation: The system state changes only when an event occurs.) Next, the appropriate event method is called to execute the imminent event, update cumulative statistics, and generate future events (to be placed on the `FutureEventList`). Executing the imminent event means that the system state, entity attributes, and set membership are changed to reflect the fact that the event has occurred. Notice that all actions in an event method take place at one instant of simulated time. The value of the variable `Clock` does not change in an event method. If the simulation is not over, control passes again to the time-advance method, then to the appropriate event method, and so on. When the simulation is over, control passes to the report generator, which computes the desired summary statistics from the collected cumulative statistics and prints a report.

The efficiency of a simulation model in terms of computer runtime is determined to a large extent by the techniques used to manipulate the `FutureEventList` and other sets. As discussed earlier in Section 4.3,

removal of the imminent event and addition of a new event are the two main operations performed on the `FutureEventList`. Java includes general, efficient data structures for searching and priority lists; it is usual to build a customized interface to these to suit the application. In the example to follow, we use customized interfaces to implement the event list and the list of waiting customers. The underlying priority-queue organization is efficient, in the sense of having access costs that grow only in the logarithm of the number of elements in the list.

Example 4.2: Single-Server Queue Simulation in Java

The grocery checkout counter, defined in detail in Example 4.1, is now simulated by using Java. A version of this example was simulated manually in Examples 3.3 and 3.4, where the system state, entities and attributes, sets, events, activities, and delays were analyzed and defined.

Class `Event` represents an event. It stores a code for the event type (`arrival` or `departure`), and the event time-stamp. It has associated methods (functions) for creating an event and accessing its data. It also has an associated method **compareTo**, which compares the event with another (passed as an argument) and reports whether the first event should be considered to be smaller, equal, or larger than the argument event. The methods for this model and the flow of control are shown in Figure 4.2, which is an adaptation of Figure 4.1 for this particular problem. Table 4.6 lists the variables used for system state, entity attributes and sets, activity durations, and cumulative and summary statistics; the functions used to generate samples from the exponential and normal distributions; and all the other methods needed.

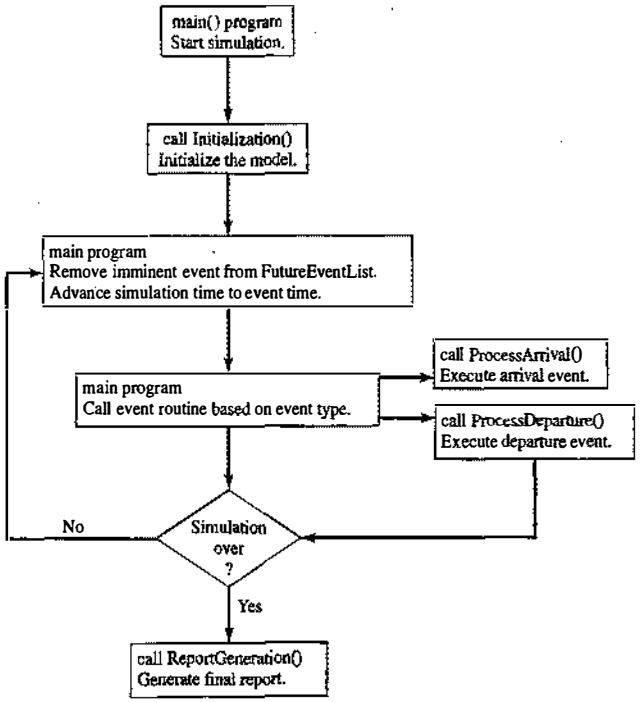


Figure 4.2 Overall structure of Java simulation of a single-server queue.

Table 4.6 Definitions of Variables, Functions, and Subroutines in the Java Model of the Single-Server Queue

Variables	*Description*
System state	
`QueueLength`	Number of customers enqueued (but not in service) at current simulated time
`NumberInService`	Number being served at current simulated time
Entity attributes and sets	
`Customers`	FCFS Queue of customers in system
Future event list	
`FutureEventList`	Priority-ordered list of pending events
Activity durations	
`MeanInterArrivalTime`	The interarrival time between the previous customer's arrival and the next arrival
`MeanServiceTime`	The service time of the most recent customer to begin service
Input parameters	
`MeanInterarrivalTime`	Mean interarrival time (4.5 minutes)
`MeanServiceTime`	Mean service time (3.2 minutes)
`SIGMA`	Standard deviation of service time (0.6 minute)
`TotalCustomers`	The stopping criterion— number of customers to be served (1000)
Simulation variables	
`Clock`	The current value of simulated time
Statistical Accumulators	
`LastEventTime`	Time of occurrence of the last event
`TotalBusy`	Total busy time of server (so far)
`MaxQueueLength`	Maximum length of waiting line (so far)
`SumResponseTime`	Sum of customer response times for all customers who have departed (so far)
`NumberOfDepartures`	Number of departures (so far)
`LongService`	Number of customers who spent 4 or more minutes at the checkout counter (so far)
Summary statistics	
`RHO = BusyTime/Clock`	Proportion of time server is busy (here the value of `Clock` is the final value of simulated time)
`AVGR`	Average response time (equal to `SumResponseTime/TotalCustomers`)
`PC4`	Proportion of customers who spent 4 or more minutes at the checkout counter

Functions	*Description*
`exponential(mu)`	Function to generate samples from an exponential distribution with mean `mu`
`normal(xmu,SIGMA)`	Function to generate samples from a normal distribution with mean `xmu` and standard deviation `SIGMA`

(continued overleaf)

Table 4.6 *(continued)*

Methods	*Description*
`Initialization`	Initialization method
`ProcessArrival`	Event method that executes the arrival event
`ProcessDeparture`	Event method that executes the departure event
`ReportGeneration`	Report generator

The entry point of the program and the location of the control logic is through class `Sim`, shown in Figure 4.3. Variables of classes `EventList` and `Queue` are declared. As these classes are all useful for programs other than `Sim`, their declarations are given in other files, per Java rules. A variable of the Java built-in class `Random` is also declared; instances of this class provided random-number streams. The main method controls the overall flow of the event-scheduling/time-advance algorithm.

```
class Sim {

// Class Sim variables
public static double Clock, MeanInterArrivalTime, MeanServiceTime,
       SIGMA, LastEventTime, TotalBusy, MaxQueueLength, SumResponseTime;
public static long NumberOfCustomers, QueueLength, NumberInService,
       TotalCustomers, NumberOfDepartures, LongService;

public final static int arrival = 1;
public final static int departure = 2;

public static EventList FutureEventList;
public static Queue Customers;
public static Random stream;

public static void main(String argv[]) {

  MeanInterArrivalTime = 4.5; MeanServiceTime = 3.2;
  SIGMA                = 0.6; TotalCustomers  = 1000;
  long seed            = Long.parseLong(argv[0]);
  stream = new Random(seed);            // initialize rng stream
  FutureEventList = new EventList();
  Customers = new Queue();

  Initialization();

  // Loop until first "TotalCustomers" have departed
  while(NumberOfDepartures < TotalCustomers ) {
    Event evt = (Event) FutureEventList.getMin();     // get imminent event
    FutureEventList.dequeue();                        // be rid of it
    Clock = evt.get_time();                           // advance in time
    if( evt.get_type() == arrival ) ProcessArrival(evt);
    else ProcessDeparture(evt);
    }
  ReportGeneration();
}
```

Figure 4.3 Java main program for the single-server queue simulation.

The main program method first gives values to variables describing model parameters; it creates instances of the random-number generator, event list, and customer queue; and then it calls method `Initialization` to initialize other variables, such as the statistics-gathering variables. Control then enters a loop which is exited only after `TotalCustomers` customers have received service. Inside the loop, a copy of the imminent event is obtained by calling the `getMin` method of the priority queue, and then that event is removed from the event list by a call to dequeue. The global simulation time `Clock` is set to the time-stamp contained in the imminent event, and then either `ProcessArrival` or `ProcessDeparture` is called, depending on the type of the event. When the simulation is finally over, a call is made to method `ReportGeneration` to create and print out the final report.

A listing for the `Sim` class method `Initialization` is given in Figure 4.4. The simulation clock, system state, and other variables are initialized. Note that the first arrival event is created by generating a local `Event` variable whose constructor accepts the event's type and time. The event time-stamp is generated randomly by a call to `Sim` class method `exponential` and is passed to the random-number stream to use with the mean of the exponential distribution from which to sample. The event is inserted into the future event list by calling method `enqueue`. This logic assumes that the system is empty at simulated time `Clock=0`, so that no departure can be scheduled. It is straightforward to modify the code to accommodate alternative starting conditions by adding events to `FutureEventList` and `Customers` as needed.

Figure 4.5 gives a listing of `Sim` class method `ProcessArrival`, which is called to process each arrival event. The basic logic of the arrival event for a single-server queue was given in Figure 3.5 (where LQ corresponds to `QueueLength` and LS corresponds to `NumberInService`). First, the new arrival is added to the queue `Customers` of customers in the system. Next, if the server is idle (`NumberInService==0`) then the new customer is to go immediately into service, so `Sim` class method `ScheduleDeparture` is called to do that scheduling. An arrival to an idle queue does not update the cumulative statistics, except possibly the maximum queue length. An arrival to a busy queue does *not* cause the scheduling of a departure, but does increase the total busy time by the amount of simulation time between the current event and the one immediately preceding it (because, if the server is busy now, it had to have had at least one customer in service by the end of processing the previous event). In either case, a new arrival is responsible for scheduling the next arrival, one random interarrival time into the future. An arrival event is created with simulation time equal to the current `Clock` value plus an exponential increment, that event is inserted into the future event list, the variable `LastEventTime` recording the time of the last event processed is set to the current time, and control is returned to the `main` method of class `Sim`.

```
public static void Initialization()  {
  Clock = 0.0;
  QueueLength = 0;
  NumberInService = 0;
  LastEventTime = 0.0;
  TotalBusy = 0 ;
  MaxQueueLength = 0;
  SumResponseTime = 0;
  NumberOfDepartures = 0;
  LongService = 0;

  // create first arrival event
  Event evt =
   new Event(arrival, exponential( stream, MeanInterArrivalTime));
  FutureEventList.enqueue( evt );
}
```

Figure 4.4 Java initialization method for the single-server queue simulation.


```
public static void ProcessArrival(Event evt)  {
  Customers.enqueue(evt);
  QueueLength++;
  // if the server is idle, fetch the event, do statistics
  // and put into service
  if( NumberInService == 0) ScheduleDeparture();
  else TotalBusy += (Clock - LastEventTime);     // server is busy

  // adjust max queue length statistics
  if (MaxQueueLength < QueueLength) MaxQueueLength = QueueLength;

  // schedule the next arrival
  Event next_arrival =
   new Event(arrival, Clock+exponential(stream,MeanInterArrivalTime));
  FutureEventList.enqueue( next_arrival );
  LastEventTime = Clock;
}
```

Figure 4.5 Java arrival event method for the single-server queue simulation.

`Sim` class method `ProcessDeparture`, which executes the departure event, is listed in Figure 4.6, as is method `ScheduleDeparture`. A flowchart for the logic of the departure event was given in Figure 3.6. After removing the event from the queue of all customers, the number in service is examined. If there are customers waiting, then the departure of the next one to enter service is scheduled. Then, cumulative statistics recording the sum of all response times, sum of busy time, number of customers who used more than 4 minutes of service time, and number of departures are updated. (Note that the maximum queue length cannot change in value when a departure occurs.) Notice that customers are removed from `Customers` in

```
public static void ScheduleDeparture()  {
  double ServiceTime;
  // get the job at the head of the queue
  while (( ServiceTime = normal(stream,MeanServiceTime, SIGMA)) < 0 );
  Event depart = new Event(departure,Clock+ServiceTime);
  FutureEventList.enqueue( depart );
  NumberInService = 1;
  QueueLength--;
  }

public static void ProcessDeparture(Event e) {
  // get the customer description
  Event finished = (Event) Customers.dequeue();
  // if there are customers in the queue then schedule
  // the departure of the next one
   if( QueueLength > 0 ) ScheduleDeparture();
   else NumberInService = 0;
   // measure the response time and add to the sum
   double response = (Clock - finished.get_time());
   SumResponseTime += response;
   if( response > 4.0 ) LongService++; // record long service
   TotalBusy += (Clock - LastEventTime );
   NumberOfDepartures++;
   LastEventTime = Clock;
  }
```

Figure 4.6 Java departure event method for the single-server queue simulation.

FIFO order; hence, the response time response of the departing customer can be computed by subtracting the arrival time of the job leaving service (obtained from the copy of the arrival event removed from the Customers queue) from the current simulation time. After the incrementing of the total number of departures and the saving of the time of this event, control is returned to the main program.

Figure 4.6 also gives the logic of method ScheduleDeparture, called by both ProcessArrival and ProcessDeparture to put the next customer into service. The Sim class method normal, which generates normally distributed service times, is called until it produces a nonnegative sample. A new event with type departure is created, with event time equal to the current simulation time plus the service time just sampled. That event is pushed onto FutureEventList, the number in service is set to one, and the number waiting (QueueLength) is decremented to reflect the fact that the customer entering service is waiting no longer.

The report generator, Sim class method ReportGeneration, is listed in Figure 4.7. The summary statistics, RHO, AVGR, and PC4, are computed by the formulas in Table 4.6; then the input parameters are printed, followed by the summary statistics. It is a good idea to print the input parameters at the end of the simulation, in order to verify that their values are correct and that these values have not been inadvertently changed.

Figure 4.8 provides a listing of Sim class methods exponential and normal, used to generate random variates. Both of these functions call method nextDouble, which is defined for the built-in Java Random class generates a random number uniformly distributed on the (0,1) interval. We use Random here for simplicity of explanation; superior random-number generators can be built by hand, as described in Chapter 7.

```
public static void ReportGeneration() {
double RHO  = TotalBusy/Clock;
double AVGR = SumResponseTime/TotalCustomers;
double PC4  = ((double)LongService)/TotalCustomers;


System.out.print( "SINGLE SERVER QUEUE SIMULATION ");
System.out.println( "- GROCERY STORE CHECKOUT COUNTER ");
System.out.println( "\tMEAN INTERARRIVAL TIME                "
   + MeanInterArrivalTime );
System.out.println( "\tMEAN SERVICE TIME                     "
   + MeanServiceTime );
System.out.println( "\tSTANDARD DEVIATION OF SERVICE TIMES   "
   + SIGMA );
System.out.println( "\tNUMBER OF CUSTOMERS SERVED            "
   + TotalCustomers );
System.out.println();
System.out.println( "\tSERVER UTILIZATION                    "
   + RHO );
System.out.println( "\tMAXIMUM LINE LENGTH                   "
   + MaxQueueLength );
System.out.println( "\tAVERAGE RESPONSE TIME                 "
   + AVGR + "  MINUTES" );
System.out.println( "\tPROPORTION WHO SPEND FOUR ");
System.out.println( "\t MINUTES OR MORE IN SYSTEM            "
   + PC4 );
System.out.println( "\tSIMULATION RUNLENGTH                  "
   + Clock + " MINUTES" );
System.out.println( "\tNUMBER OF DEPARTURES                  "
   + TotalCustomers );
}
```

Figure 4.7 Java report generator for the single-server queue simulation.


```
public static double exponential(Random rng, double mean) {
 return -mean*Math.log( rng.nextDouble() );
}

public static double SaveNormal;
public static int NumNormals = 0;
public static final double PI = 3.1415927 ;

public static double normal(Random rng, double mean, double sigma) {
        double ReturnNormal;
        // should we generate two normals?
        if(NumNormals == 0 ) {
          double r1 = rng.nextDouble();
          double r2 = rng.nextDouble();
          ReturnNormal = Math.sqrt(-2*Math.log(r1))*Math.cos(2*PI*r2);
          SaveNormal   = Math.sqrt(-2*Math.log(r1))*Math.sin(2*PI*r2);
          NumNormals = 1;
        } else {
          NumNormals = 0;
          ReturnNormal = SaveNormal;
        }
        return ReturnNormal*sigma + mean ;
}
```

Figure 4.8 Random-variate generators for the single-server queue simulation.

The techniques for generating exponentially and normally distributed random variates, discussed in Chapter 8, are based on first generating a $U(0,1)$ random number. For further explanation, the reader is referred to Chapters 7 and 8.

The output from the grocery-checkout-counter simulation is shown in Figure 4.9. It should be emphasized that the output statistics are estimates that contain random error. The values shown are influenced by the particular random numbers that happened to have been used, by the initial conditions at time 0, and by the run length (in this case, 1000 departures). Methods for estimating the standard error of such estimates are discussed in Chapter 11.

In some simulations, it is desired to stop the simulation after a fixed length of time, say TE = 12 hours = 720 minutes. In this case, an additional event type, stop event, is defined and is scheduled to occur by scheduling a stop event as part of simulation initialization. When the stopping event does occur, the cumulative

```
SINGLE SERVER QUEUE SIMULATION - GROCERY STORE CHECKOUT COUNTER
        MEAN INTERARRIVAL TIME                  4.5
        MEAN SERVICE TIME                       3.2
        STANDARD DEVIATION OF SERVICE TIMES     0.6
        NUMBER OF CUSTOMERS SERVED              1000

        SERVER UTILIZATION                      0.671
        MAXIMUM LINE LENGTH                     9.0
        AVERAGE RESPONSE TIME                   6.375 MINUTES
        PROPORTION WHO SPEND FOUR
         MINUTES OR MORE IN SYSTEM              0.604
        SIMULATION RUNLENGTH                    4728.936 MINUTES
        NUMBER OF DEPARTURES                    1000
```

Figure 4.9 Output from the Java single-server queue simulation.

statistics will be updated and the report generator called. The main program and method `Initialization` will require minor changes. Exercise 1 asks the reader to make these changes. Exercise 2 considers balking of customers.

4.5 SIMULATION IN GPSS

GPSS is a highly structured, special-purpose simulation programming language based on the process-interaction approach and oriented toward queueing systems. A block diagram provides a convenient way to describe the system being simulated. There are over 40 standard blocks in GPSS. Entities called transactions may be viewed as flowing through the block diagram. Blocks represent events, delays, and other actions that affect transaction flow. Thus, GPSS can be used to model any situation where transactions (entities, customers, units of traffic) are flowing through a system (e.g., a network of queues, with the queues preceding scarce resources). The block diagram is converted to block statements, control statements are added, and the result is a GPSS model.

The first version of GPSS was released by IBM in 1961. It was the first process-interaction simulation language and became popular; it has been implemented anew and improved by many parties since 1961, with GPSS/H being the most widely used version in use today. Example 4.3 is based on GPSS/H.

GPSS/H is a product of Wolverine Software Corporation, Annandale, VA (Banks, Carson, and Sy, 1995; Henriksen, 1999). It is a flexible, yet powerful tool for simulation. Unlike the original IBM implementation, GPSS/H includes built-in file and screen I/O, use of an arithmetic expression as a block operand, an interactive debugger, faster execution, expanded control statements, ordinary variables and arrays, a floating-point clock, built-in math functions, and built-in random-variate generators.

The animator for GPSS/H is Proof Animation™, another product of Wolverine Software Corporation (Henriksen, 1999). Proof Animation provides a 2-D animation, usually based on a scale drawing. It can run in postprocessed mode (after the simulation has finished running) or concurrently. In postprocessed mode, the animation is driven by two files: the layout file for the static background, and a trace file that contains commands to make objects move and produce other dynamic events. It can work with any simulation package that can write the ASCII trace file. Alternately, it can run concurrently with the simulation by sending the trace file commands as messages, or it can be controlled directly by using its DLL (dynamic link library) version.

Example 4.3: Single-Server Queue Simulation in GPSS/H

Figure 4.10 exhibits the block diagram and Figure 4.11 the GPSS program for the grocery-store checkout-counter model described in Example 4.2. Note that the program (Figure 4.11) is a translation of the block diagram together with additional definition and control statements.

In Figure 4.10, the GENERATE block represents the arrival event, with the interarrival times specified by RVEXPO(1,*&IAT*). RVEXPO stands for "random variable, exponentially distributed," the 1 indicates the random-number stream to use, and *&IAT* indicates that the mean time for the exponential distribution comes from a so-called ampervariable *&IAT*. Ampervariable names begin with the "&" character; Wolverine added ampervariables to GPSS because the original IBM implementation had limited support for ordinary global variables, with no user freedom for naming them. (In the discussion that follows, all nonreserved words are shown in italics.)

The next block is a QUEUE with a queue named *SYSTIME*. It should be noted that the QUEUE block is not needed for queues or waiting lines to form in GPSS. The true purpose of the QUEUE block is to work in conjunction with the DEPART block to collect data on queues or any other subsystem. In Example 4.3,

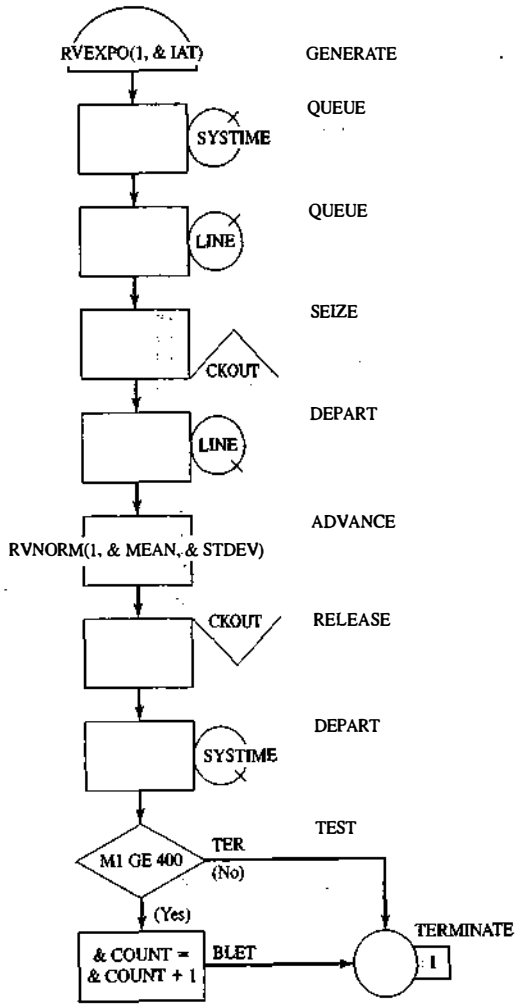


Figure 4.10 GPSS block diagram for the single-server queue simulation.

we want to measure the system response time—that is, the time a transaction spends in the system. Placing a QUEUE block at the point that transactions enter the system and placing the counterpart of the QUEUE block, the DEPART block, at the point that the transactions complete their processing causes the response times to be collected automatically. The purpose of the DEPART block is to signal the end of data collection for an individual transaction. The QUEUE and DEPART block combination is not necessary for queues to be modeled, but rather is used for statistical data collection.

The next QUEUE block (with name *LINE*) begins data collection for the waiting line before the cashier. The customers may or may not have to wait for the cashier. Upon arrival to an idle checkout counter, or after

```
      SIMULATE
*
*     Define Ampervariables
*
      INTEGER     &LIMIT
      REAL        &IAT,&MEAN,&STDEV,&COUNT
      LET         &IAT=4.5
      LET         &MEAN=3.2
      LET         &STDEV=.6
      LET         &LIMIT=1000
*
*     Write Input Data to File
*
      PUTPIC      FILE=OUT,LINES=5,(&IAT,&MEAN,&STDEV,&LIMIT)
      Mean interarrival time                  **.** minutes
      Mean service time                       **.** minutes
      Standard deviation of service time      **.** minutes
      Number of customers to be served        *****
*
*     GPSS/H Block Section
*
      GENERATE    RVEXPO(1,&IAT)  Exponential arrivals
      QUEUE       SYSTIME         Begin response time data collection
      QUEUE       LINE            Customer joins waiting line
      SEIZE       CHECKOUT        Begin checkout at cash register
      DEPART      LINE            Customer starting service leaves queue
      ADVANCE     RVNORM(1,&MEAN,&STDEV)   Customer's service time
      RELEASE     CHECKOUT        Customer leaves checkout area
      DEPART      SYSTIME         End response time data collection
      TEST GE     M1,4,TER        Is response time GE 4 minutes?
      BLET        &COUNT=&COUNT+1 If so, add 1 to counter
TER   TERMINATE   1
*
      START       &LIMIT          Simulate for required number
*
*     Write Customized Output Data to File
*
      PUTPIC      FILE=OUT,LINES=7,(FR(CHECKOUT)/1000,QM(LINE),_
      QT(SYSTIME),&COUNT/N(TER),AC1,N(TER))
      Server utilization                       .***
      Maximum line length                     **
      Average response time                   **.**    minutes
      Proportion who spend four minutes        .***
          or more in the system
      Simulation runlength                    ****.** minutes
      Number of departures                    ****
*
      END
```

Figure 4.11 GPSS/H program for the single-server queue simulation.

advancing to the head of the waiting line, a customer captures the cashier, as represented by the SEIZE block with the resource named *CHECKOUT*. Once the transaction representing a customer captures the cashier represented by the resource CHECKOUT, the data collection for the waiting-line statistics ends, as represented by the DEPART block for the queue named LINE. The transaction's service time at the cashier is

represented by an ADVANCE block. RVNORM indicates "random variable, normally distributed." Again, random-number stream 1 is being used, the mean time for the normal distribution is given by ampervariable *&MEAN*, and its standard deviation is given by ampervariable *&STDEV*. Next, the customer gives up the use of the facility *CHECKOUT* with a RELEASE block. The end of the data collection for response times is indicated by the DEPART block for the queue *SYSTIME*.

Next, there is a TEST block that checks to see whether the time in the system, M1, is greater than or equal to 4 minutes. (Note that M1 is a reserved word in GPSS/H; it automatically tracks transaction total time in system.) In GPSS/H, the maxim is "if true, pass through." Thus, if the customer has been in the system four minutes or longer, the next BLET block (for block LET) adds one to the counter *&COUNT*. If not true, the escape route is to the block labeled *TER*. That label appears before the TERMINATE block whose purpose is the removal of the transaction from the system. The TERMINATE block has a value "1" indicating that one more transaction is added toward the limiting value (or "transactions to go").

The control statements in this example are all of those lines in Figure 4.11 that precede or follow the block section. (There are eleven blocks in the model from the GENERATE block to the TERMINATE block.) The control statements that begin with an "*" are comments, some of which are used for spacing purposes. The control statement SIMULATE tells GPSS/H to conduct a simulation; if it is omitted, GPSS/H compiles the model and checks for errors only. The ampervariables are defined as integer or real by control statements INTEGER and REAL. It seems that the ampervariable *&COUNT* should be defined as an integer; however, it will be divided later by a real value. If it is integer, the result of an integer divided by a real value is truncation, and that is not desired in this case. The four assignment statements (LET) provide data for the simulation. These four values could have been placed directly in the program; however, the preferred practice is to place them in ampervariables at the top of the program so that changes can be made more easily or the model can be modified to read them from a data file.

To ensure that the model data is correct, and for the purpose of managing different scenarios simulated, it is good practice to echo the input data. This is accomplished with a PUTPIC (for "put picture") control statement. The five lines following PUTPIC provide formatting information, with the asterisks being markers (called picture formatting) in which the values of the four ampervariables replace the asterisks when PUTPIC is executed. Thus, "**.**" indicates a value that may have two digits following the decimal point and up to two before it.

The START control statement controls simulation execution. It starts the simulation, sets up a "termination-to-go" counter with initial value its operand (*&LIMIT*), and controls the length of the simulation.

After the simulation completes, a second PUTPIC control statement is used to write the desired output data to the same file *OUT*. The printed statistics are all gathered automatically by GPSS. The first output in the parenthesized list is the server utilization. FR*(CHECKOUT)*/1000 indicates that the fractional utilization of the facility *CHECKOUT* is printed. Because FR*(CHECKOUT)* is in parts per thousand, the denominator is provided to compute fractional utilization. QM*(LINE)* is the maximum value in the queue *LINE* during the simulation. QT*(SYSTIME)* is the average time in the queue *SYSTIME*. *&COUNT*/N*(TER)* is the number of customers who had a response time of four or more minutes divided by the number of customers that went through the block with label *TER*, or N*(TER)*. AC1 is the clock time, whose last value gives the length of the simulation.

The contents of the custom output file *OUT* are shown in Figure 4.12. The standard GPSS/H output file is displayed in Figure 4.13. Although much of the same data shown in the file *OUT* can be found in the standard GPSS/H output, the custom file is more compact and uses the language of the problem rather than GPSS jargon. There are many other reasons that customized output files are useful. For example, if 50 replications of the model are to be made and the lowest, highest, and average value of a response are desired, this can be accomplished by using control statements, with the results in a very compact form, rather than extracting the desired values from 50 standard output files.

```
Mean interarrival time                  4.50 minutes
Mean service time                       3.20 minutes
Standard deviation of service time      0.60 minutes
Number of customers to be served        1000

Server utilization                      0.676
Maximum line length                     7
Average response time                   6.33 minutes
Proportion who spend four minutes       0.646
    or more in the system
Simulation runlength                    4767.27 minutes
Number of departures                    1000
```

Figure 4.12 Customized GPSS/H output report for the single-server queue simulation.

```
RELATIVE CLOCK: 4767.2740 ABSOLUTE CLOCK: 4767.2740

BLOCK CURRENT      TOTAL BLOCK CURRENT     TOTAL
1                   1003 TER               1000
2                   1003
3           3       1003
4                   1000
5                   1000
6                   1000
7                   1000
8                   1000
9                   1000
10                   646

          --AVG-UTIL-DURING--
FACILITY   TOTAL   AVAIL   UNAVL    ENTRIES    AVERAGE CURRENT  PERCENT  SEIZING  PREEMPTING
            TIME    TIME    TIME             TIME/XACT STATUS   AVAIL     XACT       XACT
CHECKOUT   0.676                      1000      3.224   AVAIL

  QUEUE  MAXIMUM   AVERAGE   TOTAL     ZERO   PERCENT   AVERAGE   $AVERAGE  QTABLE   CURRENT
         CONTENTS  CONTENTS ENTRIES  ENTRIES  ZEROS  TIME/UNIT TIME/UNIT NUMBER  CONTENTS
SYSTIME        8     1.331    1003        0             6.325     6.235              3
   LINE        7     0.655    1003      334    33.3     3.111     4.665              3

RANDOM   ANTITHETIC    INITIAL     CURRENT    SAMPLE    CHI-SQUARE
STREAM     VARIATES   POSITION    POSITION     COUNT    UNIFORMITY
     1          OFF     100000      103004      3004        0.83
```

Figure 4.13 Standard GPSS/H output report for the single-server queue simulation.

4.6 SIMULATION IN SSF

The Scalable Simulation Framework (SSF) is an Application Program Interface (API) that describes a set of capabilities for object-oriented, process-view simulation. The API is sparse and was designed to allow implementations to achieve high performance (e.g. on parallel computers). SSF APIs exist for both C++ and in Java,

and implementations exist in both languages. SSF has a wide user base—particularly in network simulation by using the add-on framework SSFNet (`www.ssfnet.org`). Our chapter on network simulation uses SSFNet.

The SSF API defines five base classes. `process` is a class that implements threads of control; the `action` method of a derived class contains the execution body of the thread. The `Entity` class is used to describe simulation objects. It contains state variables, processes, and communication endpoints. The `inChannel` and `outChannel` classes are communication endpoints. The `Event` class defines messages sent between entities. One model entity communicates with another by "writing" an `Event` into an `outChannel`; at some later time, it is available at one or more `inChannels`. A `process` that expects input on an `inChannel` can suspend, waiting for an event on it. These points, and others, will be elaborated upon as we work through an SSF implementation of the single-server queue.

Source code given in Figure 4.14 expresses the logic of arrival generation in SSF for the single-server queue example. The example is built on two SSF processes. One of these generates jobs and adds them to the system; the other services the enqueued jobs. Class `SSQueue` is a class that contains the whole simulation experiment. It uses the auxiliary classes `Random` (for random-number generation) and `Queue` (to implement FIFO queueing of general objects). `SSQueue` defines experimental constants ("public static final" types) and contains SSF communication endpoints `out` and `in`, through which the two processes communicate. `SSQueue` also defines an inner class `arrival`, which stores the identity and arrival time of each job.

Class `Arrivals` is an SSF process. Its constructor stores the identity of the entity that owns it, and creates a random-number generator that is initialized with the seed passed to it. For all but the initial call, method `action` generates and enqueues a new arrival, then blocks (via SSF method `waitFor`) for an inter-arrival time; on the first call, it by-passes the job-generation step and blocks for an initial interarrival time. The call to `waitFor` highlights details needing explanation. An `SSQueue` object calls the `Arrival` constructor and is saved as the "owner." This class contains an auxiliary method `exponential`, which samples an exponential random variable with specified mean by using a specified random-number stream. It also contains methods `d2t` and `t2d` that translate between a discrete "tick"-based integer clock and a double-precision floating-point representation. In the `waitFor` call, we use the same code seen earlier to sample the exponential in double-precision format, but then use `d2t` to convert it into the simulator's integer clock format. The specific conversion factor is listed as a `SSQueue` constant, 10^9 ticks per unit time.

SSF interprocess communication is used sparingly in this example. Because service is nonpreemptive, when a job's service completes, the process providing service can examine the list of waiting customers (in variable `owner.Waiting`) to see whether it needs to give service to another customer. Thus, the only time the server process needs to be told that there is a job waiting is when a job arrives to an empty system. This is reflected in `Arrivals.action` by use of its owner's `out` channel.

A last point of interest is that `Arrivals` is, in SSF terminology, a "simple" process. This means that every statement in `action` that might suspend the process would be the last statement executed under normal execution semantics. The `Arrivals` class tells SSF that it is simple by overriding a default method `isSimple` to return the value true, rather than the default value (false). The key reason for using simple processes is performance—they require that no state be saved, only the condition under which the process ought to be reanimated. And, when it is reanimated, it starts executing at the first line of `action`.

Figure 4.15 illustrates the code for the `Server` process. Like process `Arrival`, its constructor is called by an instance of `SSQueue` and is given the identity of that instance and a random-number seed. Like `Arrival`, it is a simple process. It maintains state variable `in_service` to remember the specifics of a job in service and state variable `service_time` to remember the value of the service time

```
// SSF MODEL OF JOB ARRIVAL PROCESS
class SSQueue extends Entity {

 private static Random rng;
 public static final double MeanServiceTime = 3.2;
 public static final double SIGMA = 0.6;
 public static final double MeanInterarrivalTime = 4.5;
 public static final long ticksPerUnitTime = 1000000000;
 public long generated=0;
 public Queue Waiting;
 outChannel out;
 inChannel in;

 public static long TotalCustomers=0, MaxQueueLength=0, TotalServiceTime=0;
 public static long LongResponse=0, SumResponseTime=0, jobStart;

 class arrival {
  long id, arrival_time;
  public arrival(long num, long a) { id=num; arrival_time = a; }
 }

class Arrivals extends process {
   private Random rng;
   private SSQueue owner;
   public Arrivals (SSQueue _owner, long seed) {
        super(_owner); owner = _owner;
        rng = new Random(seed);
   }
   public boolean isSimple() { return true; }
   public void action() {
      if ( generated++ > 0 ) {
        // put a new Customer on the queue with the present arrival time
        int Size = owner.Waiting.numElements();
        owner.Waiting.enqueue( new arrival(generated, now()));
        if( Size == 0) owner.out.write( new Event() ); // signal start of burst
      }
      waitFor(owner.d2t( owner.exponential(rng, owner.MeanInterarrivalTime)) );
   }
 }
}
```

Figure 4.14 SSF Model of Job-Arrival Process.

sampled for the job in service. When the SSF kernel calls `action`, either a job has completed service, or the `Arrival` process has just signaled `Server` though the `inChannel`. We distinguish the cases by looking at variable `in_service`, which will be nonnull if a job is in service, just now completed. In this case, some statistics are updated. After this task is done, a test is made for customers waiting for service. The first waiting customer is dequeued from the waiting list and is copied into the `in_service` variable; the process then samples a service time and suspends through a `waitFor` call. If no customer was waiting, the process suspends on a `waitOn` statement until an event from the `Arrival` process awakens it.

SSF bridges the gap between models developed in pure Java and models developed in languages specifically designed for simulation. It provides the flexibility offered by a general-programming language, yet has essential support for simulation.


```
// SSF MODEL OF SINGLE SERVER QUEUE : ACCEPTING JOBS
class Server extends process {
 private Random rng;
 private SSQueue owner ;
 private arrival in_service;
 private long service_time;

 public Server(SSQueue _owner, long seed) {
       super(_owner);
       owner = _owner;
       rng = new Random(seed);
 }
 public boolean isSimple() { return true; }

 public void action() {
   //  executes due to being idle and getting a job, or by service time expiration.
   //  if there is a job awaiting service, take it out of the queue
   //  sample a service time, do statistics, and wait for the service epoch

   // if in_service is not null, we entered because of a job completion
   if( in_service != null ) {
       owner.TotalServiceTime += service_time;
       long in_system = (now() -in_service.arrival_time);
       owner.SumResponseTime += in_system;
       if( owner.t2d(in_system) > 4.0 ) owner.LongResponse++;
       in_service = null;
       if( owner.MaxQueueLength < owner.Waiting.numElements() + 1 )
            owner.MaxQueueLength = owner.Waiting.numElements() + 1;
       owner.TotalCustomers++;
   }
   if( owner.Waiting.numElements() > 0 ) {
       in_service = (arrival)owner.Waiting.dequeue();
       service_time = -1;
       while ( service_time < 0.0 )
         service_time = owner.d2t(owner.normal( rng, owner.MeanServiceTime, owner.SIGMA));
                                                                  // model service time
       waitFor( service_time );
   } else {
       waitOn( owner.in ); // we await a wake-up call
   }
 }
}
```

Figure 4.15 SSF Model of Single-Server Queue : Server.

4.7 SIMULATION SOFTWARE

All the simulation packages described in later subsections run on a PC under Microsoft Windows 2000 or XP. Although in terms of specifics the packages all differ, generally they have many things in common.

Common characteristics include a graphical user interface, animation, and automatically collected outputs to measure system performance. In virtually all packages, simulation results may be displayed in tabular or graphical form in standard reports and interactively while running a simulation. Outputs from different scenarios can be compared graphically or in tabular form. Most provide statistical analyses that include confidence intervals for performance measures and comparisons, plus a variety of other analysis methods. Some of the statistical-analysis modules are described in Section 4.8.

All the packages described here take the process-interaction worldview. A few also allow event-scheduling models and mixed discrete–continuous models. For animation, some emphasize scale drawings in 2-D or 3-D; others emphasize iconic-type animations based on schematic drawings or process-flow diagrams. A few offer both scale drawing and schematic-type animations. Almost all offer dynamic business graphing in the form of time lines, bar charts, and pie charts.

In addition to the information contained in this chapter, the websites given below can be investigated:

Arena
`www.arenasimulation.com/`
AutoMod
`www.automod.com`
Delmia/QUEST
`www.delmia.com` and `www.3ds.com`
Extend
`www.imaginethatinc.com/`
Flexsim
`www.flexsim.com/`
Micro Saint
`www.maad.com`
ProModel
`www.promodel.com/`
SIMUL8
`www.simul8.com/`
WITNESS
`www.witness-for-simulation.com/`

4.7.1 Arena

Arena Basic, Standard, and Professional Editions are offered by Systems Modeling Corporation [Bapat and Sturrock, 2003]. Arena can be used for simulating discrete and continuous systems. A recent addition to the Arena family of products is OptQuest for Arena, an optimization software package (discussed in Section 4.8.2.)

The Arena Basic Edition is targeted at modeling business processes and other systems in support of high-level analysis needs. It represents process dynamics in a hierarchical flowchart and stores system information in data spreadsheets. It has built-in activity-based costing and is closely integrated with the flowcharting software Visio.

The Arena Standard Edition is designed for more detailed models of discrete and continuous systems. First released in 1993, Arena employs an object-based design for entirely graphical model development. Simulation models are built from graphical objects called modules to define system logic and such physical components as machines, operators, and clerks. Modules are represented by icons plus associated data entered in a dialog window. These icons are connected to represent entity flow. Modules are organized into collections called templates. The Arena template is the core collection of modules providing general-purpose features for modeling all types of applications. In addition to standard features, such as resources, queues, process logic, and system data, the Arena template includes modules focused on specific aspects of manufacturing and material-handling systems. Arena SE can also be used to model combined discrete/continuous systems, such as pharmaceutical and chemical production, through its built-in continuous-modeling capabilities.

The Arena Professional Edition enhances Arena SE with the capability to craft custom simulation objects that mirror components of the real system, including terminology, process logic, data, performance metrics, and animation. The Arena family also includes products designed specifically to model call centers and high-speed production lines, namely Arena Contact Center and Arena Packaging.

At the heart of Arena is the SIMAN simulation language. For animating simulation models, Arena's core modeling constructs are accompanied by standard graphics for showing queues, resource status, and entity flow. Arena's 2-D animations are created by using Arena's built-in drawing tools and by incorporating clip art, AutoCAD, Visio, and other graphics.

Arena's Input Analyzer automates the process of selecting the proper distribution and its parameters for representing existing data, such as process and interarrival times. The Output Analyzer and Process Analyzer (discussed in Section 4.8.2) automate comparison of different design alternatives.

4.7.2 AutoMod

The AutoMod Product Suite is offered by Brooks Automation [Rohrer, 2003]. It includes the AutoMod simulation package, AutoStat for experimentation and analysis, and AutoView for making AVI movies of the built-in 3-D animation. The main focus of the AutoMod simulation product is manufacturing and material-handling systems. AutoMod's strength is in detailed, large models used for planning, operational decision support, and control-systems testing.

AutoMod has built-in templates for most common material-handling systems, including vehicle systems, conveyors, automated storage and retrieval systems, bridge cranes, power and free conveyors, and kinematics for robotics. With its Tanks and Pipes module, it also supports continuous modeling of fluid and bulk-material flow.

The pathmover vehicle system can be used to model lift trucks, humans walking or pushing carts, automated guided vehicles, trucks, and cars. All the movement templates are based on a 3-D scale drawing (drawn or imported from CAD as 2-D or 3-D). All the components of a template are highly parameterized. For example, the conveyor template contains conveyor sections, stations for load induction or removal, motors, and photo-eyes. Sections are defined by length, width, speed, acceleration, and type (accumulating or nonaccumulating), plus other specialized parameters. Photo-eyes have blocked and cleared timeouts that facilitate modeling of detailed conveyor logic.

In addition to the material-handling templates, AutoMod contains a full simulation programming language. Its 3-D animation can be viewed from any angle or perspective in real time. The user can freely zoom, pan, or rotate the 3-D world.

An AutoMod model consists of one or more systems. A system can be either a process system, in which flow and control logic are defined, or a movement system based on one of the material-handling templates. A model may contain any number of systems, which can be saved and reused as objects in other models. Processes can contain complex logic to control the flow of either manufacturing materials or control messages, to contend for resources, or to wait for user-specified times. Loads can move between processes with or without using movement systems.

In the AutoMod worldview, loads (products, parts, etc.) move from process to process and compete for resources (equipment, operators, vehicles, and queues). The load is the active entity, executing action statements in each process. To move between processes, loads may use a conveyor or vehicle in a movement system.

AutoStat, described in Section 4.8.2, works with AutoMod models to provide a complete environment for the user to define scenarios, conduct experimentation, and perform analyses. It offers optimization based on an evolutionary strategies algorithm.

4.7.3 Extend

The Extend family of products is offered by Imagine That, Inc. [Krahl, 2003]. Extend OR, Industry, and Suite are used for simulating discrete and mixed discrete–continuous systems; Extend CP is for continuous modeling only. Extend combines a block-diagram approach to model-building with a development environment for creating new blocks.

Each Extend block has an icon and encapsulates code, parameters, user interface, animation, and online help. Extend includes a large set of elemental blocks; libraries of blocks for specific application areas, such as manufacturing, business processes, and high-speed processes, are also available. Third-party developers have created Extend libraries for vertical market applications, including supply-chain dynamics, reliability engineering, and pulp and paper processing.

Models are built by placing and connecting blocks and entering the parameters on the block's dialog window. Elemental blocks in Extend include Generator, Queue, Activity, Resource Pool, and Exit. The active entities, called items in Extend, are created at Generator blocks and move from block to block by way of item connectors. Separate value connectors allow the attachment of a calculation to a block parameter or the retrieval of statistical information for reporting purposes. Input parameters can be changed interactively during a model run and can come from external sources. Outputs are displayed dynamically and in graphical and tabular format. The Industry and Suite products also provide an embedded database for centralized information management.

Extend provides iconic process-flow animation of the block diagram. For scaled 2-D animation, Proof Animation [Henriksen, 2002] from Wolverine Software is included in the Suite product. Collections of blocks representing a submodel, such as a subassembly line or functional process, can be grouped into a hierarchical block on the model worksheet; hierarchical blocks can also be stored in a library for reuse. Parameters from the submodel can be grouped and displayed at the level of the hierarchical block for access to model I/O. Extend supports the Microsoft component object model (COM/ActiveX), open database connectivity (ODBC), and Internet data exchange. Activity-based costing, statistical analysis of output data with confidence intervals, and the Evolutionary Optimizer are included.

For creating new blocks, Extend comes with a compiled C-like programming environment. The message-based language includes simulation-specific functions and supports custom interface development. Extend has an open architecture; in most cases, the source code for blocks is available for custom development. The architecture also supports linking to and using code and routines written in external languages.

4.7.4 Flexsim

Flexsim simulation software is developed and owned by Flexsim Software Products, Inc. of Orem, Utah (Nordgren, 2003). Flexsim is a discrete-event, object-oriented simulator developed in C++, using Open GL technology. Animation can be shown in tree view, 2-D, 3-D, and virtual reality. All views can be shown concurrently during the model development of run phase. It integrates Microsoft's Visual C++ IDE and compiler within a graphical 3-D click-and-drag simulation environment.

Flexsim software is used to build models that behave like the actual physical or conceptual systems they represent. A simulation model of any flow system or process can be created in Flexsim by using drag-and-drop model-building objects.

Flexsim is used to improve production efficiencies and reduce operating costs through simulation, experimentation, and optimization of dynamic flow systems. Engineers and managers use Flexsim to evaluate plant capacity, balance packaging and manufacturing lines, manage bottlenecks, solve work-in-process problems, justify capital expenditures, plan equipment maintenance schedules, establish proper inventory levels, improve order-picking systems, and optimize production rates. Flexsim allows end users to introduce and simulate new conditions for the model and to analyze their effects and results in order to find ways to improve the system being studied. By using Flexsim, efficiencies—increased throughput and decreased costs—can be identified, tested, and proven prior to implementing them in the actual system. The results of each simulation can be analyzed graphically through 3-D animation and through statistical reports and graphs, which are all also useful in communicating a model's purpose and results to both technical and nontechnical audiences.

4.7.5 Micro Saint

Micro Saint is offered by Micro Analysis and Design, Inc. [Bloechle and Schunk, 2003]. Micro Saint is a general-purpose, discrete-event, network simulation-software package for building models that simulate real-life processes. With Micro Saint models, users can gain useful information about processes that might be too expensive or time-consuming to test in the real world.

Micro Saint does not use the terminology or graphic representations of a specific industry. A Micro Saint model can be built for any process that can be represented by a flowchart diagram. The terms that are used are defined by the user. In addition, the icons and background for the ActionView animation and the flowcharting symbols are customizable. Micro Saint provides two views of the simulation model. The network diagram view shows the process flowchart in action, and ActionView provides a realistic 2-D picture of the process.

Micro Saint supports the development of models of various complexity to match the user's needs. Simple, functional models can be built by drawing a network diagram and filling in the task-timing information. More complex models can also be built that include dynamically changing variables, probabilistic and tactical branching logic, sorted queues, conditional task execution, animation, optimization, and extensive data collection.

A separate module (called COM Services) is available that enables Micro Saint to exchange data with other software applications and makes it easy to customize the model. In addition, OptQuest optimization is included with Micro Saint and is designed to automatically search for and find optimal or near-optimal solutions to the model.

4.7.6 ProModel

ProModel is offered by PROMODEL Corporation [Harrell, 2003]. It is a simulation and animation tool designed to model manufacturing systems. The company also offers MedModel for healthcare systems and ServiceModel for service systems. ProModel offers 2-D animation with an optional 3-D like perspective view. ProModel's animation is generated automatically as the model is developed.

ProModel has manufacturing-oriented modeling elements and rule-based decision logic. Some systems can be modeled by selecting from ProModel's set of highly parameterized modeling elements. In addition, its simulation programming language provides for modeling special situations not covered by the built-in choices.

The modeling elements in ProModel are parts (entities), locations, resources, path networks, routing and processing logic, and arrivals. Parts arrive and follow the routing and processing logic from location to location. Resources are used to represent people, tools, or vehicles that transport parts between locations, perform an operation on a part at a location, or perform maintenance on a location or other resource that is down. Resources may travel on path networks with given speeds, accelerations, and pickup and setdown travel times. The routing and processing element allows user-defined procedural logic in ProModel's simulation-programming language.

ProModel includes logic for automatically generating cost data associated with a process. Costs can be added for location usage, resources, and entities.

ProModel comes complete with an output viewer, allowing for straightforward data presentation and useful graphics and charts, such as state diagrams.

ProModel's runtime interface allows a user to define multiple scenarios for experimentation. SimRunner (discussed in Section 4.8.2) adds the capability to perform an optimization. It is based on an evolutionary-strategy algorithm, a variant of the genetic algorithm approach. The OptQuest Optimizer (OptQuest for ProModel) is available as an add-on product.

4.7.7 QUEST

QUEST® is offered by Delmia Corp. QUEST (Queuing Event Simulation Tool) is a manufacturing-oriented simulation package. QUEST combines an object-based, true 3-D simulation environment with a graphical user interface and material-flow modules for modeling labor, conveyors, automated guided vehicles, kinematic devices, cranes, fluids, power and free conveyors, and automated storage and retrieval systems. QUEST models incorporate 2-D and 3-D CAD geometry to create a virtual factory environment.

Delmia also offers a number of workcell simulators, including IGRIP® for robotic simulation and programming and ERGO™ for ergonomic analyses. Robots and human-based workcells that are simulated in IGRIP and ERGO can be imported into QUEST models both visually and numerically.

Delmia provides even further integration with QUEST and other manufacturing technologies through PROCESS ENGINEER™, Delmia's process-planning environment. The Manufacturing Hub infrastructure behind this software consists of an object-oriented database for storing Product, Process, and Resource objects that are configuration-managed and effectivity-controlled. A QUEST model is automatically created from the information stored in the database, and the resulting model can be linked to the database for automatic update purposes. QUEST can be used to introduce and update resource-specific information and model output results into the Manufacturing Hub for use in other products.

A QUEST model consists of elements from a number of element classes. Built-in element classes include AGVs and transporters, subresources, buffers, conveyors, power and free systems, labor, machines, parts, container parts, and processes. Each element has associated geometric data and parameters that define its behavior. Parts may have a route and control rules to govern part flow. Commonly needed behavior logic is selected from comprehensive menus, many parameter-driven.

For unique problems, Delmia's QUEST Simulation Control Language (SCL) can be used. This structured simulation-programming language provides distributed processing with access to all system variables. SCL allows expert users to define custom behaviors and to gain control over the simulation.

Delmia QUEST's open architecture allows the advanced user to perform batch simulation runs to automatically collect and tabulate data by using the Batch Control Language (BCL). Replications and parameter optimization are controlled with batch command files or by the OptQuest optimization software, as described in Section 4.8.2.

Output is available both numerically (with the statistical reporting mechanisms) and visually (with a resulting virtual factory-like animation). Statistical output results are available internally through the graphical user interface or externally through HTML and can be customized by using XML or QUEST's own BCL. Digital movies can be created from the animation, or a read-only encrypted version of the model can be authored for viewing and experimentation in QUEST Express™, a "lite" version of QUEST.

4.7.8 SIMUL8

SIMUL8 is provided by SIMUL8 Corporation and was first introduced in 1995. In SIMUL8, models are created by drawing the flow of work with the computer mouse, using a series of icons and arrows to represent the resources and queues in the system. Default values are provided for all properties of the icons, so that the animation can be viewed very early in the modeling process. Drilling down in property boxes opens up progressively more detailed properties. The main focus of SIMUL8 is service industries where people are processing transactions.

Like some other packages, SIMUL8 has the concepts of "Templates" and "Components." Templates, or prebuilt simulations, focus on particular recurring decision types that can be quickly parameterized to fit a specific company issue. Components are user-defined icons that can be reused and shared across a company's simulations. This reduces the time to build simulations, standardizes how some situation are handled across a corporation, and often removes much of the data-collection phase of a simulation study.

SIMUL8 Corporation's approach to business is different from most of the other packages here in that they claim to be aiming to spread simulation very widely across businesses, rather than concentrate it in the hands of dedicated and highly trained simulation professionals. This means they have very different pricing and support policies, but it also means the software has to contain features that watch how the product is being used and provide assistance if some potentially invalid analysis is conducted.

SIMUL8 saves its simulation model and data in XML format so that it will be easy to transfer it to and from other applications. It provides some nonsimulation features that make it possible for the model-builder to create custom user interfaces in spreadsheet, dialog, or wizard form. SIMUL8 has a VBA interface and supports ActiveX/COM so that external applications can build and control SIMUL8 simulations.

The product is available in two levels, Standard and Professional. The two levels provide the same simulation features, but Professional adds 3-D, "Virtual Reality" views of the simulation, and database links to corporate databases and has certain features that are likely to be useful only to full-time simulation modelers. SIMUL8 Professional comes with a license to distribute simulations with a free SIMUL8 Viewer.

4.7.9 WITNESS

WITNESS is offered by the Lanner Group and has separate versions for manufacturing and service industries. It contains many elements for discrete-part manufacturing and also contains elements for continuous processing, such as the flow of fluids through processors, tanks, and pipes.

WITNESS models are based on template elements. These may be customized and combined into module elements and templates for reuse. The standard machine elements can be single, batch, production, assembly, multistation, or multicycle. Other discrete modeling elements include multiple types of conveyor, tracks, vehicles, labor, and carriers. The behavior of each element is described in detailed form in the WITNESS user interface.

The models are displayed in a 2-D layout animation with multiple windows and display layers; there are optional process-flow displays and element-routing overlays. Models can be changed at any point in a model run and saved at any run point for future reload.

Optional WITNESS modules include WITNESS VR, an integrated virtual reality 3-D view of the working model, where there is full mouse control of the camera flight and position. Options exist, too, for post-processed VR with multiscreen projection and various headset technologies. Other WITNESS modules include links to CAD systems, a model documentor, and the WITNESS Optimizer outlined in one section below.

WITNESS has object-model and ActiveX control for simulation embedding and includes direct data links to Microsoft Excel, MINITAB, and any OLEDB database source. XML data format saves offer additional linkage functionality.

4.8 EXPERIMENTATION AND STATISTICAL-ANALYSIS TOOLS

4.8.1 Common Features

Virtually all simulation packages offer various degrees of support for statistical analysis of simulation outputs. In recent years, many packages have added optimization as one of the analysis tools. To support analysis, most packages provide scenario definition, run-management capabilities, and data export to spreadsheets and other external applications.

Optimization is used to find a "near-optimal" solution. The user must define an objective or fitness function, usually a cost or cost-like function that incorporates the trade-off between additional throughput and additional resources. Until recently, the methods available for optimizing a system had difficulty coping with the random and nonlinear nature of most simulation outputs. Advances in the field of metaheuristics have offered new approaches to simulation optimization, ones based on artificial intelligence, neural networks, genetic algorithms, evolutionary strategies, tabu search, and scatter search.

4.8.2 Products

This section briefly discusses Arena's Output and Process Analyzer, AutoStat for AutoMod, OptQuest (which is used in a number of simulation products) and SimRunner for ProModel.

Arena's Output and Process Analyzer

Arena comes with the Output Analyzer and Process Analyzer. In addition, Arena uses OptQuest for optimization.

The Output Analyzer provides confidence intervals, comparison of multiple systems, and warm-up determination to reduce initial condition biases. It creates various plots, charts, and histograms, smoothes responses, and does correlation analysis. To compute accurate confidence intervals, it does internal batching (both within and across replications, with no user intervention) and data truncation to provide stationary, independent, and normally distributed data sets.

The Process Analyzer adds sophisticated scenario-management capabilities to Arena for comprehensive design of experiments. It allows a user to define scenarios, make the desired runs, and analyze the results. It allows an arbitrary number of controls and responses. Responses can be added after runs have been completed. It will rank scenarios by any response and provide summaries and statistical measures of the responses. A user can view 2-D and 3-D charts of response values across either replications or scenarios.

AutoStat

AutoStat is the run manager and statistical-analysis product in the AutoMod product family [Rohrer, 2003]. AutoStat provides a number of analyses, including warm-up determination for steady-state analysis, absolute and comparison confidence intervals, design of experiments, sensitivity analysis, and optimization via an evolutionary strategy. The evolutionary-strategies algorithm used by AutoStat is well suited to finding a near-optimal solution without getting trapped at a local optimum.

With AutoStat, an end user can define any number of scenarios by defining factors and their range of values. Factors include single parameters, such as resource capacity or vehicle speed; single cells in a data file; and complete data files. By allowing a data file to be a factor, a user can experiment with, for example, alternate production schedules, customer orders for different days, different labor schedules, or any other numerical inputs typically specified in a data file. Any standard or custom output can be designated as a response. For each defined response, AutoStat computes descriptive statistics (average, standard deviation, minimum, and maximum) and confidence intervals. New responses can be defined after runs are made, because AutoStat archives and compresses the standard and custom outputs from all runs. Various charts and plots are available to provide graphical comparisons.

AutoStat supports correlated sampling (see Chapter 12) using common random numbers. This sampling technique minimizes variation between paired samples, giving a better indication of the true effects of model changes.

AutoStat is capable of distributing simulation runs across a local area network and pulling back all results to the user's machine. Support for multiple machines and CPU's gives users the ability to make many more runs of the simulation than would otherwise be possible, by using idle machines during off hours. This is especially useful in multifactor analysis and optimization, both of which could require large numbers of runs. AutoStat also has a diagnostics capability that automatically detects "unusual" runs, where the definition of "unusual" is user-definable.

AutoStat also works with two other products from AutoSimulations: the AutoMod Simulator, a spreadsheet-based job-shop simulator; and AutoSched AP, a rule-based simulation package for finite-capacity scheduling in the semiconductor industry.

OptQuest

OptQuest® was developed by Dr. Fred Glover of the University of Colorado, cofounder of OptTek Systems, Inc. [April *et al.*, 2003].

OptQuest is based on a combination of methods: scatter search, tabu search, linear/integer programming, and neural networks. Scatter search is a population-based approach where existing solutions are combined to create new solutions. Tabu search is then superimposed to prohibit the search from reinvestigating previous solutions, and neural networks screen out solutions likely to be poor. The combination of methods allows the search process to escape local optimality in the quest for the best solution.

Some of the differences between OptTek's methods and other methods include

- the ability to avoid being trapped in locally optimal solutions to problems that contain nonlinearities (which commonly are present in real-world problems);
- the ability to handle nonlinear and discontinuous relationships that are not specifiable by the kinds of equations and formulas that are used in standard mathematical programming formulations;
- the ability to solve problems that involve uncertainties, such as those arising from uncertain supplies, demands, prices, costs, flow rates, and queuing rates.

SimRunner

SimRunner was developed by PROMODEL Corporation out of the simulation-optimization research of Royce Bowden, Mississippi State University [Harrell *et al.*, 2003]. It is available for ProModel, MedModel, and ServiceModel.

SimRunner uses genetic algorithms and evolution strategies, which are variants of evolutionary algorithms. Evolutionary algorithms are population-based direct-search techniques. A user first specifies input factors (integer or real-valued decision variables) composed of ProModel macros and then specifies an objective function composed of simulation-output responses. SimRunner manipulates the input factors within boundaries specified by the user seeking to minimize, to maximize, or to achieve a user-specified target value for the objective function. The optimization-output report includes a confidence interval on the mean value of the objective function for each solution evaluated over the course of the optimization and displays 3-D plots of the simulation's output-response surface for the solutions evaluated. In addition to the multivariable optimization module, SimRunner has a utility for helping users estimate the end of the warm-up phase (initialization bias) of a steady-state simulation and the number of replications needed to obtain an estimate of the objective function's mean value to within a specified percentage error and confidence level.

REFERENCES

APRIL, J., F. GLOVER, J. P. KELLY, AND M. LAGUNA [2003], "Practical Introduction to Simulation Optimization," *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., New Orleans, LA, Dec. 7–10, pp. 71–78.

BANKS, J., J. S. CARSON, AND J. N. SY [1995], *Getting Started with GPSS/H*, 2d ed., Wolverine Software Corporation, Annandale, VA.

BANKS, J. [1996], "Interpreting Software Checklists," *OR/MS Today*, June.

BAPAT, V. AND D. STURROCK [2003]. "The Arena Product Family: Enterprise Modeling Solutions," *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., New Orleans, LA. Dec. 7–10, pp. 210–217.

BLOECHLE, W. K., AND D. SCHUNK [2003], "Micro Saint Sharp Simulation Software," *Proceedings of the 200. Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., New Orleans, LA, Dec. 7–10. pp. 182–187.

COWIE, J. [1999], "Scalable Simulation Framework API Reference Manual," www.ssfnet.org/SSFdocs/ssfapiManual.pdf.

CRAIN, R. C., AND J. O. HENRIKSEN [1999], "Simulation Using GPSS/H," *Proceedings of the 1999 Winter Simulation Conference*, P. A. Farrington, H. B. Nembhard, D. T. Sturrock, G. W. Evans, eds., Phoenix, AZ, Dec. 5–8, pp. 182–187.

HARRELL, C. R., B. K. GHOSH, AND R. BOWDEN [2003], *Simulation Using ProModel*, 2d ed., New York: McGraw–Hill.

HARRELL, C. R., AND R. N. PRICE [2003], "Simulation Modeling Using ProModel," *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., New Orleans, LA, Dec. 7–10, pp. 175–181.

HENRIKSEN, J. O. [1999], "General-Purpose Concurrent and Post-Processed Animation with Proof," *Proceedings of the 1999 Winter Simulation Conference*, P. A. Farrington, H. B. Nembhard, D. T. Sturrock, G. W. Evans, eds., Phoenix, AZ, Dec. 5–8, pp. 176–181.

KRAHL, D. [2003], "Extend: An Interactive Simulation Environment," *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., New Orleans, LA, Dec. 7–10, pp. 188–196.

MEHTA, A., AND I. RAWLS [1999], "Business Solutions Using Witness," *Proceedings of the 1999 Winter Simulation Conference*, P. A. Farrington, H. B. Nembhard, D. T. Sturrock, G. W. Evans, eds., Phoenix, AZ, Dec. 5–8, pp. 230–233.

NANCE, R. E. [1995], "Simulation Programming Languages: An Abridged History," Proceedings of the 1995 Winter Simulation Conference, X. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, eds., Arlington, VA, Dec. 13–16, pp. 1307–1313.

NORDGREN, W. B. [2003], "Flexsim Simulation Environment," *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., New Orleans, LA, Dec. 7–10, pp. 197–200.

PRITSKER, A. A. B., AND C. D. PEGDEN [1979], *Introduction to Simulation and SLAM*, John Wiley, New York.

ROHRER, M. W. [2003], "Maximizing Simulation ROI with AutoMod," *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., New Orleans., LA, Dec. 7–10, pp. 201–209.

SWAIN, J. J. [2003], "Simulation Reloaded: Sixth Biennial Survey of Discrete-Event Software Tools," *OR/MS Today*, August, Vol. 30, No. 4, pp. 46–57.

TOCHER, D. D., AND D. G. OWEN [1960], "The Automatic Programming of Simulations," *Proceedings of the Second International Conference on Operational Research*, J. Banbury and J. Maitland, eds., pp. 50–68.

WILSON, J. R., *et al.* [1992], "The Winter Simulation Conference: Perspectives of the Founding Fathers," *Proceedings of the 1992 Winter Simulation Conference*, J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, eds., Arlington, VA, Dec. 13–16, pp.37–62.

EXERCISES

For the exercises below, reader should code the model in a general-purpose language (such as C, C++, or Java), a special-purpose simulation language (such as GPSS/H), or any desired simulation package.

Most problems contain activities that are uniformly distributed over an interval $[a,b]$. Assume that all values between a and b are possible; that is, the activity time is a *continuous* random variable.

The uniform distribution is denoted by $U(a, b)$, where a and b are the endpoints of the interval, or by $m \pm h$, where m is the mean and h is the "spread" of the distribution. These four parameters are related by the equations

$$m = \frac{a+b}{2} \quad h = \frac{b-a}{2}$$
$$a = m-h \quad b = m+h$$

Some of the uniform-random-variate generators available require specification of a and b; others require m and h.

Some problems have activities that are assumed to be normally distributed, as denoted by $N(\mu, \sigma^2)$, where μ is the mean and σ^2 the variance. (Since activity times are nonnegative, the normal distribution is appropriate only if $\mu \geq k\sigma$, where k is at least 4 and preferably 5 or larger. If a negative value is generated, it is discarded.) Other problems use the exponential distribution with some rate λ or mean $1/\lambda$. Chapter 5 reviews these distributions; Chapter 8 covers the generation of random variates having these distributions. All of the languages have a facility to easily generate samples from these distributions. For C, C++, or Java simulations, the student may use the functions given in Section 4.4 for generating samples from the normal and exponential distributions.

1. Make the necessary modifications to the Java model of the checkout counter (Example 4.2) so that the simulation will run for exactly 60 hours.
2. In addition to the changes in Exercise 1, assume that an arriving customer does not join the queue if three or more customers are waiting for service. Make necessary changes to the Java code and run the model.
3. Implement the changes in Exercises 1 and 2 in any of the simulation packages.
4. Ambulances are dispatched at a rate of one every 15 ± 10 minutes in a large metropolitan area. Fifteen percent of the calls are false alarms, which require 12 ± 2 minutes to complete. All other calls can be one of two kinds. The first kind are classified as serious. They constitute 15% of the non-false alarm calls and take 25 ± 5 minutes to complete. The remaining calls take 20 ± 10 minutes to complete. Assume that there are a very large number of available ambulances, and that any number can be on call at any time. Simulate the system until 500 calls are completed.
5. In Exercise 4, estimate the number of ambulances required to provide 100% service.
6. **(a)** In Exercise 4, suppose that there is only one ambulance available. Any calls that arrive while the ambulance is out must wait. Can one ambulance handle the work load?
   (b) Simulate with x ambulances, where $x = 1,2,3$, or 4, and compare the alternatives on the basis of length of time a call must wait, percentage of calls that must wait, and percentage of time the ambulance is out on call.
7. Passengers arrive at the security screening area at Chattahoochee Airport according to a time given by N(20, 3) seconds. At the first point, the boarding pass and ID are checked by one of two people in a time that is distributed N(12, 1) seconds. (Passengers always pick the shortest line when there is an option.) The next step is the X-ray area which takes a time that is N(15, 2) seconds; there are two lanes open at all times. Some 15% of the people have to be rechecked for a time that N(100, 10) seconds. The number of recheckers needed is to be determined. Simulate this system for eight hours with one and two recheckers.
8. A superhighway connects one large metropolitan area to another. A vehicle leaves the first city every 20 ± 15 seconds. Twenty percent of the vehicles have 1 passenger, 30% of the vehicles have 2 passengers, 10% have 3 passengers, and 10% have 4 passengers. The remaining 30% of the vehicles are buses, which carry 40 people. It takes 60 ± 10 minutes for a vehicle to travel between the two metropolitan areas. How long does it take for 5000 people to arrive in the second city?
9. A restaurant has two sections, that is, meals section and tiffin section. Customers arrive at the restaurant at the rate of one every 60 ± 30 seconds. Of the arriving customers, 50% take only tiffin and 50% take only meals. Immaterial of the type of the customer, it takes 75 ± 40 seconds to provide service. Assuming that there are sufficient number of servers available, determine the time taken to serve 100 customers.
10. Re-do Exercise 9, assuming that of the arriving customers, 50% take only tiffin, 30% take only meals, and the remaining 20% take a combination of meals and tiffin.
11. For Exercise 10, what is the maximum number of servers needed during the course of simulation? Reduce the number of servers one by one and determine the total time to complete 100 services.

12. Customers arrive at an Internet center at the rate of one every 15 ± 5 minutes. 80% of the customers check simply their email inbox, while the remaining 20% download and upload files. An email customer spends 5 ± 2 minutes in the center and the download customer spends 15 ± 5 minutes. Simulate the service completion of 500 customers. Of these 500 customers, determine the number of email and download customers and compare with the input percentage.

13. An airport has two concourses. Concourse 1 passengers arrive at a rate of one every 15 ± 2 seconds. Concourse 2 passengers arrive at a rate of one every 10 ± 5 seconds. It takes 30 ± 5 seconds to walk down concourse 1 and 35 ± 10 seconds to walk down concourse 2. Both concourses empty into the main lobby, adjacent to the baggage claim. It takes 10 ± 3 seconds to reach the baggage claim area from the main lobby. Only 60% of the passengers go to the baggage claim area. Simulate the passage of 500 passengers through the airport system. How many of these passengers went through the baggage claim area? In this problem, the expected number through the baggage claim area can be computed by 0.60(500)=300. How close is the simulation estimate to the expected number? Why the difference?

14. In a multiphasic screening clinic, patients arrive at a rate of one every 5 ± 2 minutes to enter the audiology section. The examination takes 3 ± 1 minutes. Eighty percent of the patients were passed on to the next test with no problems. Of the remaining 20%, one-half require simple procedures that take 2 ± 1 minutes and are then sent for reexamination with the same probability of failure. The other half are sent home with medication. Simulate the system to estimate how long it takes to screen and pass 200 patients. (*Note:* Persons sent home with medication are not considered "passed.")

15. Consider a bank with four tellers. Tellers 3 and 4 deal only with business accounts; Tellers 1 and 2 deal only with general accounts. Clients arrive at the bank at a rate of one every 3 ± 1 minutes. Of the clients, 33% are business accounts. Clients randomly choose between the two tellers available for each type of account. (Assume that a customer chooses a line without regard to its length and does not change lines.) Business accounts take 15 ± 10 minutes to complete, and general accounts take 6 ± 5 minutes to complete. Simulate the system for 500 transactions to be completed. What percentage of time is each type of teller busy? What is the average time that each type of customer spends in the bank?

16. Repeat Exercise 15, but assuming that customers join the shortest line for the teller handling their type of account.

17. In Exercises 15 and 16, estimate the mean delay of business customers and of general customers. (Delay is time spent in the waiting line, and is exclusive of service time.) Also estimate the mean length of the waiting line, and the mean proportion of customers who are delayed longer than 1 minute.

18. Three different machines are available for machining a special type of part for 1 hour of each day. The processing-time data is as follows:

Machine	*Time to Machine One Part*
1	20 ± 4 seconds
2	10 ± 3 seconds
3	15 ± 5 seconds

Assume that parts arrive by conveyor at a rate of one every 15 ± 5 seconds for the first 3 hours of the day. Machine 1 is available for the first hour, machine 2 for the second hour, and machine 3 for the third hour of each day. How many parts are produced in a day? How large a storage area is needed for parts waiting for a machine? Do parts "pile up" at any particular time? Why?

19. People arrive at a self-service cafeteria at the rate of one every 30 ± 20 seconds. Forty percent go to the sandwich counter, where one worker makes a sandwich in 60 ± 30 seconds. The rest go to the main counter, where one server spoons the prepared meal onto a plate in 45 ± 30 seconds. All customers must pay a single cashier, which takes 25 ± 10 seconds. For all customers, eating takes 20 ± 10 minutes. After eating, 10% of the people go back for dessert, spending an additional 10 ± 2 minutes altogether in the cafeteria. Simulate until 100 people have left the cafeteria. How many people are left in the cafeteria, and what are they doing, at the time the simulation stops?

20. Customers arrive at a nationalized bank at the rate of one every 60 ± 40 seconds. 60% of the customers perform money transactions and the remaining 40% do other things such as getting the draft, updating passbooks, etc., which require 3 ± 1 and 4 ± 1 minutes, respectively. Currently, there are separate counters for both the activities. Customers feel that if single window concept is introduced, average waiting time could be reduced. Justify by simulating 200 arrivals.

21. In Exercise 20, in single window system, if an arriving customer balks if three or more customers are in the queue, determine the number of customers balked in each category.

22. Loana Tool Company rents chain saws. Customers arrive to rent chain saws at the rate of one every 30 ± 30 minutes. Dave and Betty handle these customers. Dave can rent a chain saw in 14 ± 4 minutes. Betty takes 10 ± 5 minutes. Customers returning chain saws arrive at the same rate as those renting chain saws. Dave and Betty spend 2 minutes with a customer to check in the returned chain saw. Service is first-come-first-served. When no customers are present, or Betty alone is busy, Dave gets these returned saws ready for rerenting. For each saw, this maintenance and cleanup takes him 6 ± 4 minutes and 10 ± 6 minutes, respectively. Whenever Dave is idle, he begins the next maintenance or cleanup. Upon finishing a maintenance or cleanup, Dave begins serving customers if one or more is waiting. Betty is always available for serving customers. Simulate the operation of the system starting with an empty shop at 8:00 A.M., closing the doors at 6:00 P.M., and getting chain saws ready for re-renting until 7:00 P.M. From 6:00 until 7:00 P.M., both Dave and Betty do maintenance and cleanup. Estimate the mean delay of customers who are renting chain saws.

23. The Department of Industrial Engineering of a university has one Xerox machine. Users of this machine arrive at the rate of one every 20 ± 2 minutes and use it for 15 ± 10 minutes. If the machine is busy, 90% of the users wait and finish the job, while the 10% of the users come back after 10 minutes. Assume that they do not balk again. Simulate for 500 customers and find out the probability that a balking customer need not wait during the second attempt.

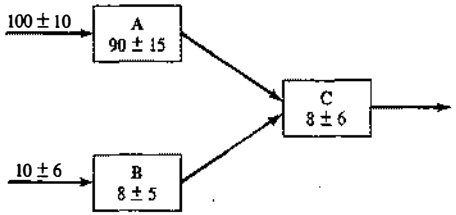
24. Go Ape! buys a Banana II computer to handle all of its web-browsing needs. Web-browsing employees arrive every 10 ± 10 minutes to use the computer. Web-browsing takes 7 ± 7 minutes. The monkeys that run the computer cause a system failure every 60 ± 60 minutes. The failure lasts for 8 ± 4 minutes. When a failure occurs, the web-browsing that was being done resumes processing from where it was left off. Simulate the operation of this system for 24 hours. Estimate the mean system response time. (A system response time is the length of time from arrival until web-browsing is completed.) Also estimate the mean delay for those web-browsing employees that are in service when a computer system failure occurs.

25. Able, Baker, and Charlie are three carhops at the Sonic Drive-In (service at the speed of sound!). Cars arrive every 5 ± 5 minutes. The carhops service customers at the rate of one every 10 ± 6 minutes. However, the customers prefer Able over Baker, and Baker over Charlie. If the carhop of choice is busy, the customers choose the first available carhop. Simulate the system for 1000 service completions. Estimate Able's, Baker's, and Charlie's utilization (percentage of time busy).

26. Jiffy Car Wash is a five-stage operation that takes 2 ± 1 minutes for each stage. There is room for 6 cars to wait to begin the car wash. The car wash facility holds 5 cars, which move through the system in

order, one car not being able to move until the car ahead of it moves. Cars arrive every 2.5 ± 2 minutes for a wash. If the car cannot get into the system, it drives across the street to Speedy Car Wash. Estimate the balking rate per hour. That is, how many cars drive off per hour? Simulate for one 12-hour day.

27. Consider the three machines *A*, *B*, and *C* pictured below. Arrivals of parts and processing times are as indicated (times in ininutes).



Machine *A* processes type *I* parts, machine *B* processes type *II* parts, and machine *C* processes both types of parts. All machines are subject to random breakdown: machine *A* every 400 ± 350 minutes with a down time of 15 ± 14 minutes, machine *B* every 200 ± 150 minutes with a downtime of 10 ± 8 minutes, and machine *C* almost never, so its downtime is ignored. Parts from machine *A* are processed at machine *C* as soon as possible, ahead of any type *II* parts from machine *B*. When machine *A* breaks down, any part in it is sent to machine *B* and processed as soon as *B* becomes free, but processing begins over again, taking 100 ± 20 minutes. Again, type *I* parts from machine *A* are processed ahead of any parts waiting at *B*, but after any part currently being processed. When machine *B* breaks down, any part being processed resumes processing as soon as *B* becomes available. All machines handle one part at a time. Make two independent replications of the simulation. Each replication will consist of an 8-hour initialization phase to load the system with parts, followed by a 40-hour steady-state run. (Independent replications means that each run uses a different stream of random numbers.) Management is interested in the long-run throughput [i.e., the number of parts of each type (*I* and *II*) produced per 8-hour day], long-run utilization of each machine, and the existence of bottlenecks (long "lines" of waiting parts, as measured by the queue length at each machine). Report the output data in a table similar to the following:

	Run 1	*Run 2*	*Average of 2 Runs*
Utilization A			
Utilization B			
Etc.			

Include a brief statement summarizing the important results.

28. Students are arriving at the college office at the rate of one every 6 ± 2 minutes to pay the fees. They hand over the forms to one of the two clerks available and it takes 10 ± 2 minutes for the clerk to verify each form. Then the forms are sent to a single cashier who takes 6 ± 1 minute per form. Simulate the system for 100 hours and determine the

(a) utilization of each clerk
(b) utilization of the cashier
(c) average time required to process a form (clerk + cashier)

29. People arrive at a visa office at the rate of one every 15 ±10 minutes. There are three officers (A, B, and C) who scrutinize the applications for a duration of 30 ± 10 minutes. From the past records, it is found

that on an average, 25% of the applications are rejected. Visa applicants form a single line and go to the officer whoever becomes free. If all the three are free, customers always select officer B who is believed to be considerate. Simulate for 500 visa applicants and determine

a) How many of them selected officer B?
b) How many visa applications are rejected?

30. People arrive at a microscope exhibit at a rate of one every 8 ± 2 minutes. Only one person can see the exhibit at a time. It takes 5 ± 2 minutes to see the exhibit. A person can buy a "privilege" ticket for $1 which gives him or her priority in line over those who are too cheap to spend the buck. Some 50% of the viewers are willing to do this, but they make their decision to do so only if one or more people are in line when they arrive. The exhibit is open continuously from 10:00 A.M. to 4:00 P.M. Simulate the operation of the system for one complete day. How much money is generated from the sale of privilege tickets?

31. Two machines are available for drilling parts (A-type and B-type). A-type parts arrive at a rate of one every 10 ± 3 minutes, B-type parts at a rate of one every 3 ± 2 minutes. For B-type parts, workers choose an idle machine, or if both drills, the Dewey and the Truman, are busy, they choose a machine at random and stay with their choice. A-type parts must be drilled as soon as possible; therefore, if a machine is available, preferably the Dewey, it is used; otherwise the part goes to the head of the line for the Dewey drill. All jobs take 4 ± 3 minutes to complete. Simulate the completion of 100 A-type parts. Estimate the mean number of A-type parts waiting to be drilled.

32. A computer center has two color printers. Students arrive at a rate of one every 8 ± 2 minutes to use the color printer. They can be interrupted by professors, who arrive at a rate of one every 12 ± 2 minutes. There is one systems analyst who can interrupt anyone, but students are interrupted before professors. The systems analyst spends 6 ± 4 minutes on the color printer and then returns in 20 ± 5 minutes. Professors and students spend 4 ± 2 minutes on the color printer. If a person is interrupted, that person joins the head of the queue and resumes service as soon as possible. Simnlate for 50 professor-or-analyst jobs. Estimate the interruption rate per hour, and the mean length of the waiting line of students.

33. Parts are machined on a drill press. They arrive at a rate of one every 5 ± 3 minutes, and it takes 3 ± 2 minutes to machine them. Every 60 ± 60 minutes, a rush job arrives, which takes 12 ± 3 minutes to complete. The rush job interrupts any nonrush job. When the regular job returns to the machine, it stays only for its remaining process time. Simulate the machining of 10 rush jobs. Estimate the mean system response time for each type of part. (A response time is the total time that a part spends in the system.)

34. Pull system is used to assemble items in an assembly line. There are two stations. Station I receives items at the rate of one every 12 ± 3 minutes. The operator in station I takes 14 ± 4 minutes, while the station II operator takes 15 ± 2 minutes. The space between the two stations can accommodate only three parts. Hence, if the space is full, the station I operator has to wait till the station II operator removes one part. Simulate the system for 8 hours of operation.

35. For Exercise 34, comment on the output of the model as to whether it will give the true utilization of the station I server.

36. A patient arrives at the Emergency Room at Hello-Hospital about every 40 ± 19 minutes. Each patient will be treated by either Doctor Slipup or Doctor Gutcut. Twenty percent of the patients are classified as NIA (need immediate attention) and the rest as CW (can wait). NIA patients are given the highest priority (3), see a doctor as soon as possible for 40 ± 37 minutes, but then their priority is reduced to 2 and they wait until a doctor is free again, when they receive further treatment for 30 ± 25 minutes and are then discharged. CW patients initially receive the priority 1 and are treated (when their turn comes) for 15 ± 14 minutes; their priority is then increased to 2, they wait again until a doctor is free and receive

10 ± 8 minutes of final treatment, and are then discharged. Simulate for 20 days of continuous operation, 24 hours per day. Precede this by a 2-day initialization period to load the system with patients. Report conditions at times 0 days, 2 days, and 22 days. Does a 2-day initialization appear long enough to load the system to a level reasonably close to steady-state conditions? (a) Measure the average and maximum queue length of NIA patients from arrival to first seeing a doctor. What percent do not have to wait at all? Also tabulate and plot the distribution of this initial waiting time for NIA patients. What percent wait less than 5 minutes before seeing a doctor? (b) Tabulate and plot the distribution of total time in system for all patients. Estimate the 90% quantile—that is, 90% of the patients spend less than x amount of time in the system. Estimate x. (c) Tabulate and plot the distribution of remaining time in system from after the first treatment to discharge, for all patients. Estimate the 90% quantile. (*Note:* Most simulation packages provide the facility to automatically tabulate the distribution of any specified variable.)

37. People arrive at a newspaper stand with an interarrival time that is exponentially distributed with a mean of 0.5 minute. Fifty-five percent of the people buy just the morning paper, 25% buy the morning paper and a *Wall Street Journal*. The remainder buy only the *Wall Street Journal*. One clerk handles the *Wall Street Journal* sales, another clerk morning-paper sales. A person buying both goes to the *Wall Street Journal* clerk. The time it takes to serve a customer is normally distributed with a mean of 40 seconds and a standard deviation of 4 seconds for all transactions. Collect statistics on queues for each type of transaction. Suggest ways for making the system more efficient. Simulate for 4 hours.

38. Bernie remodels houses and makes room additions. The time it takes to finish a job is normally distributed with a mean of 17 elapsed days and a standard deviation of 3 days. Homeowners sign contracts for jobs at exponentially distributed intervals having a mean of 20 days. Bernie has only one crew. Estimate the mean waiting time (from signing the contract until work begins) for those jobs where a wait occurs. Also estimate the percentage of time the crew is idle. Simulate until 100 jobs have been completed.

39. In a certain factory, the tool crib is manned by a single clerk. There are two types of tool request and the time to process a tool request depends on the type of tool request as

Type of Request	*Interarrival Time (Second)*	*Service Time (Second)*
1	Exponential with mean 420	Normal (300,75)
2	Exponential with mean 300	Normal (100,40)

The clerk has been serving the mechanics on FCFS basis. Simulate the system for one day operation (8 hours).

40. In Exercise 39, the management feels that the average number of waiting mechanics can be reduced if Type 2 requests are served ahead of Type 1. Justify.

41. The interarrival time for parts needing processing is given as follows:

Interarrival Time (Seconds)	*Proportion*
10–20	0.20
20–30	0.30
30–40	0.50

There are three types of parts: A, B, and C. The proportion of each part, and the mean and standard deviation of the normally distributed processing times are as follows:

Part Type	*Proportion*	*Mean*	*Standard Deviation*
A	0.5	30 seconds	3 seconds
B	0.3	40 seconds	4 seconds
C	0.2	50 seconds	7 seconds

Each machine processes any type of part, one part at a time. Use simulation to compare one with two with three machines working in parallel. What criteria would be appropriate for such a comparison?

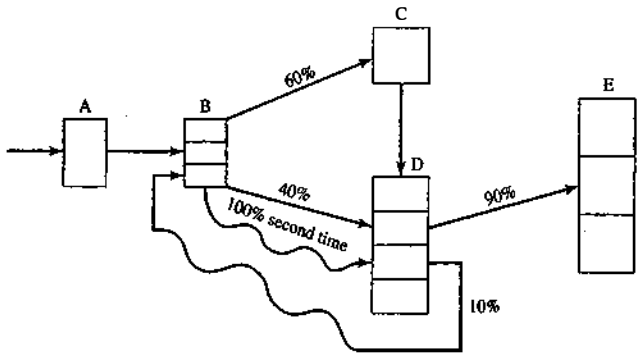
42. Orders are received for one of four types of parts. The interarrival time between orders is exponentially distributed with a mean of 10 minutes. The table that follows shows the proportion of the parts by type and the time to fill each type of order by the single clerk.

Part Type	*Percentage*	*Service Time (Minutes)*
A	40	N(6.1, 1.3)
B	30	N(9.1, 2.9)
C	20	N(11.8, 4.1)
D	10	N(15.1, 4.5)

Orders of types A and B are picked up immediately after they are filled, but orders of types C and D must wait 10 ± 5 minutes to be picked up. Tabulate the distribution of time to complete delivery for all orders combined. What proportion take less than 15 minutes? What proportion take less than 25 minutes? Simulate for an 8-hour initialization period, followed by a 40-hour run. Do not use any data collected in the 8-hour initialization period.

43. Three independent widget-producing machines all require the same type of vital part, which needs frequent maintenance. To increase production it is decided to keep two spare parts on hand (for a total of $2 + 3 = 5$ parts). After 2 hours of use, the part is removed from the machine and taken to a single technician, who can do the required maintenance in 30 ± 20 minutes. After maintenance, the part is placed in the pool of spare parts, to be put into the first machine that requires it. The technician has other duties, namely, repairing other items which have a higher priority and which arrive every 60 ± 20 minutes requiring 15 ± 15 minutes to repair. Also, the technician takes a 15-minute break in each 2-hour time period. That is, the technician works 1 hour 45 minutes, takes off 15 minutes, works 1 hour 45 minutes, takes off 15 minutes, and so on. (a) What are the model's initial conditions—that is, where are the parts at time 0 and what is their condition? Are these conditions typical of "steady state"? (b) Make each replication of this experiment consist of an 8-hour initialization phase followed by a 40-hour data-collection phase. Make four statistically independent replications of the experiment all in one computer run (i.e., make four runs with each using a different set of random numbers). (c) Estimate the mean number of busy machines and the proportion of time the technician is busy. (d) Parts are estimated to cost the company $50 per part per 8-hour day (regardless of how much they are in use). The cost of the technician is $20 per hour. A working machine produces widgets worth $100 for each hour of production. Develop an expression to represent total cost per hour which can be attributed to widget production (i.e., not all of the technician's time is due to widget production). Evaluate this expression, given the results of the simulation.

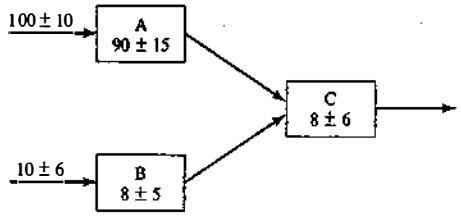
44. The Wee Willy Widget Shop overhauls and repairs all types of widgets. The shop consists of five work stations, and the flow of jobs through the shop is as depicted here:



Regular jobs arrive at station A at the rate of one every 15 ± 13 minutes. Rush jobs arrive every 4 ± 3 hours and are given a higher priority except at station C, where they are put on a conveyor and sent through a cleaning and degreasing operation along with all other jobs. For jobs the first time through a station, processing and repair times are as follows:

Station	*Number Machines or Workers*	*Processing and/or Repair Times (Minutes)*	*Description*
A	1	12 ± 21	Receiving clerk
B	3	40 ± 20	Disassembly and parts replacement
C	1	20	Degreaser
D	4	50 ± 40	Reassembly and adjustments
E	3	40 ± 5	Packing and shipping

The times listed above hold for all jobs that follow one of the two sequences $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$ or $A \rightarrow B \rightarrow D \rightarrow E$. However, about 10% of the jobs coming out of station D are sent back to B for further work (which takes 30 ± 10 minutes) and then are sent to D and finally to E. The path of these jobs is as follows:



Every 2 hours, beginning 1 hour after opening, the degreasing station C shuts down for routine maintenance, which takes 10 ± 1 minute. However, this routine maintenance does not begin until the current widget, if any, has completed its processing.

(a) Make three independent replications of the simulation model, where one replication equals an 8-hour simulation run, preceded by a 2-hour initialization run. The three sets of output represent three typical days. The main performance measure of interest is mean response time per job, where a response time is the total time a job spends in the shop. The shop is never empty in the morning, but the model will be empty without the initialization phase. So run the model for a 2-hour initialization period and collect statistics from time 2 hours to time 10 hours. This "warm-up" period will reduce the downward bias in the estimate of mean response time. Note that the 2-hour warm-up is a device to load a simulation model to some more realistic level than empty. From each of the three independent replications, obtain an estimate of mean response time. Also obtain an overall estimate, the sample average of the three estimates.

(b) Management is considering putting one additional worker at the busiest station (A, B, D, or E). Would this significantly improve mean response time?

(c) As an alternative to part (b), management is considering replacing machine C with a faster one that processes a widget in only 14 minutes. Would this significantly improve mean response time?

45. A building-materials firm loads trucks with two payloader tractors. The distribution of truck-loading times has been found to be exponential with a mean loading time of 6 minutes. The truck interarrival time is exponentially distributed with an arrival rate of 16 per hour. The waiting time of a truck and driver is estimated to cost \$50 per hour. How much (if any) could the firm save (per 10 hour day) if an overhead hopper system that would fill any truck in a constant time of 2 minutes is installed? (Assume that the present tractors could and would adequately service the conveyors loading the hoppers.)

46. A milling-machine department has 10 machines. The runtime until failure occurs on a machine is exponentially distributed with a mean of 20 hours. Repair times are uniformly distributed between 3 and 7 hours. Select an appropriate run length and appropriate initial conditions.

(a) How many repair persons are needed to ensure that the mean number of machines running is greater than eight?

(b) If there are two repair persons, estimate the number of machines that are either running or being served.

47. Jobs arrive every 300 ± 30 seconds to be processed through a process that consists of four operations: OP10 requires 50 ± 20 seconds, OP20 requires 70 ± 25 seconds, OP30 requires 60 ± 15 seconds, OP40 requires 90 ± 30 seconds. Simulate this process until 250 jobs are completed; then combine the four operations of the job into one with the distribution 240 ± 100 seconds and simulate the process with this distribution. Does the average time in the system change for the two alternatives?

48. Ships arrive at a harbor at the rate of one every 60 ± 30 minutes. There are six berths to accommodate them. They also need the service of a crane for unloading and only one crane is available. After unloading, 10% of the ships stay for refuel before leaving, while the others leave immediately. Ships do not require the use of crane for refueling. It takes 7 ± 3 hours for unloading and 60 ± 20 minutes for refueling. Assume that the crane is subjected to routine maintenance once in every 100 hours, and it takes 5 ± 2 hours to complete the maintenance. The crane's unloading operation is not interrupted for maintenance. The crane is taken for maintenance as early as possible after completing the current unloading activity. Simulate the system for unloading 500 ships that require refueling.

49. Two types of jobs arrive to be processed on the same machine. Type 1 jobs arrive every 80 ± 30 seconds and require 35 ± 20 seconds for processing. Type 2 jobs arrive every 100 ± 40 seconds and require

20 ± 15 seconds for processing. Engineering has judged that there is excess capacity on the machine. For a simulation of 8 hours of operation of the system, find X for Type 3 jobs that arrive every $X \pm 0.4X$ seconds and require a time of 30 seconds on the machine so that the average number of jobs waiting to be processed is two or less.

50. Using spreadsheet software, generate 1000 uniformly distributed random values with mean 10 and spread 2. Plot these values with intervals of width 0.5 between 8 and 12. How close did the simulated set of values come to the expected number in each interval?

51. Using a spreadsheet, generate 1000 exponentially distributed random values with a mean of 10. What is the maximum of the simulated values? What fraction of the generated values is less than the mean of 10? Plot a histogram of the generated values. (Hint: If you cannot find an exponential generator in the spreadsheet you use, use the formula −10*LOG(1−R), where R is a uniformly distributed random number from 0 to 1 and LOG is the natural logarithm. The rationale for this formula is explained in Chapter 8 on random-variate generators.)

Part II

Mathematical and Statistical Models

5

Statistical Models in Simulation

In modeling real-world phenomena, there are few situations where the actions of the entities within the system under study can be predicted completely. The world the model-builder sees is probabilistic rather than deterministic. There are many causes of variation. The time it takes a repairperson to fix a broken machine is a function of the complexity of the breakdown, whether the repairperson brought the proper replacement parts and tools to the site, whether another repairperson asks for assistance during the course of the repair, whether the machine operator receives a lesson in preventive maintenance, and so on. To the model-builder, these variations appear to occur by chance and cannot be predicted. However, some statistical model might well describe the time to make a repair.

An appropriate model can be developed by sampling the phenomenon of interest. Then, through educated guesses (or using software for the purpose), the model-builder would select a known distribution form, make an estimate of the parameter(s) of this distribution, and then test to see how good a fit has been obtained. Through continued efforts in the selection of an appropriate distribution form, a postulated model could be accepted. This multistep process is described in Chapter 9.

Section 5.1 contains a review of probability terminology and concepts. Some typical applications of statistical models, or distribution forms, are given in Section 5.2. Then, a number of selected discrete and continuous distributions are discussed in Sections 5.3 and 5.4. The selected distributions are those that describe a wide variety of probabilistic events and, further, appear in different contexts in other chapters of this text. Additional discussion about the distribution forms appearing in this chapter, and about distribution forms mentioned but not described, is available from a number of sources [Hines and Montgomery, 1990; Ross, 2002; Papoulis, 1990; Devore, 1999; Walpole and Myers, 2002; Law and Kelton, 2000]. Section 5.5 describes the Poisson process and its relationship to the exponential distribution. Section 5.6 discusses empirical distributions.

5.1 REVIEW OF TERMINOLOGY AND CONCEPTS

1. *Discrete random variables.* Let X be a random variable. If the number of possible values of X is finite, or countably infinite, X is called a discrete random variable. The possible values of X may be listed as $x_1, x_2, \ldots$. In the finite case, the list terminates; in the countably infinite case, the list continues indefinitely.

Example 5.1

The number of jobs arriving each week at a job shop is observed. The random variable of interest is X, where

$$X = \text{number of jobs arriving each week}$$

The possible values of X are given by the range space of X, which is denoted by R_X. Here $R_X = \{0, 1, 2, \ldots\}$.

Let X be a discrete random variable. With each possible outcome x_i in R_X, a number $p(x_i) = P(X = x_i)$ gives the probability that the random variable equals the value of x_i. The numbers $p(x_i)$, $i = 1, 2, \ldots$, must satisfy the following two conditions:

1. $p(x_i) \geq 0$, for all i
2. $\sum_{i=1}^{\infty} p(x_i) = 1$

The collection of pairs $(x_i, p(x_i))$, $i = 1, 2, \ldots$ is called the probability distribution of X, and $p(x_i)$ is called the probability mass function (pmf) of X.

Example 5.2

Consider the experiment of tossing a single die. Define X as the number of spots on the up face of the die after a toss. Then $R_X = \{1, 2, 3, 4, 5, 6\}$. Assume the die is loaded so that the probability that a given face lands up is proportional to the number of spots showing. The discrete probability distribution for this random experiment is given by

x_i	1	2	3	4	5	6
$p(x_i)$	1/21	2/21	3/21	4/21	5/21	6/21

The conditions stated earlier are satisfied—that is, $p(x_i) \geq 0$ for $i = 1, 2, \ldots, 6$ and $\sum_{i=1}^{\infty} p(x_i) = 1/21 + \cdots + 6/21 = 1$. The distribution is shown graphically in Figure 5.1.

2. *Continuous random variables.* If the range space R_X of the random variable X is an interval or a collection of intervals, X is called a continuous random variable. For a continuous random variable X, the probability that X lies in the interval $[a, b]$ is given by

$$P(a \leq X \leq b) = \int_a^b f(x)\,dx \tag{5.1}$$

The function $f(x)$ is called the probability density function (pdf) of the random variable X. The pdf satisfies the following conditions:

a. $f(x) \geq 0$ for all x in R_X

b. $\int_{R_X} f(x)dx = 1$

c. $f(x) = 0$ if x is not in R_X

As a result of Equation (5.1), for any specified value x_0, $P(X = x_0) = 0$, because

$$\int_{x_0}^{x_0} f(x)\,dx = 0$$

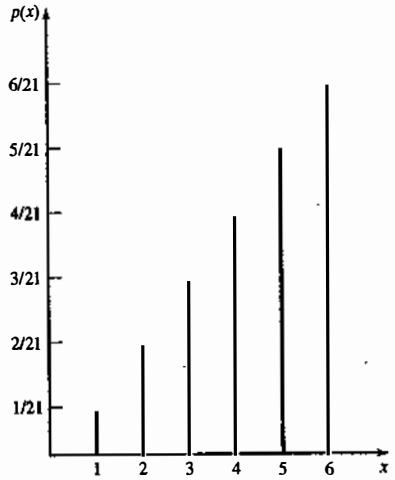


Figure 5.1 Probability mass function for loaded-die example.

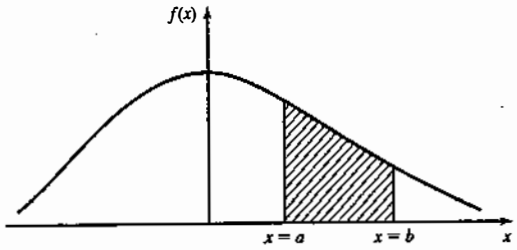


Figure 5.2 Graphical interpretation of $P(a < X < b)$.

$P(X = x_0) = 0$ also means that the following equations hold:

$$P(a \leq X \leq b) = P(a < X \leq b) = P(a \leq X < b) = P(a < X < b) \tag{5.2}$$

The graphical interpretation of Equation (5.1) is shown in Figure 5.2. The shaded area represents the probability that X lies in the interval $[a, b]$.

Example 5.3

The life of a device used to inspect cracks in aircraft wings is given by X, a continuous random variable assuming all values in the range $x \geq 0$. The pdf of the lifetime, in years, is as follows:

$$f(x) = \begin{cases} \frac{1}{2}e^{-x/2}, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

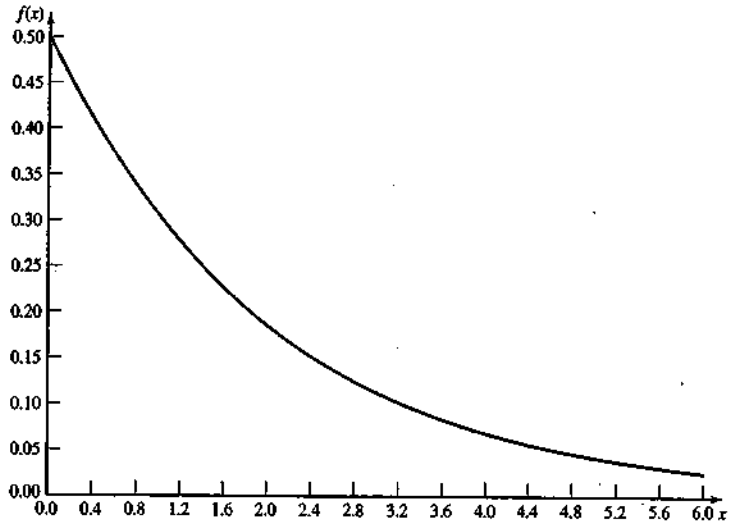


Figure 5.3 pdf for inspection-device life.

This pdf is shown graphically in Figure 5.3. The random variable X is said to have an exponential distribution with mean 2 years.

The probability that the life of the device is between 2 and 3 years is calculated as

$$P(2 \le X \le 3) = \frac{1}{2}\int_2^3 e^{-x/2}dx$$
$$= -e^{-3/2} + e^{-1} = -0.223 + 0.368 = 0.145$$

3. *Cumulative distribution function.* The cumulative distribution function (cdf), denoted by $F(x)$, measures the probability that the random variable X assumes a value less than or equal to x, that is, $F(x) = P(X \le x)$.

If X is discrete, then

$$F(x) = \sum_{\substack{\text{all} \\ x_i \le x}} p(x_i) \tag{5.3}$$

If X is continuous, then

$$F(x) = \int_{-\infty}^{x} f(t)dt \tag{5.4}$$

Some properties of the cdf are listed here:

a. F is a nondecreasing function. If $a < b$, then $F(a) \le F(b)$.
b. $\lim_{x \to \infty} F(x) = 1$
c. $\lim_{x \to -\infty} F(x) = 0$

All probability questions about X can be answered in terms of the cdf. For example,

$$P(a < X \le b) = F(b) - F(a) \quad \text{for all } a < b \tag{5.5}$$

For continuous distributions, not only does Equation (5.5) hold, but also the probabilities in Equation (5.2) are equal to $F(b) - F(a)$.

Example 5.4

The die-tossing experiment described in Example 5.2 has a cdf given as follows:

x	$(-\infty, 1)$	$[1, 2)$	$[2, 3)$	$[3, 4)$	$[4, 5)$	$[5, 6)$	$[6, \infty)$
$F(x)$	0	1/21	3/21	6/21	10/21	15/21	21/21

where $[a, b) = \{a \le x < b\}$. The cdf for this example is shown graphically in Figure 5.4.

If X is a discrete random variable with possible values $x_1, x_2, \ldots$, where $x_1 < x_2 < \ldots$, the cdf is a step function. The value of the cdf is constant in the interval $[x_{i-1}, x_i)$ and then takes a step, or jump, of size $p(x_i)$ at x_i. Thus, in Example 5.4, $p(3) = 3/21$, which is the size of the step when $x = 3$.

Example 5.5

The cdf for the device described in Example 5.3 is given by

$$F(x) = \frac{1}{2}\int_0^x e^{-x/2}dt = 1 - e^{-x/2}$$

The probability that the device will last for less than 2 years is given by

$$P(0 \le X \le 2) = F(2) - F(0) = F(2) = 1 - e^{-1} = 0.632$$

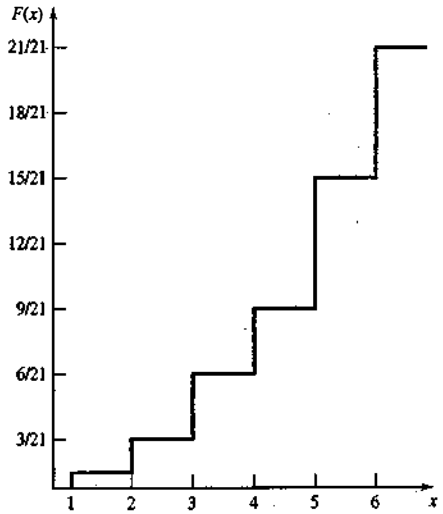


Figure 5.4 cdf for loaded-die example.

The probability that the life of the device is between 2 and 3 years is calculated as

$$P(2 \le X \le 3) = F(3) - F(2) = (1 - e^{-3/2}) - (1 - e^{-1})$$
$$= -e^{-3/2} + e^{-1} = -0.223 + 0.368 = 0.145$$

as found in Example 5.3.

4. *Expectation.* An important concept in probability theory is that of the expectation of a random variable. If X is a random variable, the expected value of X, denoted by $E(X)$, for discrete and continuous variables is defined as follows:

$$E(X) = \sum_{\text{all } i} x_i p(x_i) \qquad \text{if } X \text{ is discrete} \tag{5.6}$$

and

$$E(X) = \int_{-\infty}^{\infty} x f(x)\,dx \qquad \text{if } X \text{ is continuous} \tag{5.7}$$

The expected value $E(X)$ of a random variable X is also referred to as the mean, μ, or the first moment of X. The quantity $E(X^n)$, $n \ge 1$, is called the nth moment of X, and is computed as follows:

$$E(X^n) = \sum_{\text{all } i} x_i^n p(x_i) \qquad \text{if } X \text{ is discrete} \tag{5.8}$$

and

$$E(X^n) = \int_{-\infty}^{\infty} x^n f(x)\,dx \qquad \text{if } X \text{ is continuous} \tag{5.9}$$

The variance of a random variable, X, denoted by $V(X)$ or $\text{var}(X)$ or σ^2, is defined by

$$V(X) = E[(X - E[X])^2]$$

A useful identity in computing $V(X)$ is given by

$$V(X) = E(X^2) - [E(X)]^2 \tag{5.10}$$

The mean $E(X)$ is a measure of the central tendency of a random variable. The variance of X measures the expected value of the squared difference between the random variable and its mean. Thus, the variance, $V(X)$, is a measure of the spread or variation of the possible values of X around the mean $E(X)$. The standard deviation, σ, is defined to be the square root of the variance, σ^2. The mean, $E(X)$, and the standard deviation, $\sigma = \sqrt{V(X)}$, are expressed in the same units.

Example 5.6

The mean and variance of the die-tossing experiment described in Example 5.2 are computed as follows:

$$E(X) = 1\left(\frac{1}{21}\right) + 2\left(\frac{2}{21}\right) + \cdots + 6\left(\frac{6}{21}\right) = \frac{91}{21} = 4.33$$

To compute $V(X)$ from Equation (5.10), first compute $E(X^2)$ from Equation (5.8) as follows:

$$E(X^2) = 1^2\left(\frac{1}{21}\right) + 2^2\left(\frac{2}{21}\right) + \cdots + 6^2\left(\frac{6}{21}\right) = 21$$

Thus,

$$V(X) = 21 - \left(\frac{91}{21}\right)^2 = 21 - 18.78 = 2.22$$

and

$$\sigma = \sqrt{V(X)} = 1.49$$

Example 5.7

The mean and variance of the life of the device described in Example 5.3 are computed as follows:

$$E(X) = \frac{1}{2}\int_0^{\infty} x e^{-x/2}\,dx = -x e^{-x/2}\Big|_0^{\infty} + \int_0^{\infty} e^{-x/2}\,dx$$
$$= 0 + \frac{1}{1/2} e^{-x/2}\Big|_0^{\infty} = 2 \text{ years}$$

To compute $V(X)$ from Equation (5.10), first compute $E(X^2)$ from Equation (5.9) as follows:

$$E(X^2) = \frac{1}{2}\int_0^{\infty} x^2 e^{-x/2}\,dx$$

Thus,

$$E(X^2) = -x^2 e^{-x/2}\Big|_0^{\infty} + 2\int_0^{\infty} x e^{-x/2}\,dx = 8$$

giving

$$V(X) = 8 - 2^2 = 4 \text{ years}^2$$

and

$$\sigma = \sqrt{V(X)} = 2 \text{ years}$$

With a mean life of 2 years and a standard deviation of 2 years, most analysts would conclude that actual lifetimes, X, have a fairly large variability.

5. *The mode.* The mode is used in describing several statistical models that appear in this chapter. In the discrete case, the mode is the value of the random variable that occurs most frequently. In the continuous case, the mode is the value at which the pdf is maximized. The mode might not be unique; if the modal value occurs at two values of the random variable, the distribution is said to be bimodal.

5.2 USEFUL STATISTICAL MODELS

Numerous situations arise in the conduct of a simulation where an investigator may choose to introduce probabilistic events. In Chapter 2, queueing, inventory, and reliability examples were given. In a queueing system, interarrival and service times are often probabilistic. In an inventory model, the time between demands and the lead times (time between placing and receiving an order) can be probabilistic. In a reliability model, the time to failure could be probabilistic. In each of these instances, the simulation analyst desires to generate random events and to use a known statistical model if the underlying distribution can be found. In the following

paragraphs, statistical models appropriate to these application areas will be discussed. Additionally, statistical models useful in the case of limited data are mentioned.

1. *Queueing systems.* In Chapter 2, examples of waiting-line problems were given. In Chapters 2, 3, and 4, these problems were solved via simulation. In the queueing examples, interarrival- and service-time patterns were given. In these examples, the times between arrivals and the service times were always probabilistic, as is usually the case. However, it is possible to have a constant interarrival time (as in the case of a line moving at a constant speed in the assembly of an automobile), or a constant service time (as in the case of robotized spot welding on the same assembly line). The following example illustrates how probabilistic interarrival times might occur.

Example 5.8

Mechanics arrive at a centralized tool crib as shown in Table 5.1. Attendants check in and check out the requested tools to the mechanics. The collection of data begins at 10:00 A.M. and continues until 20 different interarrival times are recorded. Rather than record the actual time of day, the absolute time from a given origin could have been computed. Thus, the first mechanic could have arrived at time zero, the second mechanic at time 7:13 (7 minutes, 13 seconds), and so on.

Example 5.9

Another way of presenting interarrival data is to find the number of arrivals per time period. Here, such arrivals occur over approximately 1 1/2 hours; it is convenient to look at 10-minute time intervals for the first 20 mechanics. That is, in the first 10-minute time period, one arrival occurred at 10:05::03. In the second time period, two mechanics arrived, and so on. The results are summarized in Table 5.2. This data could then be plotted in a histogram, as shown in Figure 5.5.

Table 5.1 Arrival Data

Arrival Number	*Arrival (Hour:Minutes::Seconds)*	*Interarrival Time (Minutes::Seconds)*
1	10:05::03	—
2	10:12::16	7::13
3	10:15::48	3::32
4	10:24::27	8::39
5	10:32::19	7::52
6	10:35::43	3::24
7	10:39::51	4::08
8	10:40::30	0::39
9	10:41::17	0::47
10	10:44::12	2::55
11	10:45::47	1::35
12	10:50::47	5::00
13	11:00::05	9::18
14	11:04::58	4::53
15	11:06::12	1::14
16	11:11::23	5::11
17	11:16::31	5::08
18	11:17::18	0::47
19	11:21::26	4::08
20	11:24::43	3::17
21	11:31::19	6::36

Table 5.2 Arrivals in Successive Time Periods

Time Period	*Number of Arrivals*	*Time Period*	*Number of Arrivals*
1	1	6	1
2	2	7	3
3	1	8	3
4	3	9	2
5	4	—	—

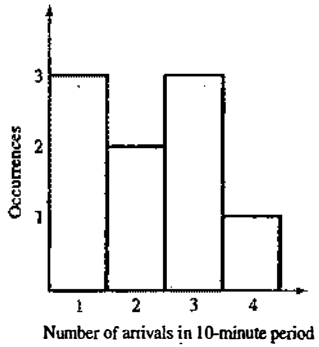


Figure 5.5 Histogram of arrivals per time period.

The distribution of time between arrivals and the distribution of the number of arrivals per time period are important in the simulation of waiting-line systems. "Arrivals" occur in numerous ways; as machine breakdowns, as jobs coming into a jobshop, as units being assembled on a line, as orders to a warehouse, as data packets to a computer system, as calls to a call center, and so on.

Service times could be constant or probabilistic. If service times are completely random, the exponential distribution is often used for simulation purposes; however, there are several other possibilities. It could happen that the service times are constant, but some random variability causes fluctuations in either a positive or a negative way. For example, the time it takes for a lathe to traverse a 10-centimeter shaft should always be the same. However, the material could have slight differences in hardness or the tool might wear; either event could cause different processing times. In these cases, the normal distribution might describe the service time.

A special case occurs when the phenomenon of interest seems to follow the normal probability distribution, but the random variable is restricted to be greater than or less than a certain value. In this case, the truncated normal distribution can be utilized.

The gamma and Weibull distributions are also used to model interarrival and service times. (Actually, the exponential distribution is a special case of both the gamma and the Weibull distributions.) The differences between the exponential, gamma, and Weibull distributions involve the location of the modes of the pdf's and the shapes of their tails for large and small times. The exponential distribution has its mode at the origin, but the gamma and Weibull distributions have their modes at some point (≥ 0) that is a function of the parameter values selected. The tail of the gamma distribution is long, like an exponential distribution; the tail of the Weibull distribution can decline more rapidly or less rapidly than that of an exponential distribution.

In practice, this means that, if there are more large service times than an exponential distribution can account for, a Weibull distribution might provide a better model of these service times.

2. *Inventory and supply-chain systems.* In realistic inventory and supply-chain systems, there are at least three random variables: (1) the number of units demanded per order or per time period, (2) the time between demands, and (3) the lead time. (The lead time is defined as the time between the placing of an order for stocking the inventory system and the receipt of that order.) In very simple mathematical models of inventory systems, demand is a constant over time, and lead time is zero, or a constant. However, in most real-world cases, and, hence, in simulation models, demand occurs randomly in time, and the number of units demanded each time a demand occurs is also random, as illustrated by Figure 5.6.

Distributional assumptions for demand and lead time in inventory theory texts are usually based on mathematical tractability, but those assumptions could be invalid in a realistic context. In practice, the lead-time distribution can often be fitted fairly well by a gamma distribution [Hadley and Whitin, 1963]. Unlike analytic models, simulation models can accommodate whatever assumptions appear most reasonable.

The geometric, Poisson, and negative binomial distributions provide a range of distribution shapes that satisfy a variety of demand patterns. The geometric distribution, which is a special case of the negative binomial, has its mode at unity, given that at least one demand has occurred. If demand data are characterized by a long tail, the negative binomial distribution might be appropriate. The Poisson distribution is often used to model demand because it is simple, it is extensively tabulated, and it is well known. The tail of the Poisson distribution is generally shorter than that of the negative binomial, which means that fewer large demands will occur if a Poisson model is used than if a negative binomial distribution is used (assuming that both models have the same mean demand).

3. *Reliability and maintainability.* Time to failure has been modeled with numerous distributions, including the exponential, gamma, and Weibull. If only random failures occur, the time-to-failure distribution may be modeled as exponential. The gamma distribution arises from modeling standby redundancy, where each component has an exponential time to failure. The Weibull distribution has been extensively used to represent time to failure, and its nature is such that it can be made to approximate many observed phenomena [Hines and Montgomery, 1990]. When there are a number of components in a system and failure is due to

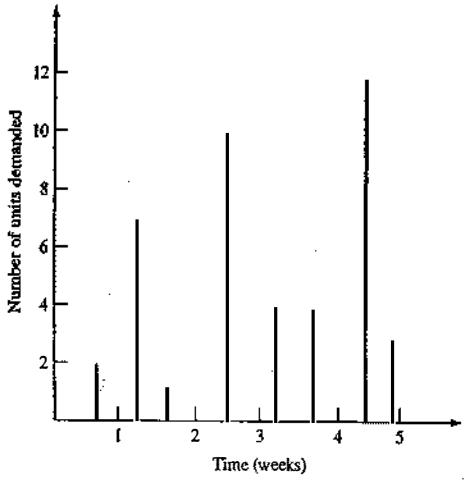


Figure 5.6 Random demands in time.

the most serious of a large number of defects, or possible defects, the Weibull distribution seems to do particularly well as a model. In situations where most failures are due to wear, the normal distribution might very well be appropriate [Hines and Montgomery, 1990]. The lognormal distribution has been found to be applicable in describing time to failure for some types of components.

4. *Limited data.* In many instances, simulations begin before data collection has been completed. There are three distributions that have application to incomplete or limited data. These are the uniform, triangular, and beta distributions. The uniform distribution can be used when an interarrival or service time is known to be random, but no information is immediately available about the distribution [Gordon, 1975]. However, there are those who do not favor using the uniform distribution, calling it the "distribution of maximum ignorance" because it is not necessary to specify more than the continuous interval in which the random variable may occur. The triangular distribution can be used when assumptions are made about the minimum, maximum, and modal values of the random variable. Finally, the beta distribution provides a variety of distributional forms on the unit interval, ones that, with appropriate modification, can be shifted to any desired interval. The uniform distribution is a special case of the beta distribution. Pegden, Shannon, and Sadowski [1995] discuss the subject of limited data in some detail, and we include further discussion in Chapter 9.

5. *Other distributions.* Several other distributions may be useful in discrete-system simulation. The Bernoulli and binomial distributions are two discrete distributions which might describe phenomena of interest. The hyperexponential distribution is similar to the exponential distribution, but its greater variability might make it useful in certain instances.

5.3 DISCRETE DISTRIBUTIONS

Discrete random variables are used to describe random phenomena in which only integer values can occur. Numerous examples were given in Section 5.2—for example, demands for inventory items. Four distributions are described in the following subsections.

1. *Bernoulli trials and the Bernoulli distribution.* Consider an experiment consisting of n trials, each of which can be a success or a failure. Let $X_j = 1$ if the jth experiment resulted in a success, and let $X_j = 0$ if the jth experiment resulted in a failure. The n Bernoulli trials are called a Bernoulli process if the trials are independent, each trial has only two possible outcomes (success or failure), and the probability of a success remains constant from trial to trial. Thus,

$$p(x_1, x_2, \ldots, x_n) = p_1(x_1) \cdot p_2(x_2) \cdots p_n(x_n)$$

and

$$p_j(x_j) = p(x_j) = \begin{cases} p, & x_j = 1, j = 1, 2, \ldots, n \\ 1 - p = q, & x_j = 0, j = 1, 2, \ldots, n \\ 0, & \text{otherwise} \end{cases} \tag{5.11}$$

For one trial, the distribution given in Equation (5.11) is called the Bernoulli distribution. The mean and variance of X_j are calculated as follows:

$$E(X_j) = 0 \cdot q + 1 \cdot p = p$$

and

$$V(X_j) = [(0^2 \cdot q) + (1^2 \cdot p)] - p^2 = p(1 - p)$$

2. *Binomial distribution.* The random variable X that denotes the number of successes in n Bernoulli trials has a binomial distribution given by $p(x)$, where

$$p(x)=\begin{cases}\binom{n}{x}p^x q^{n-x}, & x=0,1,2,\ldots,n\\ 0, & \text{otherwise}\end{cases} \tag{5.12}$$

Equation (5.12) is motivated by computing the probability of a particular outcome with all the successes, each denoted by S, occurring in the first x trials, followed by the $n-x$ failures, each denoted by an F—that is,

$$P(\overbrace{SSS\ldots\ldots SS}^{x\text{ of these}}\overbrace{FF\ldots\ldots FF}^{n-x\text{ of these}})=p^x q^{n-x}$$

where $q=1-p$. There are

$$\binom{n}{x}=\frac{n!}{x!(n-x)!}$$

outcomes having the required number of S's and F's. Therefore, Equation (5.12) results. An easy approach to calculating the mean and variance of the binomial distribution is to consider X as a sum of n independent Bernoulli random variables, each with mean p and variance $p(1-p)=pq$. Then,

$$X=X_1+X_2+\cdots+X_n$$

and the mean, $E(X)$, is given by

$$E(X)=p+p+\cdots+p=np \tag{5.13}$$

and the variance $V(X)$ is given by

$$V(X)=pq+pq+\cdots+pq=npq \tag{5.14}$$

Example 5.10

A production process manufactures computer chips on the average at 2% nonconforming. Every day, a random sample of size 50 is taken from the process. If the sample contains more than two nonconforming chips, the process will be stopped. Compute the probability that the process is stopped by the sampling scheme.

Consider the sampling process as $n=50$ Bernoulli trials, each with $p=0.02$; then the total number of nonconforming chips in the sample, X, would have a binomial distribution given by

$$p(x)=\begin{cases}\binom{50}{x}(0.02)^x(0.98)^{50-x}, & x=0,1,2,\ldots,50\\ 0, & \text{otherwise}\end{cases}$$

It is much easier to compute the right-hand side of the following identity to compute the probability that more than two nonconforming chips are found in a sample:

$$P(X>2)=1-P(X\le 2)$$

The probability $P(X\le 2)$ is calculated from

$$\begin{aligned}P(X\le 2)&=\sum_{x=0}^{2}\binom{50}{x}(0.02)^x(0.98)^{50-x}\\ &=(0.98)^{50}+50(0.02)(0.98)^{49}+1225(0.02)^2(0.98)^{48}\\ &=0.92\end{aligned}$$

Thus, the probability that the production process is stopped on any day, based on the sampling process, is approximately 0.08. The mean number of nonconforming chips in a random sample of size 50 is given by

$$E(X)=np=50(0.02)=1$$

and the variance is given by

$$V(X)=npq=50(0.02)(0.98)=0.98$$

The cdf for the binomial distribution has been tabulated by Banks and Heikes [1984] and others. The tables decrease the effort considerably for computing probabilities such as $P(a<X\le b)$. Under certain conditions on n and p, both the Poisson distribution and the normal distribution may be used to approximate the binomial distribution [Hines and Montgomery, 1990].

3. *Geometric and Negative Binomial distributions.* The geometric distribution is related to a sequence of Bernoulli trials; the random variable of interest, X, is defined to be the number of trials to achieve the first success. The distribution of X is given by

$$p(x)=\begin{cases}q^{x-1}p, & x=1,2,\ldots\\ 0, & \text{otherwise}\end{cases} \tag{5.15}$$

The event $\{X=x\}$ occurs when there are $x-1$ failures followed by a success. Each of the failures has an associated probability of $q=1-p$, and each success has probability p. Thus,

$$P(FFF\cdots FS)=q^{x-1}p$$

The mean and variance are given by

$$E(X)=\frac{1}{p} \tag{5.16}$$

and

$$V(X)=\frac{q}{p^2} \tag{5.17}$$

More generally, the negative binomial distribution is the distribution of the number of trials until the kth success, for $k=1,2,\ldots$. If Y has a negative binomial distribution with parameters p and k, then the distribution of Y is given by

$$p(y)=\begin{cases}\binom{y-1}{k-1}q^{y-k}p^k, & y=k,k+1,k+2,\ldots\\ 0, & \text{otherwise}\end{cases} \tag{5.18}$$

Because we can think of the negative binomial random variable Y as the sum of k independent geometric random variables, it is easy to see that $E(Y)=k/p$ and $V(X)=kq/p^2$.

Example 5.11

Forty percent of the assembled ink-jet printers are rejected at the inspection station. Find the probability that the first acceptable ink-jet printer is the third one inspected. Considering each inspection as a Bernoulli trial with $q = 0.4$ and $p = 0.6$ yields

$$p(3) = 0.4^2(0.6) = 0.096$$

Thus, in only about 10% of the cases is the first acceptable printer the third one from any arbitrary starting point. To determine the probability that the third printer inspected is the second acceptable printer, we use the negative binomial distribution (5.18),

$$p(3) = \binom{3-1}{2-1} 0.4^{3-2}(0.6)^2 = \binom{2}{1} 0.4(0.6)^2 = 0.288$$

4. *Poisson distribution.* The Poisson distribution describes many random processes quite well and is mathematically quite simple. The Poisson distribution was introduced in 1837 by S. D. Poisson in a book concerning criminal and civil justice matters. (The title of this rather old text is "Recherches sur la probabilite des jugements en matiere criminelle et en matiere civile." Evidently, the rumor handed down through generations of probability theory professors concerning the origin of the Poisson distribution is just not true. Rumor has it that the Poisson distribution was first used to model deaths from the kicks of horses in the Prussian Army.)

The Poisson probability mass function is given by

$$p(x) = \begin{cases} \dfrac{e^{-\alpha}\alpha^x}{x!}, & x = 0, 1, \ldots \\ 0, & \text{otherwise} \end{cases} \tag{5.19}$$

where $\alpha > 0$. One of the important properties of the Poisson distribution is that the mean and variance are both equal to α, that is,

$$E(X) = \alpha = V(X)$$

The cumulative distribution function is given by

$$F(x) = \sum_{i=0}^{x} \frac{e^{-\alpha}\alpha^i}{i!} \tag{5.20}$$

The pmf and cdf for a Poisson distribution with $\alpha = 2$ are shown in Figure 5.7. A tabulation of the cdf is given in Table A.4.

Example 5.12

A computer repair person is "beeped" each time there is a call for service. The number of beeps per hour is known to occur in accordance with a Poisson distribution with a mean of $\alpha = 2$ per hour. The probability of three beeps in the next hour is given by Equation (5.19) with $x = 3$, as follows:

$$p(3) = \frac{e^{-2}2^3}{3!} = \frac{(0.135)(8)}{6} = 0.18$$

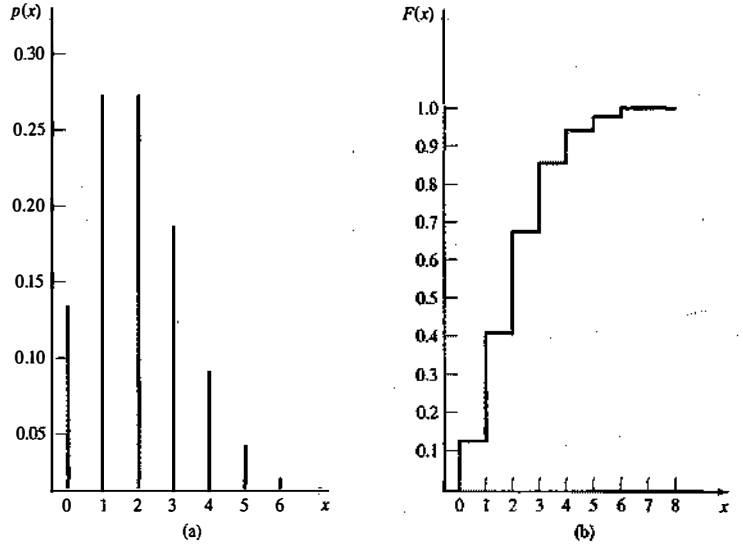


Figure 5.7 Poisson pmf and cdf.

This same result can be read from the left side of Figure 5.7 or from Table A.4 by computing

$$F(3) - F(2) = 0.857 - 0.677 = 0.18$$

Example 5.13

In Example 5.12, find the probability of two or more beeps in a 1-hour period.

$$P(2 \text{ or more}) = 1 - p(0) - p(1) = 1 - F(1)$$
$$= 1 - 0.406 = 0.594$$

The cumulative probability, $F(1)$, can be read from the right side of Figure 5.7 or from Table A.4.

Example 5.14

The lead-time demand in an inventory system is the accumulation of demand for an item from the point at which an order is placed until the order is received—that is,

$$L = \sum_{i=1}^{T} D_i \tag{5.21}$$

where L is the lead-time demand, D_i is the demand during the ith time period, and T is the number of time periods during the lead time. Both D_i and T may be random variables.

An inventory manager desires that the probability of a stockout not exceed a certain fraction during the lead time. For example, it may be stated that the probability of a shortage during the lead time not exceed 5%.

If the lead-time demand is Poisson distributed, the determination of the reorder point is greatly facilitated. The reorder point is the level of inventory at which a new order is placed.

Assume that the lead-time demand is Poisson distributed with a mean of $\alpha = 10$ units and that 95% protection from a stockout is desired. Thus, it is desired to find the smallest value of x such that the probability that the lead-time demand does not exceed x is greater than or equal to 0.95. Using Equation (5.20) requires finding the smallest x such that

$$F(x) = \sum_{i=0}^{x} \frac{e^{-10}10^i}{i!} \geq 0.95$$

The desired result occurs at $x = 15$, which can be found by using Table A.4 or by computation of $p(0), p(1), \ldots$.

5.4 CONTINUOUS DISTRIBUTIONS

Continuous random variables can be used to describe random phenomena in which the variable of interest can take on any value in some interval—for example, the time to failure or the length of a rod. Eight distributions are described in the following subsections.

1. *Uniform distribution.* A random variable X is uniformly distributed on the interval (a, b) if its pdf is given by

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{otherwise} \end{cases} \tag{5.22}$$

The cdf is given by

$$F(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x < b \\ 1, & x \geq b \end{cases} \tag{5.23}$$

Note that

$$P(x_1 < X < x_2) = F(x_2) - F(x_1) = \frac{x_2 - x_1}{b-a}$$

is proportional to the length of the interval, for all x_1 and x_2 satisfying $a \leq x_1 < x_2 \leq b$. The mean and variance of the distribution are given by

$$E(X) = \frac{a+b}{2} \tag{5.24}$$

and

$$V(X) = \frac{(b-a)^2}{12} \tag{5.25}$$

The pdf and cdf when $a = 1$ and $b = 6$ are shown in Figure 5.8.

The uniform distribution plays a vital role in simulation. Random numbers, uniformly distributed between zero and 1, provide the means to generate random events. Numerous methods for generating uniformly distributed random numbers have been devised; some will be discussed in Chapter 7. Uniformly distributed random numbers are then used to generate samples of random variates from all other distributions, as will be discussed in Chapter 8.

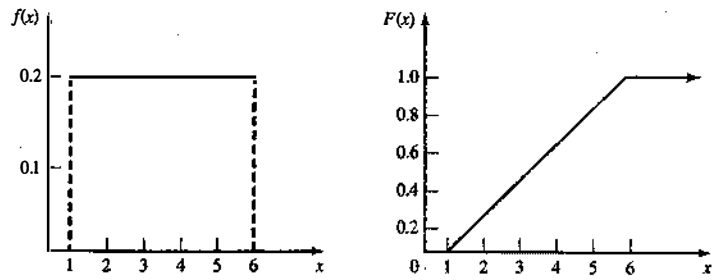


Figure 5.8 pdf and cdf for uniform distribution.

Example 5.15

A simulation of a warehouse operation is being developed. About every 3 minutes, a call comes for a forklift truck operator to proceed to a certain location. An initial assumption is made that the time between calls (arrivals) is uniformly distributed with a mean of 3 minutes. By Equation (5.25), the uniform distribution with a mean of 3 and the greatest possible variability would have parameter values of $a = 0$ and $b = 6$ minutes. With very limited data (such as a mean of approximately 3 minutes) plus the knowledge that the quantity of interest is variable in a random fashion, the uniform distribution with greatest variance can be assumed, at least until more data are available.

Example 5.16

A bus arrives every 20 minutes at a specified stop beginning at 6:40 A.M. and continuing until 8:40 A.M. A certain passenger does not know the schedule, but arrives randomly (uniformly distributed) between 7:00 A.M. and 7:30 A.M. every morning. What is the probability that the passenger waits more than 5 minutes for a bus?

The passenger has to wait more than 5 minutes only if the arrival time is between 7:00 A.M. and 7:15 A.M. or between 7:20 A.M. and 7:30 A.M. If X is a random variable that denotes the number of minutes past 7:00 A.M. that the passenger arrives, the desired probability is

$$P(0 < X < 15) + P(20 < X < 30)$$

Now, X is a uniform random variable on (0,30). Therefore, the desired probability is given by

$$F(15) + F(30) - F(20) = \frac{15}{30} + 1 - \frac{20}{30} = \frac{5}{6}$$

2. *Exponential distribution.* A random variable X is said to be exponentially distributed with parameter $\lambda > 0$ if its pdf is given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & \text{elsewhere} \end{cases} \tag{5.26}$$

The density function is shown in Figures 5.9 and 5.3. Figure 5.9 also shows the cdf.

The exponential distribution has been used to model interarrival times when arrivals are completely random and to model service times that are highly variable. In these instances, λ is a rate: arrivals per hour

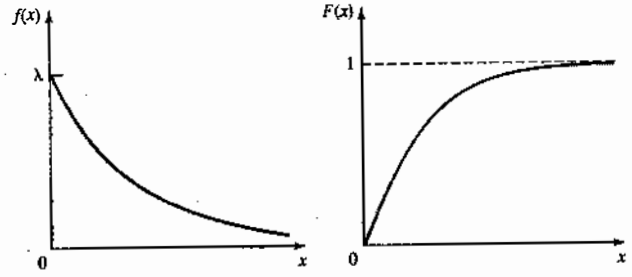


Figure 5.9 Exponential density function and cumulative distribution function.

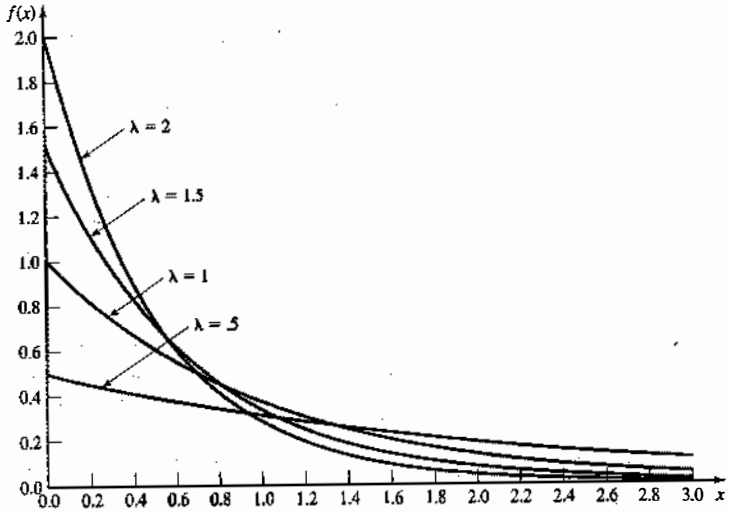


Figure 5.10 pdfs for several exponential distributions.

or services per minute. The exponential distribution has also been used to model the lifetime of a component that fails catastrophically (instantaneously), such as a light bulb; then λ is the failure rate.

Several different exponential pdf's are shown in Figure 5.10. The value of the intercept on the vertical axis is always equal to the value of λ. Note also that all pdf's eventually intersect. (Why?)

The exponential distribution has mean and variance given by

$$E(X)=\frac{1}{\lambda} \quad \text{and} \quad V(X)=\frac{1}{\lambda^2} \tag{5.27}$$

Thus, the mean and standard deviation are equal. The cdf can be exhibited by integrating Equation (5.26) to obtain

$$F(x)=\begin{cases}0, & x<0\\ \int_0^x \lambda e^{-\lambda t}dt = 1-e^{-\lambda x}, & x\geq 0\end{cases} \tag{5.28}$$

Example 5.17

Suppose that the life of an industrial lamp, in thousands of hours, is exponentially distributed with failure rate $\lambda = 1/3$ (one failure every 3000 hours, on the average). The probability that the lamp will last longer than its mean life, 3000 hours, is given by $P(X > 3) = 1 - P(X \leq 3) = 1 - F(3)$. Equation (5.28) is used to compute $F(3)$, obtaining

$$P(X>3)=1-(1-e^{-3/3})=e^{-1}=0.368$$

Regardless of the value of λ, this result will always be the same! That is, the probability that an exponential random variable is greater than its mean is 0.368, for any value of λ.

The probability that the industrial lamp will last between 2000 and 3000 hours is computed as

$$P(2\leq X\leq 3)=F(3)-F(2)$$

Again, from the cdf given by Equation (5.28),

$$\begin{aligned}F(3)-F(2)&=(1-e^{-3/3})-(1-e^{-2/3})\\&=-0.368+0.513=0.145\end{aligned}$$

One of the most important properties of the exponential distribution is that it is "memoryless," which means that, for all $s \geq 0$ and $t \geq 0$,

$$P(X>s+t|X>s)=P(X>t) \tag{5.29}$$

Let X represent the life of a component (a battery, light bulb, computer chip, laser, etc.) and assume that X is exponentially distributed. Equation (5.29) states that the probability that the component lives for at least $s+t$ hours, given that it has survived s hours, is the same as the initial probability that it lives for at least t hours. If the component is alive at time s (if $X > s$), then the distribution of the remaining amount of time that it survives, namely $X-s$, is the same as the original distribution of a new component. That is, the component does not "remember" that it has already been in use for a time s. A used component is as good as new.

That Equation (5.29) holds is shown by examining the conditional probability

$$P(X>s+t\,|\,X>s)=\frac{P(X>s+t)}{P(X>s)} \tag{5.30}$$

Equation (5.28) can be used to determine the numerator and denominator of Equation (5.30), yielding

$$\begin{aligned}P(X>s+t\,|\,X>s)&=\frac{e^{-\lambda(s+t)}}{e^{-\lambda s}}=e^{-\lambda t}\\&=P(X>t)\end{aligned}$$

Example 5.18

Find the probability that the industrial lamp in Example 5.17 will last for another 1000 hours, given that it is operating after 2500 hours. This determination can be found using Equations (5.29) and (5.28), as follows:

$$P(X>3.5|X>2.5)=P(X>1)=e^{-1/3}=0.717$$

Example 5.18 illustrates the *memoryless* property---namely, that a used component that follows an exponential distribution is as good as a new component. The probability that a new component will have

a life greater than 1000 hours is also equal to 0.717. Stated in general, suppose that a component which has a lifetime that follows the exponential distribution with parameter λ is observed and found to be operating at an arbitrary time. Then, the distribution of the remaining lifetime is also exponential with parameter λ. The exponential distribution is the only continuous distribution that has the memoryless property. (The geometric distribution is the only discrete distribution that possesses the memoryless property.)

3. *Gamma distribution.* A function used in defining the gamma distribution is the gamma function, which is defined for all $\beta > 0$ as

$$\Gamma(\beta) = \int_0^\infty x^{\beta-1} e^{-x} dx \tag{5.31}$$

By integrating Equation (5.31) by parts, it can be shown that

$$\Gamma(\beta) = (\beta - 1)\Gamma(\beta - 1) \tag{5.32}$$

If β is an integer, then, by using $\Gamma(1) = 1$ and applying Equation (5.32), it can be seen that

$$\Gamma(\beta) = (\beta - 1)! \tag{5.33}$$

The gamma function can be thought of as a generalization of the factorial notion to all positive numbers, not just integers.

A random variable X is gamma distributed with parameters β and θ if its pdf is given by

$$f(x) = \begin{cases} \dfrac{\beta\theta}{\Gamma(\beta)}(\beta\theta x)^{\beta-1} e^{-\beta\theta x}, & x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.34}$$

β is called the shape parameter, and θ is called the scale parameter. Several gamma distributions for $\theta = 1$ and various values of β are shown in Figure 5.10a.

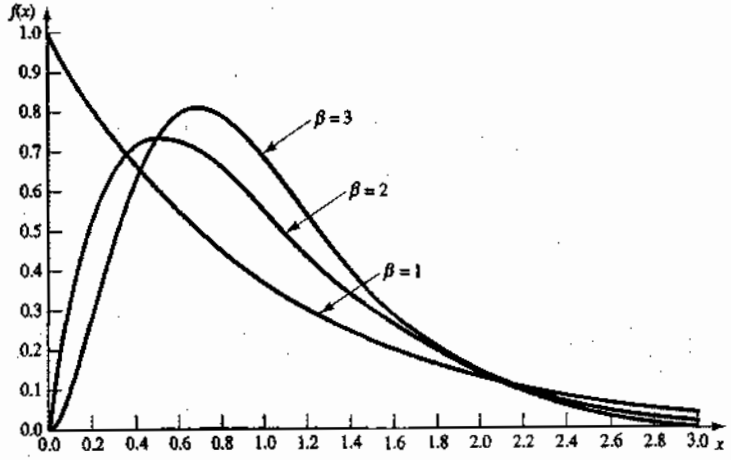


Figure 5.10a

The mean and variance of the gamma distribution are given by

$$E(X) = \frac{1}{\theta} \tag{5.35}$$

and

$$V(X) = \frac{1}{\beta\theta^2} \tag{5.36}$$

The cdf of X is given by

$$F(x) = \begin{cases} 1 - \int_x^\infty \dfrac{\beta\theta}{\Gamma(\beta)}(\beta\theta t)^{\beta-1} e^{-\beta\theta t} dt, & x > 0 \\ 0, & x \le 0 \end{cases} \tag{5.37}$$

When β is an integer, the gamma distribution is related to the exponential distribution in the following manner: If the random variable, X, is the sum of β independent, exponentially distributed random variables, each with parameter $\beta\theta$, then X has a gamma distribution with parameters β and θ. Thus, if

$$X = X_1 + X_2 + \cdots + X_\beta \tag{5.38}$$

where the pdf of X_j is given by

$$g(x_j) = \begin{cases} (\beta\theta)e^{-\beta\theta x_j}, & x \ge 0 \\ 0, & \text{otherwise} \end{cases}$$

and the X_j are mutually independent, then X has the pdf given in Equation (5.34). Note that, when $\beta = 1$, an exponential distribution results. This result follows from Equation (5.38) or from letting $\beta = 1$ in Equation (5.34).

4. *Erlang distribution.* The pdf given by Equation (5.34) is often referred to as the Erlang distribution of order (or number of phases) k when $\beta = k$, an integer. Erlang was a Danish telephone engineer who was an early developer of queueing theory. The Erlang distribution could arise in the following context: Consider a series of k stations that must be passed through in order to complete the servicing of a customer. An additional customer cannot enter the first station until the customer in process has negotiated all the stations. Each station has an exponential distribution of service time with parameter $k\theta$. Equations (5.35) and (5.36), which state the mean and variance of a gamma distribution, are valid regardless of the value of β. However, when $\beta = k$, an integer, Equation (5.38) may be used to derive the mean of the distribution in a fairly straightforward manner. The expected value of the sum of random variables is the sum of the expected value of each random variable. Thus,

$$E(X) = E(X_1) + E(X_2) + \cdots + E(X_k)$$

The expected value of each of the exponentially distributed X_j is given by $1/k\theta$. Thus,

$$E(X) = \frac{1}{k\theta} + \frac{1}{k\theta} + \cdots + \frac{1}{k\theta} = \frac{1}{\theta}$$

If the random variables X_j are independent, the variance of their sum is the sum of the variances, or

$$V(X) = \frac{1}{(k\theta)^2} + \frac{1}{(k\theta)^2} + \cdots + \frac{1}{(k\theta)^2} = \frac{1}{k\theta^2}$$

When $\beta = k$, a positive integer, the cdf given by Equation (5.37) may be integrated by parts, giving

$$F(x) = \begin{cases} 1 - \sum_{i=0}^{k-1} \frac{e^{-k\theta x}(k\theta x)^i}{i!}, & x > 0 \\ 0, & x \le 0 \end{cases} \tag{5.39}$$

which is the sum of Poisson terms with mean $\alpha = k\theta x$. Tables of the cumulative Poisson distribution may be used to evaluate the cdf when the shape parameter is an integer.

Example 5.19

A college professor of electrical engineering is leaving home for the summer, but would like to have a light burning at all times to discourage burglars. The professor rigs up a device that will hold two light bulbs. The device will switch the current to the second bulb if the first bulb fails. The box in which the light bulbs are packaged says, "Average life 1000 hours, exponentially distributed." The professor will be gone 90 days (2160 hours). What is the probability that a light will be burning when the summer is over and the professor returns?

The probability that the system will operate at least x hours is called the reliability function $R(x)$:

$$R(x) = 1 - F(x)$$

In this case, the total system lifetime is given by Equation (5.38) with $\beta = k = 2$ bulbs and $k\theta = 1/1000$ per hour, so $\theta = 1/2000$ per hour. Thus, $F(2160)$ can be determined from Equation (5.39) as follows:

$$F(2160) = 1 - \sum_{i=0}^{1} \frac{e^{-(2)(1/2000)(2160)}[(2)(1/2000)(2160)]^i}{i!}$$

$$= 1 - e^{-2.16} \sum_{i=0}^{1} \frac{(2.16)^i}{i!} = 0.636$$

Therefore, the chances are about 36% that a light will be burning when the professor returns.

Example 5.20

A medical examination is given in three stages by a physician. Each stage is exponentially distributed with a mean service time of 20 minutes. Find the probability that the exam will take 50 minutes or less. Also, compute the expected length of the exam. In this case, $k = 3$ stages and $k\theta = 1/20$, so that $\theta = 1/60$ per minute. Thus, $F(50)$ can be calculated from Equation (5.39) as follows:

$$F(50) = 1 - \sum_{i=0}^{2} \frac{e^{-(3)(1/60)(50)}[(3)(1/60)(50)]^i}{i!}$$

$$= 1 - \sum_{i=0}^{2} \frac{e^{-5/2}(5/2)^i}{i!}$$

The cumulative Poisson distribution, shown in Table A.4, can be used to calculate that

$$F(50) = 1 - 0.543 = 0.457$$

The probability is 0.457 that the exam will take 50 minutes or less. The expected length of the exam is found from Equation (5.35):

$$E(X) = \frac{1}{\theta} = \frac{1}{1/60} = 60 \text{ minutes}$$

In addition, the variance of X is $V(X) = 1/\beta\theta^2 = 1200$ minutes^2—incidentally, the mode of the Erlang distribution is given by

$$\text{Mode} = \frac{k-1}{k\theta} \tag{5.40}$$

Thus, the modal value in this example is

$$\text{Mode} = \frac{3-1}{3(1/60)} = 40 \text{ minutes}$$

5. *Normal distribution.* A random variable X with mean $-\infty < \mu < \infty$ and variance $\sigma^2 > 0$ has a normal distribution if it has the pdf

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], \quad -\infty < x < \infty \tag{5.41}$$

The normal distribution is used so often that the notation $X \sim N(\mu, \sigma^2)$ has been adopted by many authors to indicate that the random variable X is normally distributed with mean μ and variance σ^2. The normal pdf is shown in Figure 5.11.

Some of the special properties of the normal distribution are listed here:

1. $\lim_{x \to -\infty} f(x) = 0$ and $\lim_{x \to +\infty} f(x) = 0$; the value of $f(x)$ approaches zero as x approaches negative infinity and, similarly, as x approaches positive infinity.
2. $f(\mu - x) = f(\mu + x)$; the pdf is symmetric about μ.
3. The maximum value of the pdf occurs at $x = \mu$; the mean and mode are equal.

The cdf for the normal distribution is given by

$$F(x) = P(X \le x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2\right] dt \tag{5.42}$$

It is not possible to evaluate Equation (5.42) in closed form. Numerical methods could be used, but it appears that it would be necessary to evaluate the integral for each pair (μ, σ^2). However, a transformation of

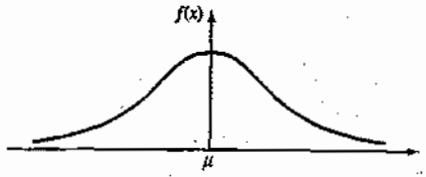


Figure 5.11 pdf of the normal distribution.

variables, $z = (t - \mu)/\sigma$, allows the evaluation to be independent of μ and σ. If $X \sim N(\mu, \sigma^2)$, let $Z = (X - \mu)/\sigma$ to obtain

$$F(x) = P(X \le x) = P\left(Z \le \frac{x-\mu}{\sigma}\right)$$
$$= \int_{-\infty}^{(x-\mu)/\sigma} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz$$
$$= \int_{-\infty}^{(x-\mu)/\sigma} \phi(z) dz = \Phi\left(\frac{x-\mu}{\sigma}\right)$$

The pdf

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}, \quad -\infty < z < \infty \tag{5.44}$$

is the pdf of a normal distribution with mean zero and variance 1. Thus, $Z \sim N(0, 1)$ and it is said that Z has a standard normal distribution. The standard normal distribution is shown in Figure 5.12. The cdf for the standard normal distribution is given by

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \tag{5.45}$$
(5.43)

Equation (5.45) has been widely tabulated. The probabilities $\Phi(z)$ for $Z \ge 0$ are given in Table A.3. Several examples are now given that indicate how Equation (5.43) and Table A.3 are used.

Example 5.21

Suppose that it is known that $X \sim N(50, 9)$. Compute $F(56) = P(X \le 56)$. Using Equation (5.43) get

$$F(56) = \Phi\left(\frac{56-50}{3}\right) = \Phi(2) = 0.9772$$

from Table A.3. The intuitive interpretation is shown in Figure 5.13. Figure 5.13(a) shows the pdf of $X \sim N(50, 9)$ with the specific value, $x_0 = 56$, marked. The shaded portion is the desired probability. Figure 5.13(b) shows the standard normal distribution or $Z \sim N(0, 1)$ with the value 2 marked; $x_0 = 56$ is 2σ (where $\sigma = 3$) greater than the mean. It is helpful to make both sketches such as those in Figure 5.13 to avoid confusion in figuring out required probabilities.

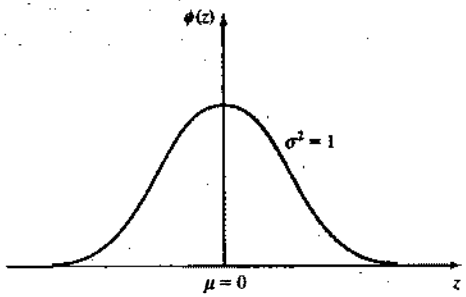


Figure 5.12 pdf of the standard normal distribution.

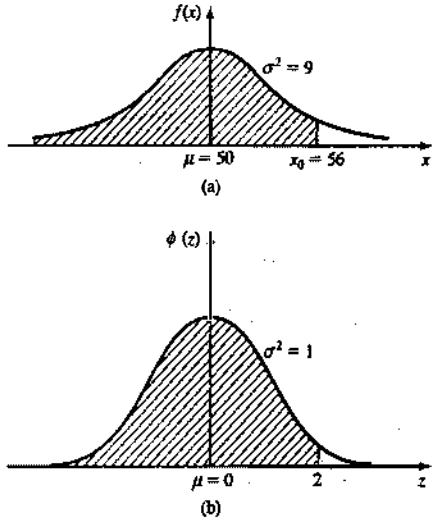


Figure 5.13 Transforming to the standard normal distribution.

Example 5.22

The time in hours required to load an oceangoing vessel, X, is distributed as $N(12,4)$. The probability that the vessel will be loaded in less than 10 hours is given by $F(10)$, where

$$F(10) = \Phi\left(\frac{10-12}{2}\right) = \Phi(-1) = 0.1587$$

The value of $\Phi(-1) = 0.1587$ is looked up in Table A.3 by using the symmetry property of the normal distribution. Note that $\Phi(1) = 0.8413$. The complement of 0.8413, or 0.1587, is contained in the tail, the shaded portion of the standard normal distribution shown in Figure 5.14(a). In Figure 5.14(b), the symmetry property is used to work out the shaded region to be $\Phi(-1) = 1 - \Phi(1) = 0.1587$. [From this logic, it can be seen that $\Phi(2) = 0.9772$ and $\Phi(-2) = 1 - \Phi(2) = 0.0228$. In general, $\Phi(-x) = 1 - \Phi(x)$.]

The probability that 12 or more hours will be required to load the ship can also be discovered by inspection, by using the symmetry property of the normal pdf and the mean as shown by Figure 5.15. The shaded portion of Figure 5.15(a) shows the problem as originally stated [i.e., evaluate $P(X < 12)$]. Now, $P(X > 12) = 1 - F(12)$. The standardized normal in Figure 5.15 (b) is used to evaluate $F(12) = \Phi(0) = 0.50$. Thus, $P(X > 12) = 1 - 0.50 = 0.50$. [The shaded portions in both Figure 5.15(a) and (b) contain 0.50 of the area under the normal pdf.]

The probability that between 10 and 12 hours will be required to load a ship is given by

$$P(10 \le X \le 12) = F(12) - F(10) = 0.5000 - 0.1587 = 0.3413$$

using earlier results presented in this example. The desired area is shown in the shaded portion of Figure 5.16(a). The equivalent problem shown in terms of the standardized normal distribution is shown in Figure 5.16(b). The probability statement is $F(12) - F(10) = \Phi(0) - \Phi(-1) = 0.5000 - 0.1587 - 0.3413$, from Table A.3.

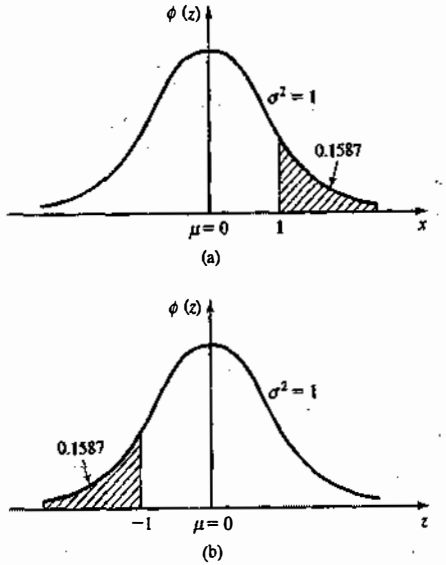


Figure 5.14 Using the symmetry property of the normal distribution.

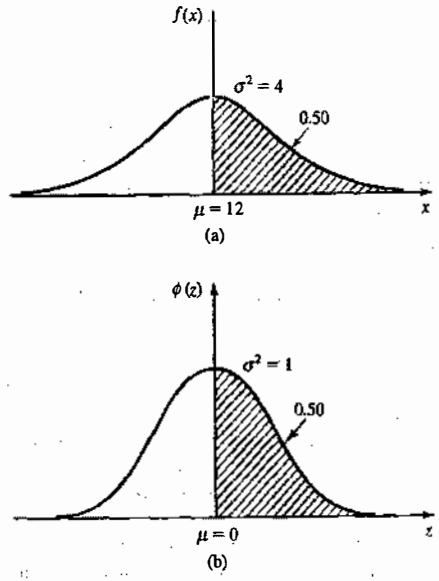


Figure 5.15 Evaluation of probability by inspection.

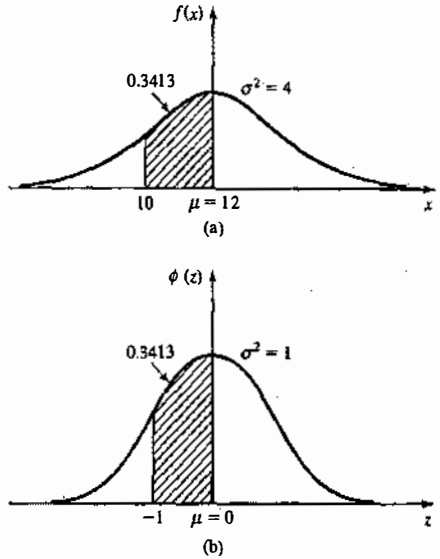


Figure 5.16 Transformation to standard normal for vessel-loading problem.

Example 5.23

The time to pass through a queue to begin self-service at a cafeteria has been found to be $N(15, 9)$. The probability that an arriving customer waits between 14 and 17 minutes is computed as follows:

$$P(14 \le X \le 17) = F(17) - F(14) = \Phi\left(\frac{17-15}{3}\right) - \Phi\left(\frac{14-15}{3}\right)$$
$$= \Phi(0.667) - \Phi(-0.333)$$

The shaded area shown in Figure 5.17(a) represents the probability $F(17) - F(14)$. The shaded area shown in Figure 5.17(b) represents the equivalent probability, $\Phi(0.667) - \Phi(-0.333)$, for the standardized normal distribution. From Table A.3, $\Phi(0.667) = 0.7476$. Now, $\Phi(-0.333) = 1 - \Phi(0.333) = 1 - 0.6304 = 0.3696$. Thus, $\Phi(0.667) - \Phi(-0.333) = 0.3780$. The probability is 0.3780 that the customer will pass through the queue in a time between 14 and 17 minutes.

Example 5.24

Lead-time demand, X, for an item is approximated by a normal distribution having mean 25 and variance 9. It is desired to compute the value for lead time that will be exceeded only 5% of the time. Thus, the problem is to find x_0 such that $P(X > x_0) = 0.05$, as shown by the shaded area in Figure 5.18(a). The equivalent problem is shown as the shaded area in Figure 5.18(b). Now,

$$P(X > x_0) = P\left(Z > \frac{x_0 - 25}{3}\right) = 1 - \Phi\left(\frac{x_0 - 25}{3}\right) = 0.05$$

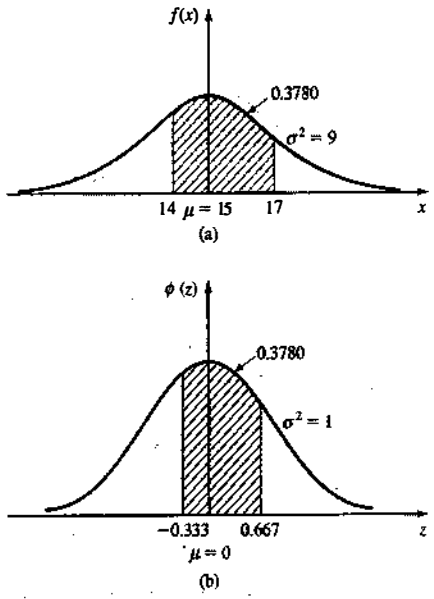


Figure 5.17 Transformation to standard normal for cafeteria problem.

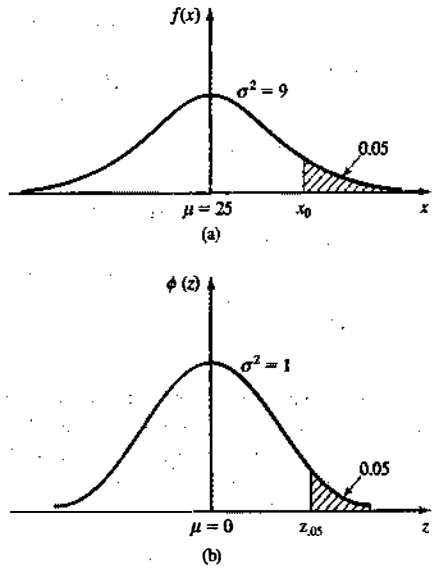


Figure 5.18 Finding x_0 for lead-time–demand problem.

or, equivalently,

$$\Phi\left(\frac{x_0-25}{3}\right)=0.95$$

From Table A.3, it can be seen that $\Phi(1.645)=0.95$. Thus, x_0 can be found by solving

$$\frac{x_0-25}{3}=1.645$$

or

$$x_0=29.935$$

Therefore, in only 5% of the cases will demand during lead time exceed available inventory if an order to purchase is made when the stock level reaches 30.

6. *Weibull distribution.* The random variable X has a Weibull distribution if its pdf has the form

$$f(x)=\begin{cases}\dfrac{\beta}{\alpha}\left(\dfrac{x-\nu}{\alpha}\right)^{\beta-1}\exp\left[-\left(\dfrac{x-\nu}{\alpha}\right)^{\beta}\right], & x\geq\nu\\ 0, & \text{otherwise}\end{cases} \tag{5.46}$$

The three parameters of the Weibull distribution are ν ($-\infty<\nu<\infty$), which is the location parameter; α ($\alpha>0$), which is the scale parameter; and β ($\beta>0$), which is the shape parameter. When $\nu=0$, the Weibull pdf becomes

$$f(x)=\begin{cases}\dfrac{\beta}{\alpha}\left(\dfrac{x}{\alpha}\right)^{\beta-1}\exp\left[-\left(\dfrac{x}{\alpha}\right)^{\beta}\right], & x\geq 0\\ 0, & \text{otherwise}\end{cases} \tag{5.47}$$

Figure 5.19 shows several Weibull densities when $\nu=0$ and $\alpha=1$. When $\beta=1$, the Weibull distribution is reduced to

$$f(x)=\begin{cases}\dfrac{1}{\alpha}e^{-x/\alpha}, & x\geq 0\\ 0, & \text{otherwise}\end{cases}$$

which is an exponential distribution with parameter $\lambda=1/\alpha$.

The mean and variance of the Weibull distribution are given by the following expressions:

$$E(X)=\nu+\alpha\Gamma\left(\frac{1}{\beta}+1\right) \tag{5.48}$$

$$V(X)=\alpha^2\left[\Gamma\left(\frac{2}{\beta}+1\right)-\left[\Gamma\left(\frac{1}{\beta}+1\right)\right]^2\right] \tag{5.49}$$

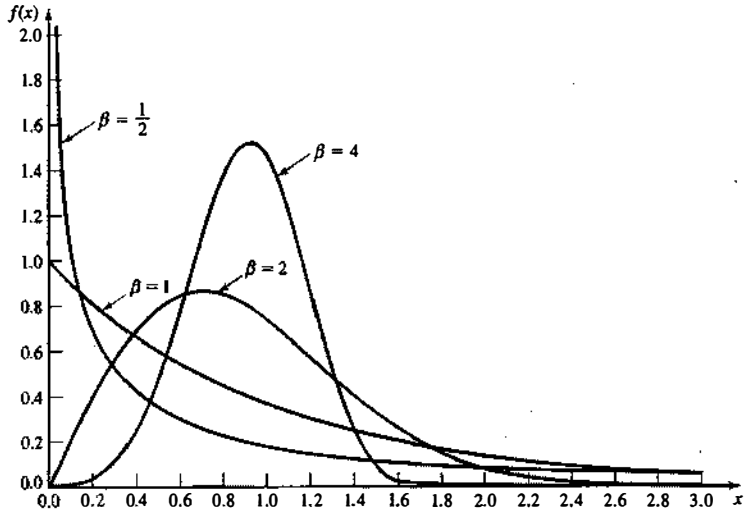


Figure 5.19 Weibull pdfs for $\nu=0;\alpha=\frac{1}{2};\beta=\frac{1}{2},1,2,4$.

where $\Gamma(.)$ is defined by Equation (5.31). Thus, the location parameter, ν, has no effect on the variance; however, the mean is increased or decreased by ν. The cdf of the Weibull distribution is given by

$$F(x)\begin{cases}0, & x<\nu\\ 1-\exp\left[-\left(\dfrac{x-\nu}{\alpha}\right)^{\beta}\right], & x\geq\nu\end{cases} \tag{5.50}$$

Example 5.25

The time to failure for a component screen is known to have a Weibull distribution with $\nu=0$, $\beta=1/3$, and $\alpha=200$ hours. The mean time to failure is given by Equation (5.48) as

$$E(X)=200\Gamma(3+1)=200(3!)=1200 \text{ hours}$$

The probability that a unit fails before 2000 hours is computed from Equation (5.50) as

$$F(2000)=1-\exp\left[-\left(\frac{2000}{200}\right)^{1/3}\right]$$
$$=1-e^{-3\sqrt{10}}=1-e^{-2.15}=0.884$$

Example 5.26

The time it takes for an aircraft to land and clear the runway at a major international airport has a Weibull distribution with $\nu=1.34$ minutes, $\beta=0.5$, and $\alpha=0.04$ minute. Find the probability that an incoming

airplane will take more than 1.5 minutes to land and clear the runway. In this case $P(X>1.5)$ is computed as follows:

$$P(X\leq 1.5)=F(1.5)$$
$$=1-\exp\left[-\left(\frac{1.5-1.34}{0.04}\right)^{0.5}\right]$$
$$=1-e^{-2}=1-0.135=0.865$$

Therefore, the probability that an aircraft will require more than 1.5 minutes to land and clear the runway is 0.135.

7. *Triangular distribution.* A random variable X has a triangular distribution if its pdf is given by

$$f(x)=\begin{cases}\dfrac{2(x-a)}{(b-a)(c-a)}, & a\leq x\leq b\\ \dfrac{2(c-x)}{(c-b)(c-a)}, & b<x\leq c\\ 0, & \text{elsewhere}\end{cases} \tag{5.51}$$

where $a\leq b\leq c$. The mode occurs at $x=b$. A triangular pdf is shown in Figure 5.20. The parameters (a, b, c) can be related to other measures, such as the mean and the mode, as follows:

$$E(X)=\frac{a+b+c}{3} \tag{5.52}$$

From Equation (5.52) the mode can be determined as

$$\text{Mode}=b=3E(X)-(a+c) \tag{5.53}$$

Because $a\leq b\leq c$,

$$\frac{2a+c}{3}\leq E(X)\leq\frac{a+2c}{3}$$

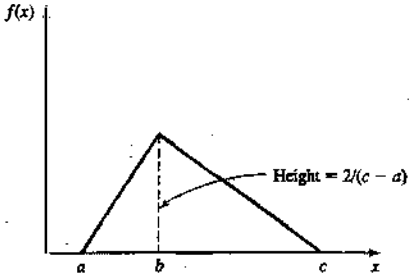


Figure 5.20 pdf of the triangular distribution.

The mode is used more often than the mean to characterize the triangular distribution. As is shown in Figure 5.20, its height is $2/(c-a)$ above the x axis. The variance, $V(X)$, of the triangular distribution is left as an exercise for the student. The cdf for the triangular distribution is given by

$$F(x)=\begin{cases}0, & x\le a\\ \dfrac{(x-a)^2}{(b-a)(c-a)}, & a<x\le b\\ 1-\dfrac{(c-x)^2}{(c-b)(c-a)}, & b<x\le c\\ 1, & x>c\end{cases} \tag{5.54}$$

Example 5.27

The central processing unit requirements, for programs that will execute, have a triangular distribution with $a = 0.05$ millisecond, $b = 1.1$ milliseconds, and $c = 6.5$ milliseconds. Find the probability that the CPU requirement for a random program is 2.5 milliseconds or less. The value of $F(2.5)$ is from the portion of the cdf in the interval (0.05, 1.1) plus that portion in the interval (1.1, 2.5). By using Equation (5.54), both portions can be addressed at one time, to yield

$$F(2.5)=1-\frac{(6.5-2.5)^2}{(6.5-0.05)(6.5-1.1)}=0.541$$

Thus, the probability is 0.541 that the CPU requirement is 2.5 milliseconds or less.

Example 5.28

An electronic sensor evaluates the quality of memory chips, rejecting those that fail. Upon demand, the sensor will give the minimum and maximum number of rejects during each hour of production over the past 24 hours. The mean is also given. Without further information, the quality control department has assumed that the number of rejected chips can be approximated by a triangular distribution. The current dump of data indicates that the minimum number of rejected chips during any hour was zero, the maximum was 10, and the mean was 4. Given that $a = 0$, $c = 10$, and $E(X) = 4$, the value of b can be found from Equation (5.53):

$$b=3(4)-(0+10)=2$$

The height of the mode is $2/(10-0) = 0.2$. Thus, Figure 5.21 can be drawn.

The median is the point at which 0.5 of the area is to the left and 0.5 is to the right. The median in this example is 3.7, also shown on Figure 5.21. Finding the median of the triangular distribution requires an initial location of the value to the left or to the right of the mode. The area to the left of the mode is computed from Equation (5.54) as

$$F(2)=\frac{2^2}{20}=0.2$$

Thus, the median is between b and c. Setting $F(x) = 0.5$ in Equation (5.54) and solving for x = median yields

$$0.5=1-\frac{(10-x)^2}{(10)(8)}$$

with

$$x=3.7$$

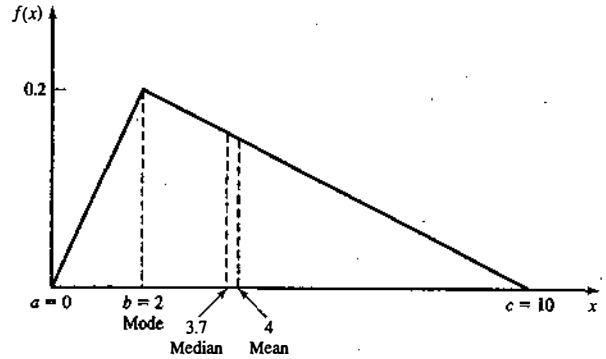


Figure 5.21 Mode, median, and mean for triangular distribution.

This example clearly shows that the mean, mode, and median are not necessarily equal.

8. *Lognormal distribution.* A random variable X has a lognormal distribution if its pdf is given by

$$f(x)=\begin{cases}\dfrac{1}{\sqrt{2\pi}\sigma x}\exp\left[-\dfrac{(\ln x-\mu)^2}{2\sigma^2}\right], & x>0\\ 0, & \text{otherwise}\end{cases} \tag{5.55}$$

where $\sigma^2 > 0$ The mean and variance of a lognormal random variable are

$$E(X)=e^{\mu+\sigma^2/2} \tag{5.56}$$

$$V(X)=e^{2\mu+\sigma^2}(e^{\sigma^2}-1) \tag{5.57}$$

Three lognormal pdf's, all having mean 1, but variances 1/2, 1, and 2, are shown in Figure 5.22.

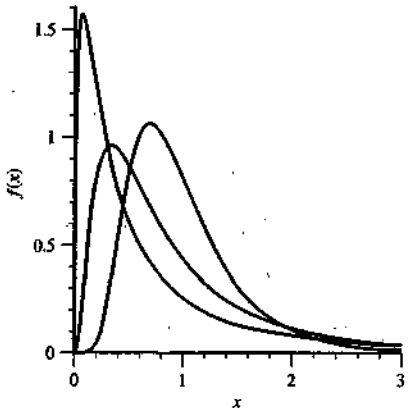


Figure 5.22 pdf of the lognormal distribution.

Notice that the parameters μ and σ^2 are not the mean and variance of the lognormal. These parameters come from the fact that when Y has a $N(\mu, \sigma^2)$ distribution then $X = e^Y$ has a lognormal distribution with parameters μ and σ^2. If the mean and variance of the lognormal are known to be μ_L and σ_L^2, respectively, then the parameters μ and σ^2 are given by

$$\mu = \ln\left(\frac{\mu_L^2}{\sqrt{\mu_L^2 + \sigma_L^2}}\right) \tag{5.58}$$

$$\sigma^2 = \ln\left(\frac{\mu_L^2 + \sigma_L^2}{\mu_L^2}\right) \tag{5.59}$$

Example 5.29

The rate of return on a volatile investment is modeled as having a lognormal distribution with mean 20% and standard deviation 5%. Compute the parameters for the lognormal distribution. From the information given, we have $\mu_L = 20$ and $\sigma_L^2 = 5^2$. Thus, from Equations (5.58) and (5.59),

$$\mu = \ln\left(\frac{20^2}{\sqrt{20^2 + 5^2}}\right) \doteq 2.9654$$

$$\sigma^2 = \ln\left(\frac{20^2 + 5^2}{20^2}\right) \doteq 0.06$$

9. *Beta distribution.* A random variable X is beta-distributed with parameters $\beta_1 > 0$ and $\beta_2 > 0$ if its pdf is given by

$$f(x) = \begin{cases} \dfrac{x^{\beta_1 - 1}(1-x)^{\beta_2 - 1}}{B(\beta_1, \beta_2)}, & 0 < x < 1 \\ 0, & \text{otherwise} \end{cases} \tag{5.60}$$

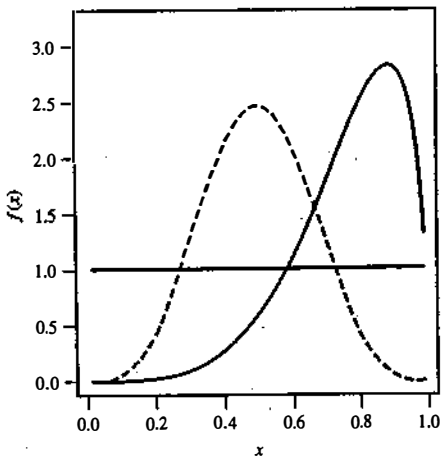


Figure 5.23 The pdf's for several beta distributions.

where $B(\beta_1, \beta_2) = \Gamma(\beta_1)\Gamma(\beta_2)/\Gamma(\beta_1 + \beta_2)$. The cdf of the beta does not have a closed form in general.

The beta distribution is very flexible and has a finite range from 0 to 1, as shown in Figure 5.23. In practice, we often need a beta distribution defined on a different range, say (a, b), with $a < b$, rather than $(0, 1)$. This is easily accomplished by defining a new random variable

$$Y = a + (b - a)X$$

The mean and variance of Y are given by

$$a + (b-a)\left(\frac{\beta_1}{\beta_1 + \beta_2}\right) \tag{5.61}$$

and

$$(b-a)^2\left(\frac{\beta_1\beta_2}{(\beta_1 + \beta_2)^2(\beta_1 + \beta_2 + 1)}\right) \tag{5.62}$$

5.5 POISSON PROCESS

Consider random events such as the arrival of jobs at a job shop, the arrival of e-mail to a mail server, the arrival of boats to a dock, the arrival of calls to a call center, the breakdown of machines in a large factory, and so on. These events may be described by a counting function $N(t)$ defined for all $t \geq 0$. This counting function will represent the number of events that occurred in $[0, t]$. Time zero is the point at which the observation began, regardless of whether an arrival occurred at that instant. For each interval $[0, t]$, the value $N(t)$ is an observation of a random variable where the only possible values that can be assumed by $N(t)$ are the integers 0, 1, 2,....

The counting process, $\{N(t), t \geq 0\}$, is said to be a Poisson process with mean rate λ if the following assumptions are fulfilled:

1. Arrivals occur one at a time.
2. $\{N(t), t \geq 0\}$ has stationary increments: The distribution of the number of arrivals between t and $t + s$ depends only on the length of the interval s, not on the starting point t. Thus, arrivals are completely at random without rush or slack periods.
3. $\{N(t), t \geq 0\}$ has independent increments: The number of arrivals during nonoverlapping time intervals are independent random variables. Thus, a large or small number of arrivals in one time interval has no effect on the number of arrivals in subsequent time intervals. Future arrivals occur completely at random, independent of the number of arrivals in past time intervals.

If arrivals occur according to a Poisson process, meeting the three preceding assumptions, it can be shown that the probability that $N(t)$ is equal to n is given by

$$P[N(t) = n] = \frac{e^{-\lambda t}(\lambda t)^n}{n!} \quad \text{for } t \geq 0 \text{ and } n = 0, 1, 2, \ldots \tag{5.63}$$

Comparing Equation (5.63) to Equation (5.19), it can be seen that $N(t)$ has the Poisson distribution with parameter $\alpha = \lambda t$. Thus, its mean and variance are given by

$$E[N(t)] = \alpha = \lambda t = V[N(t)]$$

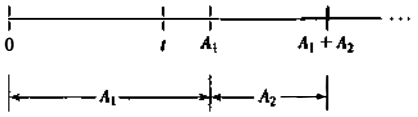


Figure 5.24 Arrival process.

For any times s and t such that $s < t$, the assumption of stationary increments implies that the random variable $N(t) - N(s)$, representing the number of arrivals in the interval from s to t, is also Poisson-distributed with mean $\lambda(t - s)$. Thus,

$$P[N(t)-N(s)=n]=\frac{e^{-\lambda(t-s)}[\lambda(t-s)]^n}{n!} \quad \text{for } n=0, 1, 2,\ldots$$

and

$$E[N(t) - N(s)] = \lambda(t - s) = V[N(t) - N(s)]$$

Now, consider the time at which arrivals occur in a Poisson process. Let the first arrival occur at time A_1, the second occur at time $A_1 + A_2$, and so on, as shown in Figure 5.24. Thus, A_1, A_2,... are successive interarrival times. The first arrival occurs after time t if and only if there are no arrivals in the interval $[0, t]$, so it is seen that

$$\{A_1 > t\} = \{N(t) = 0\}$$

and, therefore,

$$P(A_1 > t) = P[N(t) = 0] = e^{-\lambda t}$$

the last equality following from Equation (5.63). Thus, the probability that the first arrival will occur in $[0, t]$ is given by

$$P(A_1 \le t) = 1 - e^{-\lambda t}$$

which is the cdf for an exponential distribution with parameter λ. Hence, A_1 is distributed exponentially with mean $E(A_1) = 1/\lambda$. It can also be shown that all interarrival times, $A_1, A_2,\ldots$, are exponentially distributed and independent with mean $1/\lambda$. As an alternative definition of a Poisson process, it can be shown that, if interarrival times are distributed exponentially and independently, then the number of arrivals by time t, say $N(t)$, meets the three previously mentioned assumptions and, therefore, is a Poisson process.

Recall that the exponential distribution is memoryless—that is, the probability of a future arrival in a time interval of length s is independent of the time of the last arrival. The probability of the arrival depends only on the length of the time interval, s. Thus, the memoryless property is related to the properties of independent and stationary increments of the Poisson process.

Additional readings concerning the Poisson process may be obtained from many sources, including Parzen [1999], Feller [1968], and Ross [2002].

Example 5.30

The jobs at a machine shop arrive according to a Poisson process with a mean of $\lambda = 2$ jobs per hour. Therefore, the interarrival times are distributed exponentially, with the expected time between arrivals being $E(A) = 1/\lambda = \frac{1}{2}$ hour.

5.5.1 Properties of a Poisson Process

Several properties of the Poisson process, discussed by Ross [2002] and others, are useful in discrete-system simulation. The first of these properties concerns random splitting. Consider a Poisson process $\{N(t), t \ge 0\}$ having rate λ, as represented by the left side of Figure 5.25.

Suppose that, each time an event occurs, it is classified as either a type I or a type II event. Suppose further that each event is classified as a type I event with probability p and type II event with probability $1-p$, independently of all other events.

Let $N_1(t)$ and $N_2(t)$ be random variables that denote, respectively, the number of type I and type II events occurring in $[0, t]$. Note that $N(t) = N_1(t) + N_2(t)$. It can be shown that $N_1(t)$ and $N_2(t)$ are both Poisson processes having rates λp and $\lambda(1 - p)$, as shown in Figure 5.25. Furthermore, it can be shown that the two processes are independent.

Example 5.31: (Random Splitting)

Suppose that jobs arrive at a shop in accordance with a Poisson process having rate λ. Suppose further that each arrival is marked "high priority" with probability 1/3 and "low priority" with probability 2/3. Then a type I event would correspond to a high-priority arrival and a type II event would correspond to a low-priority arrival. If $N_1(t)$ and $N_2(t)$ are as just defined, both variables follow the Poisson process, with rates $\lambda/3$ and $2\lambda/3$, respectively.

Example 5.32

The rate in Example 5.31 is $\lambda = 3$ per hour. The probability that no high-priority jobs will arrive in a 2-hour period is given by the Poisson distribution with parameter $\alpha = \lambda pt = 2$. Thus,

$$P(0)=\frac{e^{-2}2^0}{0!}=0.135$$

Now, consider the opposite situation from random splitting, namely the pooling of two arrival streams. The process of interest is illustrated in Figure 5.26. It can be shown that, if $N_i(t)$ are random variables representing independent Poisson processes with rates λ_i, for $i = 1$ and 2, then $N(t) = N_1(t) + N_2(t)$ is a Poisson process with rate $\lambda_1 + \lambda_2$.

Example 5.33: (Pooled Process)

A Poisson arrival stream with $\lambda_1 = 10$ arrivals per hour is combined (or pooled) with a Poisson arrival stream with $\lambda_2 = 17$ arrivals per hour. The combined process is a Poisson process with $\lambda = 27$ arrivals per hour.

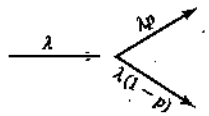


Figure 5.25 Random splitting.

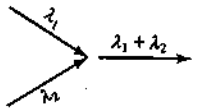


Figure 5.26 Pooled process.

5.5.2 Nonstationary Poisson Process

If we keep the Poisson Assumptions 1 and 3, but drop Assumption 2 (stationary increments) then we have a *nonstationary Poisson process* (NSPP), which is characterized by $\lambda(t)$, the arrival rate at time t. The NSPP is useful for situations in which the arrival rate varies during the period of interest, including meal times for restaurants, phone calls during business hours, and orders for pizza delivery around 6 P.M.

The key to working with a NSPP is the expected number of arrivals by time t, denoted by

$$\Lambda(t) = \int_0^t \lambda(s)ds$$

To be useful as an arrival-rate function, $\lambda(t)$ must be nonnegative and integrable. For a stationary Poisson process with rate λ we have $\Lambda(t) = \lambda t$, as expected.

Let $T_1, T_2, \ldots$ be the arrival times of stationary Poisson process $N(t)$ with $\lambda = 1$, and let $\mathcal{T}_1, \mathcal{T}_2, \ldots$ be the arrival times for a NSPP $\mathcal{N}(t)$ with arrival rate $\lambda(t)$. The fundamental relationship for working with NSPPs is the following:

$$\mathcal{T}_i = \Lambda(T_i)$$
$$T_i = \Lambda^{-1}(\mathcal{T}_i)$$

In words, an NSPP can be transformed into a stationary Poisson process with arrival rate 1, and a stationary Poisson process with arrival rate 1 can be transformed into an NSPP with rate $\lambda(t)$, and the transformation in both cases is related to $\Lambda(t)$.

Example 5.34

Suppose that arrivals to a Post Office occur at a rate of 2 per minute from 8 A.M. until 12 P.M., then drop to 1 every 2 minutes until the day ends at 4 P.M. What is the probability distribution of the number of arrivals between 11 A.M. and 2 P.M?

Let time $t = 0$ correspond to 8 A.M. Then this situation could be modeled as a NSPP $\mathcal{N}(t)$ with rate function

$$\lambda(t) = \begin{cases} 2, & 0 \le t < 4 \\ \frac{1}{2}, & 4 \le t \le 8 \end{cases}$$

The expected number of arrivals by time t is therefore

$$\Lambda(t) = \begin{cases} 2t, & 0 \le t < 4 \\ \frac{t}{2} + 6, & 4 \le t \le 8 \end{cases}$$

Notice that computing the expected number of arrivals for $4 \le t \le 8$ requires that the integration be done in two parts:

$$\Lambda(t) = \int_0^t \lambda(s)\,ds = \int_0^4 2\,ds + \int_4^t \tfrac{1}{2}\,ds = \frac{t}{2} + 6$$

Since 2 P.M. and 11 A.M. correspond to times 6 and 3, respectively, we have

$$\begin{aligned} P[\mathcal{N}(6) - \mathcal{N}(3) = k] &= P[N(\Lambda(6)) - N(\Lambda(3)) = k] \\ &= P[N(9) - N(6) = k] \\ &= \frac{e^{9-6}(9-6)^k}{k!} \\ &= \frac{e^{3}(3)^k}{k!} \end{aligned}$$

where $N(t)$ is a stationary Poisson process with arrival rate 1.

5.6 EMPIRICAL DISTRIBUTIONS

An empirical distribution, which may be either discrete or continuous in form, is a distribution whose parameters are the observed values in a sample of data. This is in contrast to parametric distribution families (such as the exponential, normal, or Poisson), which are characterized by specifying a small number of parameters such as the mean and variance. An empirical distribution may be used when it is impossible or unnecessary to establish that a random variable has any particular parametric distribution. One advantage of an empirical distribution is that nothing is assumed beyond the observed values in the sample; however, this is also a disadvantage because the sample might not cover the entire range of possible values.

Example 5.35: (Discrete)

Customers at a local restaurant arrive at lunchtime in groups of from one to eight persons. The number of persons per party in the last 300 groups has been observed; the results are summarized in Table 5.3. The relative frequencies appear in Table 5.3 and again in Figure 5.27, which provides a histogram of the data that were gathered. Figure 5.28 provides a cdf of the data. The cdf in Figure 5.28 is called the empirical distribution of the given data.

Example 5.36: (Continuous)

The time required to repair a conveyor system that has suffered a failure has been collected for the last 100 instances; the results are shown in Table 5.4. There were 21 instances in which the repair took between 0 and 0.5 hour, and so on. The empirical cdf is shown in Figure 5.29. A piecewise linear curve is formed by the connection of the points of the form $[x, F(x)]$. The points are connected by a straight line. The first connected pair is (0, 0) and (0.5, 0.21); then the points (0.5, 0.21) and (1.0, 0.33) are connected; and so on. More detail on this method is provided in Chapter 8.

Table 5.3 Arrivals per Party Distribution

Arrivals per Party	*Frequency*	*Relative Frequency*	*Cumulative Relative Frequency*
1	30	0.10	0.10
2	110	0.37	0.47
3	45	0.15	0.62
4	71	0.24	0.86
5	12	0.04	0.90
6	13	0.04	0.94
7	7	0.02	0.96
8	12	0.04	1.00

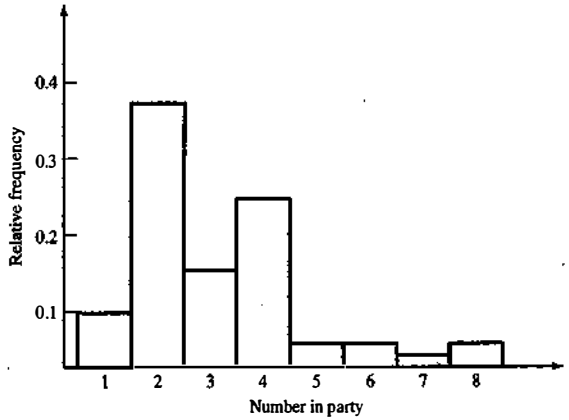


Figure 5.27 Histogram of party size.

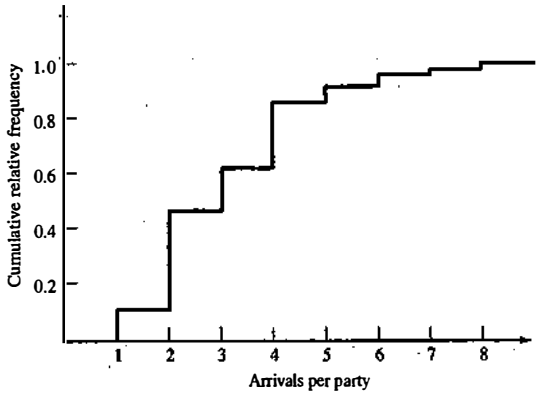


Figure 5.28 Empirical cdf of party size.

Table 5.4 Repair Times for Conveyor

Interval (Hours)	*Relative Frequency*	*Cumulative Frequency*	*Frequency*
$0 < x \le 0.5$	21	0.21	0.21
$0.5 < x \le 1.0$	12	0.12	0.33
$1.0 < x \le 1.5$	29	0.29	0.62
$1.5 < x \le 2.0$	19	0.19	0.81
$2.0 < x \le 2.5$	8	0.08	0.89
$2.5 < x \le 3.0$	11	0.11	1.00

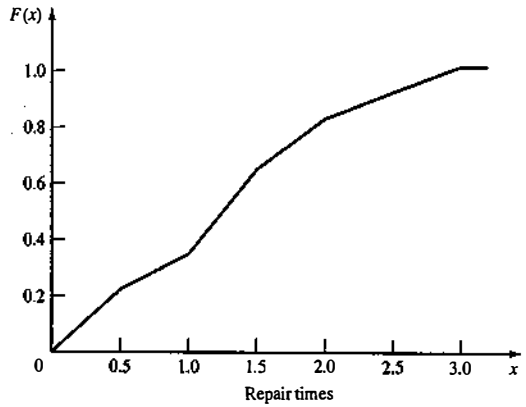


Figure 5.29 Empirical cdf for repair times.

5.7 SUMMARY

In many instances, the world the simulation analyst sees is probabilistic rather than deterministic. The purposes of this chapter were to review several important probability distributions, to familiarize the reader with the notation used in the remainder of the text, and to show applications of the probability distributions in a simulation context.

A major task in simulation is the collection and analysis of input data. One of the first steps in this task is hypothesizing a distributional form for the input data. This is accomplished by comparing the shape of the probability density function or mass function to a histogram of the data and by an understanding that certain physical processes give rise to specific distributions. (Computer software is available to assist in this effort, as will be discussed in Chapter 9.) This chapter was intended to reinforce the properties of various distributions and to give insight into how these distributions arise in practice. In addition, probabilistic models of input data are used in generating random events in a simulation.

Several features that should have made a strong impression on the reader include the differences between discrete, continuous, and empirical distributions; the Poisson process and its properties; and the versatility of the gamma and the Weibull distributions.

REFERENCES

BANKS, J., AND R. G. HEIKES [1984], *Handbook of Tables and Graphs for the Industrial Engineer and Manager*, Reston Publishing, Reston, VA.

DEVORE, J. L. [1999], *Probability and Statistics for Engineers and the Sciences*, 5th ed., Brooks/Cole, Pacific Grove, CA.

FELLER, W. [1968], *An Introduction to Probability Theory and Its Applications*, Vol. I, 3d ed., Wiley, New York.

GORDON, G. [1975], *The Application of GPSS V to Discrete System Simulation*, Prentice-Hall, Englewood Cliffs, NJ.

HADLEY, G., AND T. M. WHITIN [1963], *Analysis of Inventory Systems*, Prentice-Hall, Englewood Cliffs, NJ.

HINES, W. W., AND D. C. MONTGOMERY [1990], *Probability and Statistics in Engineering and Management Science*, 3d ed., Wiley, New York.

LAW, A. M., AND W. D. KELTON [2000], *Simulation Modeling & Analysis*, 3d ed., McGraw–Hill, New York.

PAPOULIS, A. [1990], *Probability and Statistics*, Prentice Hall, Englewood Cliffs, NJ.

PARZEN, E. [1999], *Stochastic Process*, Classics in Applied Mathematics, 24, Society for Industrial & Applied Mathematics, Philadelphia, PA.

PEGDEN, C. D., R. E. SHANNON, AND R. P. SADOWSKI [1995], *Introduction to Simulation Using SIMAN*, 2d ed., McGraw-Hill, New York.

ROSS, S. M. [2002], *Introduction to Probability Models*, 8th ed., Academic Press, New York.

WALPOLE, R. E., AND R. H. MYERS [2002], *Probability and Statistics for Engineers and Scientists*, 7th ed., Prentice Hall, Upper Saddle River, NJ.

EXERCISES

1. Of the orders a job shop receives, 25% are welding jobs and 75% are machining jobs. What is the probability that

   (a) half of the next five jobs will be machining jobs?
   (b) the next four jobs will be welding jobs?

2. Three different items are moving together in a conveyor. These items are inspected visually and defective items are removed. The previous production data are given as

	Item A	*Item B*	*Item C*
Accepted	25	280	190
Rejected	975	720	810

   What is the probability that

   (a) one item is removed at a time?
   (b) two items are removed at a time?
   (c) three items are removed simultaneously?

3. A recent survey indicated that 82% of single women aged 25 years old will be married in their lifetime. Using the binomial distribution, find the probability that two or three women in a sample of twenty will never be married.

4. The Hawks are currently winning 0.55 of their games. There are 5 games in the next two weeks. What is the probability that they will win more games than they lose?

5. Joe Coledge is the third-string quarterback for the University of Lower Alatoona. The probability that Joe gets into any game is 0.40.

   (a) What is the probability that the first game Joe enters is the fourth game of the season?
   (b) What is the probability that Joe plays in no more than two of the first five games?

6. For the random variables X_1 and X_2, which are exponentially distributed with parameter $\lambda = 1$, compute $P(X_1 + X_2 > 2)$.

7. Show that the geometric distribution is memoryless.

8. Hurricane hitting the eastern coast of India follows Poisson with a mean of 0.5 per year. Determine

   (a) the probability of more than three hurricanes hitting the Indian eastern coast in a year.
   (b) the probability of not hitting the Indian eastern coast in a year.

9. Students' arrival at a university library follows Poisson with a mean of 20 per hour. Determine

   (a) the probability that there are 50 arrivals in the next 1 hour.
   (b) the probability that no student arrives in the next 1 hour.
   (c) the probability that there are 75 arrivals in the next 2 hours.

10. Records indicate that 1.8% of the entering students at a large state university drop out of school by midterm. What is the probability that three or fewer students will drop out of a random group of 200 entering students?

11. Lane Braintwain is quite a popular student. Lane receives, on the average, four phone calls a night (Poisson distributed). What is the probability that, tomorrow night, the number of calls received will exceed the average by more than one standard deviation?

12. A car service station receives cars at the rate of 5 every hour in accordance with Poisson. What is the probability that a car will arrive 2 hours after its predecessor?

13. A random variable X that has pmf given by $p(x) = 1/(n+1)$ over the range $R_X = \{0, 1, 2, \ldots, n\}$ is said to have a discrete uniform distribution.

   (a) Find the mean and variance of this distribution. *Hint:*

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \text{ and } \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

   (b) If $R_X = \{a, a+1, a+2, \ldots, b\}$, compute the mean and variance of X.

14. The lifetime, in years, of a satellite placed in orbit is given by the following pdf:

$$f(x) = \begin{cases} 0.4e^{-0.4x}, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

   (a) What is the probability that this satellite is still "alive" after 5 years?
   (b) What is the probability that the satellite dies between 3 and 6 years from the time it is placed in orbit?

15. The cars arriving at a gas station is Poisson distributed with a mean of 10 per minute. Determine the number of pumps to be installed if the firm wants to have 50% of arriving cars as zero entries (i.e., cars serviced without waiting).

16. (The Poisson distribution can be used to approximate the binomial distribution when n is large and p is small—say, p less than 0.1. In utilizing the Poisson approximation, let $\lambda = np$.) In the production of ball bearings, bubbles or depressions occur, rendering the ball bearing unfit for sale. It has been noted that, on the average, one in every 800 of the ball bearings has one or more of these defects. What is the probability that a random sample of 4000 will yield fewer than three ball bearings with bubbles or depressions?

17. For an exponentially distributed random variable X, find the value of λ that satisfies the following relationship:

$$P(X \leq 3) = 0.9P(X \leq 4)$$

18. The time between calls to a fire service station in Chennai follows exponential with a mean of 20 hours. What is the probability that there will be no calls during the next 24 hours?

19. The time to failure of a chip follows exponential with a mean of 5000 hours.

(a) The chip is in operation for the past 1000 hours. What is the probability that the chip will be in operation for another 6000 hours?
(b) After 7000 hours of operation, what is the probability that the chip will not fail for another 2000 hours?

20. The headlight bulb of a car owned by a professor has an exponential time to failure with a mean of 100 weeks. The professor has fitted a new bulb 50 weeks ago. What is the probability that the bulb will not fuse within the next 60 weeks?

21. The service time at the college cafeteria follows exponential with a mean of 2 minutes.

(a) What is the probability that two customers in front of an arriving customer will each take less than 90 seconds to complete their transactions?
(b) What is the probability that two customers in front will finish their transactions so that an arriving customer can reach the service window within 4 minutes?

22. Determine the variance, $V(X)$, of the triangular distribution.

23. The daily demand for rice at a departmental store in thousands of kilogram is found to follow gamma distribution with shape parameter 3 and scale parameter ½. Determine the probability of demand exceeding 5000 kg on a given day.

24. When Admiral Byrd went to the North Pole, he wore battery-powered thermal underwear. The batteries failed instantaneously rather than gradually. The batteries had a life that was exponentially distributed, with a mean of 12 days. The trip took 30 days. Admiral Byrd packed three batteries. What is the probability that three batteries would be a number sufficient to keep the Admiral warm?

25. In an organization's service-complaints mail box, interarrival time of mails are exponentially distributed with a mean of 10 minutes. What is the probability that five mails will arrive in 20 minutes duration?

26. The rail shuttle cars at Atlanta airport have a dual electrical braking system. A rail car switches to the standby system automatically if the first system fails. If both systems fail, there will be a crash! Assume that the life of a single electrical braking system is exponentially distributed, with a mean of 4,000 operating hours. If the systems are inspected every 5,000 operating hours, what is the probability that a rail car will not crash before that time?

27. Suppose that cars arriving at a toll booth follow a Poisson process with a mean interarrival time of 15 seconds. What is the probability that up to one minute will elapse until three cars have arrived?

28. Suppose that an average of 30 customers per hour arrive at the Sticky Donut Shop in accordance with a Poisson process. What is the probability that more than 5 minutes will elapse before both of the next two customers walk through the door?

29. Professor Dipsy Doodle gives six problems on each exam. Each problem requires an average of 30 minutes grading time for the entire class of 15 students. The grading time for each problem is exponentially distributed, and the problems are independent of each other.

(a) What is the probability that the Professor will finish the grading in $2\frac{1}{2}$ hours or less?
(b) What is the most likely grading time?
(c) What is the expected grading time?

30. An aircraft has dual hydraulic systems. The aircraft switches to the standby system automatically if the first system fails. If both systems have failed, the plane will crash. Assume that the life of a hydraulic system is exponentially distributed, with a mean of 2000 air hours.

(a) If the hydraulic systems are inspected every 2500 hours, what is the probability that an aircraft will crash before that time?
(b) What danger would there be in moving the inspection point to 3000 hours?

31. Show that the beta distribution becomes the uniform distribution over the unit interval when $\beta_1 = \beta_2 = 1$.

32. Lead time of a product in weeks is gamma-distributed with shape parameter 2 and scale parameter 1. What is the probability that the lead time exceeds 3 weeks?

33. Lifetime of an inexpensive video card for a PC, in months, denoted by the random variable X, is gamma-distributed with $\beta = 4$ and $\theta = 1/16$. What is the probability that the card will last for at least 2 years?

34. In a statewide competitive examination for engineering admission, the register number allotted to the candidates is of the form CCNNNN, where C is a character like A, B, and C, etc., and N is a number from 0 to 9. Assume that you are scanning through the rank list (based on marks secured in the competitive examination), what is the probability that

(a) the next five entries in the list will have numbers 7000 or higher?
(b) the next three entries will have numbers greater than 3000?

35. Let X be a random variable that is normally distributed, with mean 10 and variance 4. Find the values a and b such that $P(a < X < b) = 0.90$ and $|\mu - a| = |\mu - b|$.

36. Given the following distributions,

Normal (10, 4)
Triangular (4, 10, 16)
Uniform (4, 16)
find the probability that $6 < X < 8$ for each of the distributions.

37. Demand for an item follows normal distribution with a mean of 50 units and a standard deviation of 7 units. Determine the probabilities of demand exceeding 45, 55, and 65 units.

38. The annual rainfall in Chennai is normally distributed with mean 129 cm and standard deviation 32 cm.

(a) What is the probability of getting excess rain (i.e., 140 cm and above) in a given year?
(b) What is the probability of deficient rain (i.e., 80 cm and below) in a given year?

39. Three shafts are made and assembled into a linkage. The length of each shaft, in centimeters, is distributed as follows:

Shaft 1: $N(60, 0.09)$
Shaft 2: $N(40, 0.05)$
Shaft 3: $N(50, 0.11)$

(a) What is the distribution of the length of the linkage?
(b) What is the probability that the linkage will be longer than 150.2 centimeters?

(c) The tolerance limits for the assembly are (149.83, 150.21). What proportion of assemblies are within the tolerance limits? (*Hint:* If $\{X_i\}$ are n independent normal random variables, and if X_i has mean μ_i and variance σ_i^2, then the sum

$$Y = X_1 + X_2 + \cdots + X_n$$

is normal with mean $\sum_{i=1}^{n} \mu_i$ and variance $\sum_{i=1}^{n} \sigma_i^2$.)

40. The circumferences of battery posts in a nickel–cadmium battery are Weibull-distributed with $\nu = 3.25$ centimeters, $\beta = 1/3$, and $\alpha = 0.005$ centimeters.

(a) Find the probability that a battery post chosen at random will have a circumference larger than 3.40 centimeters.
(b) If battery posts are larger than 3.50 centimeters, they will not go through the hole provided; if they are smaller than 3.30 centimeters, the clamp will not tighten sufficiently. What proportion of posts will have to be scrapped for one of these reasons?

41. The time to failure of a nickel–cadmium battery is Weibull distributed with parameters $\nu = 0$, $\beta = 1/4$, and $\alpha = 1/2$ years.

(a) Find the fraction of batteries that are expected to fail prior to 1.5 years.
(b) What fraction of batteries are expected to last longer than the mean life?
(c) What fraction of batteries are expected to fail between 1.5 and 2.5 years?

42. The time required to assemble a component follows triangular distribution with $a = 10$ seconds and $c = 25$ seconds. The median is 15 seconds. Compute the modal value of assembly time.

43. The time to failure (in months) of a computer follows Weibull distribution with location parameter = 0, scale parameter = 2, and shape parameter = 0.35.

(a) What is the mean time to failure?
(b) What is the probability that the computer will fail by 3 months?

44. The consumption of raw material for a fabrication firm follows triangular distribution with minimum of 200 units, maximum of 275 units, and mean of 220 units. What is the median value of raw material consumption?

45. A postal letter carrier has a route consisting of five segments with the time in minutes to complete each segment being normally distributed, with means and variances as shown:

Tennyson Place	$N(38, 16)$
Windsor Parkway	$N(99, 29)$
Knob Hill Apartments	$N(85, 25)$
Evergreen Drive	$N(73, 20)$
Chastain Shopping Center	$N(52, 12)$

In addition to the times just mentioned, the letter carrier must organize the mail at the central office, which activity requires a time that is distributed by $N(90, 25)$. The drive to the starting point of the route requires a time that is distributed $N(10, 4)$. The return from the route requires a time that is distributed $N(15, 4)$. The letter carrier then performs administrative tasks with a time that is distributed $N(30, 9)$.

(a) What is the expected length of the letter carrier's work day?
(b) Overtime occurs after eight hours of work on a given day. What is the probability that the letter carrier works overtime on any given day?

(c) What is the probability that the letter carrier works overtime on two or more days in a six-day week?
(d) What is the probability that the route will be completed within ±24 minutes of eight hours on any given day? (*Hint:* See Exercise 39.)

46. The light used in the operation theater of a hospital has two bulbs. One bulb is sufficient to get the necessary lighting. The bulbs are connected in such a way that when one fails, automatically the other gets switched on. The life of each bulb is exponentially distributed with a mean of 5000 hours and the lives of the bulbs are independent of one another. What is the probability that the combined life of the light is greater than 7000 hours?

47. High temperature in Biloxi, Mississippi on July 21, denoted by the random variable X, has the following probability density function, where X is in degrees F.

$$f(x) = \begin{cases} \dfrac{2(x-85)}{119}, & 85 \le x \le 92 \\ \dfrac{2(102-x)}{170}, & 92 < x \le 102 \\ 0, & \text{otherwise} \end{cases}$$

(a) What is the variance of the temperature, $V(X)$? (If you worked Exercise 22, this is quite easy.)
(b) What is the median temperature?
(c) What is the modal temperature?

48. The time to failure of Eastinghome light bulbs is Weibull distributed with $\nu = 1.8 \times 10^3$ hours, $\beta = 1/2$, and $\alpha = 1/3 \times 10^3$ hours.

(a) What fraction of bulbs are expected to last longer than the mean lifetime?
(b) What is the median lifetime of a light bulb?

49. Let time $t = 0$ correspond to 6 A.M., and suppose that the arrival rate (in arrivals per hour) of customers to a breakfast restaurant that is open from 6 to 9 A.M. is

$$\lambda(t) = \begin{cases} 30, & 0 \le t < 1 \\ 45, & 1 \le t < 2 \\ 20, & 2 \le t \le 4 \end{cases}$$

Assuming a NSPP model is appropriate, do the following: (a) Derive $\Lambda(t)$. (b) Compute the expected number of arrivals between 6:30 and 8:30 A.M. (c) Compute the probability that there are fewer than 60 arrivals between 6:30 and 8:30 A.M.

6

Queueing Models

Simulation is often used in the analysis of queueing models. In a simple but typical queueing model, shown in Figure 6.1, customers arrive from time to time and join a *queue* (waiting line), are eventually served, and finally leave the system. The term "customer" refers to any type of entity that can be viewed as requesting "service" from a system. Therefore, many service facilities, production systems, repair and maintenance facilities, communications and computer systems, and transport and material-handling systems can be viewed as queueing systems.

Queueing models, whether solved mathematically or analyzed through simulation, provide the analyst with a powerful tool for designing and evaluating the performance of queueing systems. Typical measures of system performance include server utilization (percentage of time a server is busy), length of waiting lines, and delays of customers. Quite often, when designing or attempting to improve a queueing system, the analyst (or decision maker) is involved in tradeoffs between server utilization and customer satisfaction in terms of line lengths and delays. Queueing theory and simulation analysis are used to predict these measures of system performance as a function of the input parameters. The input parameters include the arrival rate of customers, the service demands of customers, the rate at which a server works, and the number and

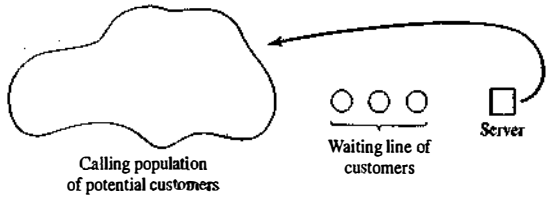


Figure 6.1 Simple queueing model.

arrangement of servers. To a certain degree, some of the input parameters are under management's direct control. Consequently, the performance measures could be under their indirect control, provided that the relationship between the performance measures and the input parameters is adequately understood for the given system.

For relatively simple systems, these performance measures can be computed mathematically—at great savings in time and expense as compared with the use of a simulation model—but, for realistic models of complex systems, simulation is usually required. Nevertheless, analytically tractable models, although usually requiring many simplifying assumptions, are valuable for rough-cut estimates of system performance. These rough-cut estimates may then be refined by use of a detailed and more realistic simulation model. Simple models are also useful for developing an understanding of the dynamic behavior of queueing systems and the relationships between various performance measures. This chapter will not develop the mathematical theory of queues but instead will discuss some of the well-known models. For an elementary treatment of queueing theory, the reader is referred to the survey chapters in Hillier and Lieberman [2005], Wagner [1975] or Winston [1997]. More extensive treatments with a view toward applications are given by Cooper [1990], Gross and Harris [1997], Hall [1991] and Nelson [1995]. The latter two texts especially emphasize engineering and management applications.

This chapter discusses the general characteristics of queues, the meanings and relationships of the important performance measures, estimation of the mean measures of performance from a simulation, the effect of varying the input parameters, and the mathematical solution of a small number of important and basic queueing models.

6.1 CHARACTERISTICS OF QUEUEING SYSTEMS

The key elements of a queueing system are the customers and servers. The term "customer" can refer to people, machines, trucks, mechanics, patients, pallets, airplanes, e-mail, cases, orders, or dirty clothes—anything that arrives at a facility and requires service. The term "server" might refer to receptionists, repairpersons, mechanics, tool-crib clerks, medical personnel, automatic storage and retrieval machines (e.g., cranes), runways at an airport, automatic packers, order pickers, CPUs in a computer, or washing machines—any resource (person, machine, etc.) that provides the requested service. Although the terminology employed will be that of a customer arriving to a service facility, sometimes the server moves to the customer; for example, a repairperson moving to a broken machine. This in no way invalidates the models but is merely a matter of terminology. Table 6.1 lists a number of different systems together with a subsystem consisting of "arriving customers" and one or more "servers." The remainder of this section describes the elements of a queueing system in more detail.

6.1.1 The Calling Population

The population of potential customers, referred to as the *calling population*, may be assumed to be finite or infinite. For example, consider a bank of five machines that are curing tires. After an interval of time, a machine automatically opens and must be attended by a worker who removes the tire and puts an uncured tire into the machine. The machines are the "customers," who "arrive" at the instant they automatically open. The worker is the "server," who "serves" an open machine as soon as possible. The calling population is finite and consists of the five machines.

In systems with a large population of potential customers, the calling population is usually assumed to be infinite. For such systems, this assumption is usually innocuous and, furthermore, it might simplify the model. Examples of infinite populations include the potential customers of a restaurant, bank, or other similar service facility and also very large groups of machines serviced by a technician. Even though the actual

Table 6.1 Examples of Queueing Systems

System	*Customers*	*Server(s)*
Reception desk	People	Receptionist
Repair facility	Machines	Repairperson
Garage	Trucks	Mechanic
Tool crib	Mechanics	Tool-crib clerk
Hospital	Patients	Nurses
Warehouse	Pallets	Fork-lift Truck
Airport	Airplanes	Runway
Production line	Cases	Case-packer
Warehouse	Orders	Order-picker
Road network	Cars	Traffic light
Grocery	Shoppers	Checkout station
Laundry	Dirty linen	Washing machines/dryers
Job shop	Jobs	Machines/workers
Lumberyard	Trucks	Overhead crane
Sawmill	Logs	Saws
Computer	Jobs	CPU, disk, CDs
Telephone	Calls	Exchange
Ticket office	Football fans	Clerk
Mass transit	Riders	Buses, trains

population could be finite but large, it is generally safe to use infinite population models—provided that the number of customers being served or waiting for service at any given time is a small proportion of the population of potential customers.

The main difference between finite and infinite population models is how the arrival rate is defined. In an infinite population model, the arrival rate (i.e., the average number of arrivals per unit of time) is not affected by the number of customers who have left the calling population and joined the queueing system. When the arrival process is homogeneous over time (e.g., there are no "rush hours"), the arrival rate is usually assumed to be constant. On the other hand, for finite calling-population models, the arrival rate to the queueing system does depend on the number of customers being served and waiting. To take an extreme case, suppose that the calling population has one member, for example, a corporate jet. When the corporate jet is being serviced by the team of mechanics who are on duty 24 hours per day, the arrival rate is zero, because there are no other potential customers (jets) who can arrive at the service facility (team of mechanics). A more typical example is that of the five tire-curing machines serviced by a single worker. When all five are closed and curing a tire, the worker is idle and the arrival rate is at a maximum, but the instant a machine opens and requires service, the arrival rate decreases. At those times when all five are open (so four machines are waiting for service while the worker is attending the other one), the arrival rate is zero; that is, no arrival is possible until the worker finishes with a machine, in which case it returns to the calling population and becomes a potential arrival. It may seem odd that the arrival rate is at its maximum when all five machines are closed. But the arrival rate is defined as the expected number of arrivals in the next unit of time, so it becomes clear that this expectation is largest when all machines could potentially open in the next unit of time.

6.1.2 System Capacity

In many queueing systems, there is a limit to the number of customers that may be in the waiting line or system. For example, an automatic car wash might have room for only 10 cars to wait in line to enter the mechanism. It might be too dangerous (or illegal) for cars to wait in the street. An arriving customer who

finds the system full does not enter but returns immediately to the calling population. Some systems, such as concert ticket sales for students, may be considered as having unlimited capacity, since there are no limits on the number of students allowed to wait to purchase tickets. As will be seen later, when a system has limited capacity, a distinction is made between the arrival rate (i.e., the number of arrivals per time unit) and the effective arrival rate (i.e., the number who arrive and enter the system per time unit).

6.1.3 The Arrival Process

The arrival process for infinite-population models is usually characterized in terms of interarrival times of successive customers. Arrivals may occur at scheduled times or at random times. When at random times, the interarrival times are usually characterized by a probability distribution. In addition, customers may arrive one at a time or in batches. The batch may be of constant size or of random size.

The most important model for random arrivals is the Poisson arrival process. If A_n represents the interarrival time between customer $n-1$ and customer n (A_1 is the actual arrival time of the first customer), then, for a Poisson arrival process, A_n is exponentially distributed with mean $1/\lambda$ time units. The arrival rate is λ customers per time unit. The number of arrivals in a time interval of length t, say $N(t)$, has the Poisson distribution with mean λt customers. For further discussion of the relationship between the Poisson distribution and the exponential distribution, the reader is referred to Section 5.5.

The Poisson arrival process has been employed successfully as a model of the arrival of people to restaurants, drive-in banks, and other service facilities; the arrival of telephone calls to a call center; the arrival of demands, or orders for a service or product; and the arrival of failed components or machines to a repair facility.

A second important class of arrivals is scheduled arrivals, such as patients to a physician's office or scheduled airline flight arrivals to an airport. In this case, the interarrival times $\{A_n, n = 1, 2, \ldots\}$ could be either *constant* or *constant plus or minus a small random amount* to represent early or late arrivals.

A third situation occurs when at least one customer is assumed to always be present in the queue, so that the server is never idle because of a lack of customers. For example, the "customers" may represent raw material for a product, and sufficient raw material is assumed to be always available.

For finite-population models, the arrival process is characterized in a completely different fashion. Define a customer as *pending* when that customer is outside the queueing system and a member of the potential calling population. For example, a tire-curing machine is "pending" when it is closed and curing a tire, and it becomes "not pending" the instant it opens and demands service from the worker. Define a *runtime* of a given customer as the length of time from departure from the queueing system until that customer's next arrival to the queue. Let $A_1^{(i)}, A_2^{(i)}, \ldots$ be the successive runtimes of customer i, and let $S_1^{(i)}, S_2^{(i)}, \ldots$ be the corresponding successive system times; that is, $S_n^{(i)}$ is the total time spent in system by customer i during the nth visit. Figure 6.2 illustrates these concepts for machine 3 in the tire-curing example. The total arrival process is the superposition of the arrival times of all customers. Figure 6.2 shows the first and second arrival of machine 3, but these two times are not necessarily two successive arrivals to the system. For instance,

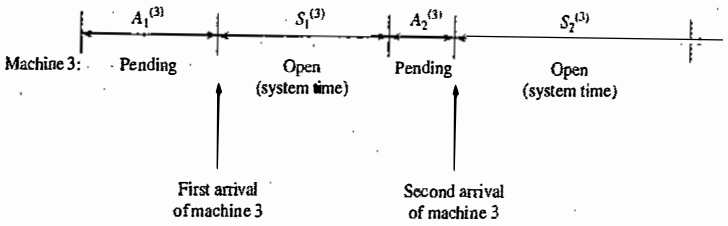


Figure 6.2 Arrival process for a finite-population model.

if it is assumed that all machines are pending at time 0, the first arrival to the system occurs at time $A_1 = \min\{A_1^{(1)}, A_1^{(2)}, A_1^{(3)}, A_1^{(4)}, A_1^{(5)}\}$. If $A_1 = A_1^{(2)}$, then machine 2 is the first arrival (i.e., the first to open) after time 0. As discussed earlier, the arrival rate is not constant but is a function of the number of pending customers.

One important application of finite-population models is the machine-repair problem. The machines are the customers, and a runtime is also called time to failure. When a machine fails, it "arrives" at the queueing system (the repair facility) and remains there until it is "served" (repaired). Times to failure for a given class of machine have been characterized by the exponential, the Weibull, and the gamma distributions. Models with an exponential runtime are sometimes analytically tractable; an example is given in Section 6.5. Successive times to failure are usually assumed to be statistically independent, but they could depend on other factors, such as the age of a machine since its last major overhaul.

6.1.4 Queue Behavior and Queue Discipline

Queue behavior refers to the actions of customers while in a queue waiting for service to begin. In some situations, there is a possibility that incoming customers will balk (leave when they see that the line is too long), renege (leave after being in the line when they see that the line is moving too slowly), or jockey (move from one line to another if they think they have chosen a slow line).

Queue discipline refers to the logical ordering of customers in a queue and determines which customer will be chosen for service when a server becomes free. Common queue disciplines include first-in–first-out (FIFO); last-in–first-out (LIFO); service in random order (SIRO); shortest processing time first (SPT); and service according to priority (PR). In a job shop, queue disciplines are sometimes based on due dates and on expected processing time for a given type of job. Notice that a FIFO queue discipline implies that services begin in the same order as arrivals, but that customers could leave the system in a different order because of different-length service times.

6.1.5 Service Times and the Service Mechanism

The service times of successive arrivals are denoted by S_1, S_2, S_3,... They may be constant or of random duration. In the latter case, $\{S_1, S_2, S_3, \ldots\}$ is usually characterized as a sequence of independent and identically distributed random variables. The exponential, Weibull, gamma, lognormal and truncated normal distributions have all been used successfully as models of service times in different situations. Sometimes services are identically distributed for all customers of a given type or class or priority, whereas customers of different types might have completely different service-time distributions. In addition, in some systems, service times depend upon the time of day or upon the length of the waiting line. For example, servers might work faster than usual when the waiting line is long, thus effectively reducing the service times.

A queueing system consists of a number of service centers and interconnecting queues. Each service center consists of some number of servers, c, working in parallel; that is, upon getting to the head of the line, a customer takes the first available server. Parallel service mechanisms are either single server ($c = 1$), multiple server ($1 < c < \infty$), or unlimited servers ($c = \infty$). A self-service facility is usually characterized as having an unlimited number of servers.

Example 6.1

Consider a discount warehouse where customers may either serve themselves or wait for one of three clerks, then finally leave after paying a single cashier. The system is represented by the flow diagram in Figure 6.3. The subsystem, consisting of queue 2 and service center 2, is shown in more detail in Figure 6.4. Other variations of service mechanisms include batch service (a server serving several customers simultaneously), and a customer's requiring several servers simultaneously. In the discount warehouse, a clerk might pick several small orders at the same time, but it may take two of the clerks to handle one heavy item.

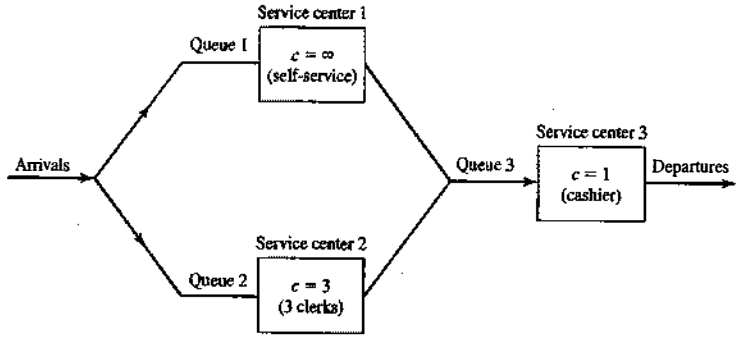


Figure 6.3 Discount warehouse with three service centers.

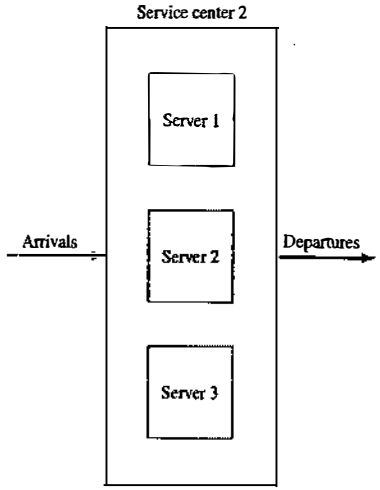


Figure 6.4 Service center 2, with $c = 3$ parallel servers.

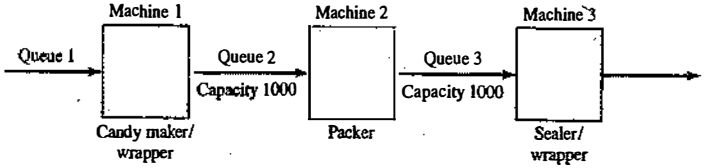


Figure 6.5 Candy-production line.

Example 6.2

A candy manufacturer has a production line that consists of three machines separated by inventory-in-proces buffers. The first machine makes and wraps the individual pieces of candy, the second packs 50 pieces in box, and the third machine seals and wraps the box. The two inventory buffers have capacities of 1000 boxe

each. As illustrated by Figure 6.5, the system is modeled as having three service centers, each center having $c = 1$ server (a machine), with queue capacity constraints between machines. It is assumed that a sufficient supply of raw material is always available at the first queue. Because of the queue capacity constraints, machine 1 shuts down whenever its inventory buffer (queue 2) fills to capacity, and machine 2 shuts down whenever its buffer empties. In brief, the system consists of three single-server queues in series with queue capacity constraints and a contisnuous arrival stream at the first queue.

6.2 QUEUEING NOTATION

Recognizing the diversity of queueing systems, Kendall [1953] proposed a notational system for parallel server systems which has been widely adopted. An abridged version of this convention is based on the format $A/B/c/N/K$. These letters represent the following system characteristics:

A represents the interarrival-time distribution.

B represents the service-time distribution.

c represents the number of parallel servers.

N represents the system capacity.

K represents the size of the calling population.

Common symbols for A and B include M (exponential or Markov), D (constant or deterministic), E_k (Erlang of order k), PH (phase-type), H (hyperexponential), G (arbitrary or general), and GI (general independent).

For example, $M/M/1/\infty/\infty$ indicates a single-server system that has unlimited queue capacity and an infinite population of potential arrivals. The interarrival times and service times are exponentially distributed. When N and K are infinite, they may be dropped from the notation. For example, $M/M/1/\infty/\infty$ is often shortened to $M/M/1$. The tire-curing system can be initially represented by $G/G/1/5/5$.

Additional notation used throughout the remainder of this chapter for parallel server systems is listed in Table 6.2. The meanings may vary slightly from system to system. All systems will be assumed to have a FIFO queue discipline.

Table 6.2 Queueing Notation for Parallel Server Systems

P_n	Steady-state probability of having n customers in system
$P_n(t)$	Probability of n customers in system at time t
λ	Arrival rate
λ_e	Effective arrival rate
μ	Service rate of one server
ρ	Server utilization
A_n	Interarrival time between customers $n-1$ and n
S_n	Service time of the nth arriving customer
W_n	Total time spent in system by the nth arriving customer
W_n^Q	Total time spent in the waiting line by customer n
$L(t)$	The number of customers in system at time t
$L_Q(t)$	The number of customers in queue at time t
L	Long-run time-average number of customers in system
L_Q	Long-run time-average number of customers in queue
w	Long-run average time spent in system per customer
w_Q	Long-run average time spent in queue per customer

6.3 LONG-RUN MEASURES OF PERFORMANCE OF QUEUEING SYSTEMS

The primary long-run measures of performance of queueing systems are the long-run time-average number of customers in the system (L) and in the queue (L_Q), the long-run average time spent in system (w) and in the queue (w_Q) per customer, and the server utilization, or proportion of time that a server is busy (ρ). The term "system" usually refers to the waiting line plus the service mechanism, but, in general, can refer to any subsystem of the queueing system; whereas the term "queue" refers to the waiting line alone. Other measures of performance of interest include the long-run proportion of customers who are delayed in queue longer than t_0 time units, the long-run proportion of customers turned away because of capacity constraints, and the long-run proportion of time the waiting line contains more than k_0 customers.

This section defines the major measures of performance for a general $G/G/c/N/K$ queueing system, discusses their relationships, and shows how they can be estimated from a simulation run. There are two types of estimators: an ordinary sample average, and a time-integrated (or time-weighted) sample average.

6.3.1 Time-Average Number in System L

Consider a queueing system over a period of time T, and let $L(t)$ denote the number of customers in the system at time t. A simulation of such a system is shown in Figure 6.6.

Let T_i denote the total time during $[0, T]$ in which the system contained exactly i customers. In Figure 6.6, it is seen that $T_0 = 3$, $T_1 = 12$, $T_2 = 4$, and $T_3 = 1$. (The line segments whose lengths total $T_1 = 12$ are labelled "T_1" in Figure 6.6, etc.) In general, $\sum_{i=0}^{\infty} T_i = T$. The time-weighted-average number in a system is defined by

$$\hat{L} = \frac{1}{T}\sum_{i=0}^{\infty} iT_i = \sum_{i=0}^{\infty} i\left(\frac{T_i}{T}\right) \tag{6.1}$$

For Figure 6.6, $\hat{L} = [0(3)+1(12)+2(4)+3(1)]/20 = 23/20 = 1.15$ customers. Notice that T_i/T is the proportion of time the system contains exactly i customers. The estimator $\hat{L}$ is an example of a time-weighted average.

By considering Figure 6.6, it can be seen that the total area under the function $L(t)$ can be decomposed into rectangles of height i and length T_i. For example, the rectangle of area $3 \times T_3$ has base running from

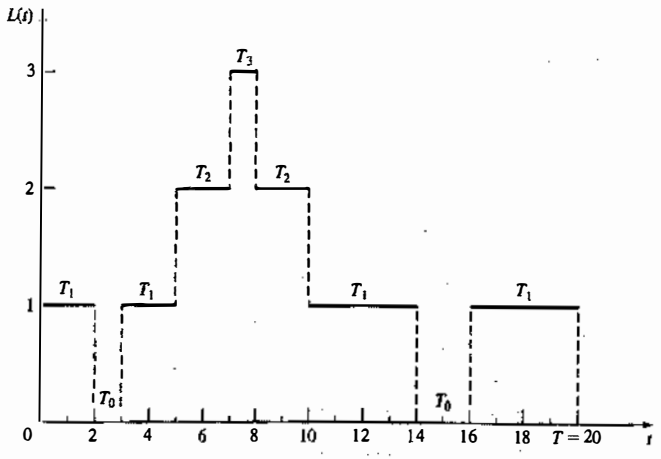


Figure 6.6 Number in system, $L(t)$, at time t.

$t = 7$ to $t = 8$ (thus $T_3 = 1$); however, most of the rectangles are broken into parts, such as the rectangle of area $2 \times T_2$ which has part of its base between $t = 5$ and $t = 7$ and the remainder from $t = 8$ to $t = 10$ (thus $T_2 = 2 + 2 = 4$). It follows that the total area is given by $\sum_{i=0}^{\infty} iT_i = \int_0^T L(t)\,dt$, and, therefore, that

$$\hat{L} = \frac{1}{T}\sum_{i=0}^{\infty} iT_i = \frac{1}{T}\int_0^T L(t)dt \tag{6.2}$$

The expressions in Equations (6.1) and (6.2) are always equal for any queueing system, regardless of the number of servers, the queue discipline, or any other special circumstances. Equation (6.2) justifies the terminology *time-integrated average.*

Many queueing systems exhibit a certain kind of long-run stability in terms of their average performance. For such systems, as time T gets large, the observed time-average number in the system $\hat{L}$ approaches a limiting value, say L, which is called the long-run time-average number in system—that is, with probability 1,

$$\hat{L} = \frac{1}{T}\int_0^T L(t)dt \to L \text{ as } T \to \infty \tag{6.3}$$

The estimator $\hat{L}$ is said to be strongly consistent for L. If simulation run length T is sufficiently long, the estimator $\hat{L}$ becomes arbitrarily close to L. Unfortunately, for $T < \infty$, $\hat{L}$ depends on the initial conditions at time 0.

Equations (6.2) and (6.3) can be applied to any subsystem of a queueing system as well as they can to the whole system. If $L_Q(t)$ denotes the number of customers waiting in line, and T_i^Q denotes the total time during $[0, T]$ in which exactly i customers are waiting in line, then

$$\hat{L}_Q = \frac{1}{T}\sum_{i=0}^{\infty} iT_i^Q = \frac{1}{T}\int_0^T L_Q(t)dt \to L_Q \text{ as } T \to \infty \tag{6.4}$$

where $\hat{L}_Q$ is the observed time-average number of customers waiting in line from time 0 to time T and L_Q is the long-run time-average number waiting in line.

Example 6.3

Suppose that Figure 6.6 represents a single-server queue—that is, a $G/G/1/N/K$ queueing system ($N \geq 3$, $K \geq 3$). Then the number of customers waiting in line is given by $L_Q(t)$ defined by

$$L_Q(t) = \begin{cases} 0 & \text{if } L(t) = 0 \\ L(t) - 1 & \text{if } L(t) \geq 1 \end{cases}$$

and shown in Figure 6.7. Thus, $T_0^Q = 5 + 10 = 15$, $T_1^Q = 2 + 2 = 4$, and $T_2^Q = 1$, Therefore,

$$\hat{L}_Q = \frac{0(15) + 1(4) + 2(1)}{20} = 0.3 \text{ customers}$$

6.3.2 Average Time Spent in System Per Customer w

If we simulate a queueing system for some period of time, say T, then we can record the time each customer spends in the system during $[0, T]$, say $W_1, W_2, \ldots, W_N$, where N is the number of arrivals during $[0, T]$. The average

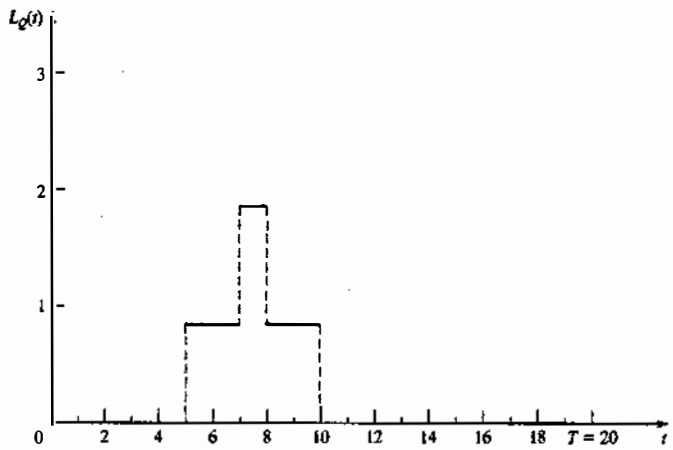


Figure 6.7 Number waiting in line, $L_Q(t)$, at time t.

time spent in system per customer, called the average system time, is given by the ordinary sample average

$$\hat{w} = \frac{1}{N}\sum_{i=1}^{N} W_i \tag{6.5}$$

For stable systems, as $N \to \infty$,

$$\hat{w} \to w \tag{6.6}$$

with probability 1, where w is called the long-run average system time.

If the system under consideration is the queue alone, Equations (6.5) and (6.6) are written as

$$\hat{w}_Q = \frac{1}{N}\sum_{i=1}^{N} W_i^Q \to w_Q \quad \text{as } N \to \infty \tag{6.7}$$

where W_i^Q is the total time customer i spends waiting in queue, $\hat{w}_Q$ is the observed average time spent in queue (called delay), and w_Q is the long-run average delay per customer. The estimators $\hat{w}$ and $\hat{w}_Q$ are influenced by initial conditions at time 0 and the run length T, analogously to $\hat{L}$.

Example 6.4

For the system history shown in Figure 6.6, $N = 5$ customers arrive, $W_1 = 2$, and $W_5 = 20 - 16 = 4$, but W_2, W_3, and W_4 cannot be computed unless more is known about the system. Assume that the system has a single server and a FIFO queue discipline. This implies that customers will depart from the system in the same order in which they arrived. Each jump upward of $L(t)$ in Figure 6.6 represents an arrival. Arrivals occur at times 0, 3, 5, 7, and 16. Similarly, departures occur at times 2, 8, 10, and 14. (A departure may or may not have occurred at time 20.) Under these assumptions, it is apparent that $W_2 = 8 - 3 = 5$, $W_3 = 10 - 5 = 5$, $W_4 = 14 - 7 = 7$, and therefore

$$\hat{w} = \frac{2 + 5 + 5 + 7 + 4}{5} = \frac{23}{5} = 4.6 \text{ time units}$$

Thus, on the average, an arbitrary customer spends 4.6 time units in the system. As for time spent in the waiting line, it can be computed that $W_1^Q = 0$, $W_2^Q = 0$, $W_3^Q = 8-5 = 3$, $W_4^Q = 10-7 = 3$, and $W_5^Q = 0$; thus,

$$\hat{w}_Q = \frac{0+0+3+3+0}{5} = 1.2 \text{ time units}$$

6.3.3 The Conservation Equation: $L = \lambda w$

For the system exhibited in Figure 6.6, there were $N = 5$ arrivals in $T = 20$ time units, and thus the observed arrival rate was $\hat{\lambda} = N/T = 1/4$ customer per time unit. Recall that $\hat{L} = 1.15$ and $\hat{w} = 4.6$; hence, it follows that

$$\hat{L} = \hat{\lambda}\hat{w} \tag{6.8}$$

This relationship between L, λ, and w is not coincidental; it holds for almost all queueing systems or subsystems regardless of the number of servers, the queue discipline, or any other special circumstances. Allowing $T \to \infty$ and $N \to \infty$, Equation (6.8) becomes

$$L = \lambda w \tag{6.9}$$

where $\hat{\lambda} \to \lambda$, and λ is the long-run average arrival rate. Equation (6.9) is called a conservation equation and is usually attributed to Little [1961]. It says that the average number of customers in the system at an arbitrary point in time is equal to the average number of arrivals per time unit, times the average time spent in the system. For Figure 6.6, there is one arrival every 4 time units (on the average) and each arrival spends 4.6 time units in the system (on the average), so at an arbitrary point in time there will be (1/4) (4.6) = 1.15 customers present (on the average).

Equation (6.8) can also be derived by reconsidering Figure 6.6 in the following manner: Figure 6.8 shows system history, $L(t)$, exactly as in Figure 6.6, with each customer's time in the system, W_i, represented by a rectangle. This representation again assumes a single-server system with a FIFO queue discipline. The

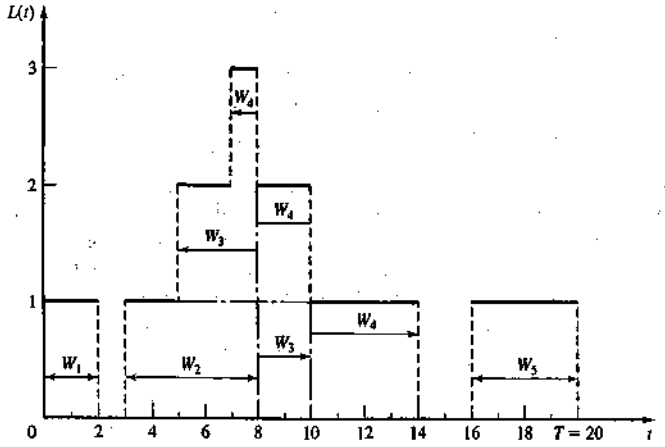


Figure 6.8 System times, W_i, for single-server FIFO system.

rectangles for the third and fourth customers are in two and three separate pieces, respectively. The ith rectangle has height 1 and length W_i for each $i = 1, 2, \ldots, N$. It follows that the total system time of all customers is given by the total area under the number-in-system function, $L(t)$; that is,

$$\sum_{i=1}^{N} W_i = \int_0^T L(t)\,dt \tag{6.10}$$

Therefore, by combining Equations (6.2) and (6.5) with $\hat{\lambda} = N/T$, it follows that

$$\hat{L} = \frac{1}{T}\int_0^T L(t)\,dt = \frac{N}{T}\frac{1}{N}\sum_{i=1}^{N} W_i = \hat{\lambda}\hat{w}$$

which is Little's equation (6.8). The intuitive and informal derivation presented here depended on the single-server FIFO assumptions, but these assumptions are not necessary. In fact, Equation (6.10), which was the key to the derivation, holds (at least approximately) in great generality, and thus so do Equations (6.8) and (6.9). Exercises 14 and 15 ask the reader to derive Equations (6.10) and (6.8) under different assumptions.

Technical note: If, as defined in Section 6.3.2, W_i is the system time for customer i during [0, T], then Equation (6.10) and hence Equation (6.8) hold exactly. Some authors choose to define W_i as total system time for customer i; this change will affect the value of W_i only for those customers i who arrive before time T but do not depart until after time T (possibly customer 5 in Figure 6.8). With this change in definition, Equations (6.10) and (6.8) hold only approximately. Nevertheless, as $T \to \infty$ and $N \to \infty$, the error in Equation (6.8) decreases to zero, and, therefore, the conservation equation (6.9) for long-run measures of performance—namely, $L = \lambda w$—holds exactly.

6.3.4 Server Utilization

Server utilization is defined as the proportion of time that a server is busy. Observed server utilization, denoted by $\hat{\rho}$, is defined over a specified time interval [0, T]. Long-run server utilization is denoted by ρ. For systems that exhibit long-run stability,

$$\hat{\rho} \to \rho \text{ as } T \to \infty$$

Example 6.5

Per Figure 6.6 or 6.8, and assuming that the system has a single server, it can be seen that the server utilization is $\hat{\rho}$ = (total busy time)/$T = (\sum_{i=1}^{\infty} T_i)/T = (T - T_0)/T = 17/20$.

Server utilization in G/G/1/∞/∞ queues

Consider any single-server queueing system with average arrival rate λ customers per time unit, average service time $E(S) = 1/\mu$ time units, and infinite queue capacity and calling population Notice that $E(S) = 1/\mu$ implies that, when busy, the server is working at the rate μ customers per time unit, on the average; μ is called the service rate. The server alone is a subsystem that can be considered as a queueing system in itself; hence, the conservation Equation (6.9), $L = \lambda w$, can be applied to the server. For stable systems, the average arrival rate to the server, say λ_s, must be identical to the average arrival rate to the system, λ (certainly $\lambda_s \le \lambda$—customers cannot be served faster than they arrive—but, if $\lambda_s < \lambda$, then the waiting line would tend to grow in length at an average rate of

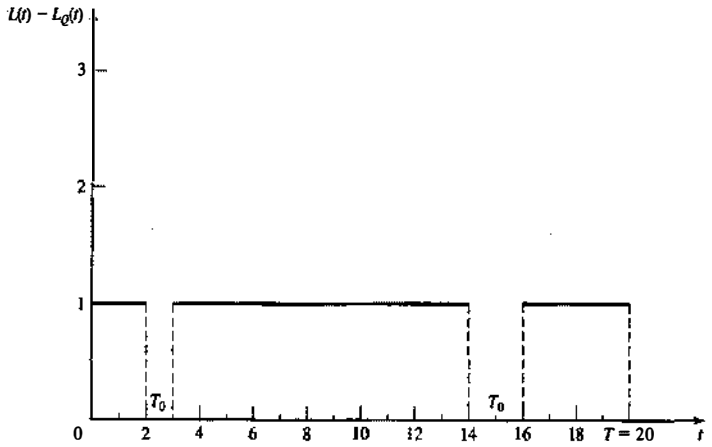


Figure 6.9 Number being served, $L(t) - L_Q(t)$, at time t.

$\lambda - \lambda_s$ customers per time unit, and so we would have an unstable system). For the server subsystem, the average system time is $w = E(S) = \mu^{-1}$. The actual number of customers in the server subsystem is either 0 or 1, as shown in Figure 6.9 for the system represented by Figure 6.6. Hence, the average number, $\hat{L}_s$, is given by

$$\hat{L}_s = \frac{1}{T}\int_0^T (L(t) - L_Q(t))\,dt = \frac{T - T_0}{T}$$

In this case, $\hat{L}_s = 17/20 = \hat{\rho}$. In general, for a single-server queue, the average number of customers being served at an arbitrary point in time is equal to server utilization. As $T \to \infty$, $\hat{L}_s = \hat{\rho} \to L_s = \rho$. Combining these results into $L = \lambda w$ for the server subsystem yields

$$\rho = \lambda E(S) = \frac{\lambda}{\mu} \tag{6.11}$$

—that is, the long-run server utilization in a single-server queue is equal to the average arrival rate divided by the average service rate. For a single-server queue to be stable, the arrival rate λ must be less than the service rate μ:

$$\lambda < \mu$$

or

$$\rho = \frac{\lambda}{\mu} < 1 \tag{6.12}$$

If the arrival rate is greater than the service rate ($\lambda > \mu$), the server will eventually get further and further behind. After a time, the server will always be busy, and the waiting line will tend to grow in length at an average rate of $(\lambda - \mu)$ customers per time unit, because departures will be occurring at rate μ per time unit. For stable single-server systems ($\lambda < \mu$ or $\rho < 1$), long-run measures of performance such as average queue

length L_Q (and also L, w, and w_Q) are well defined and have meaning. For unstable systems ($\lambda > \mu$), long-run server utilization is 1, and long-run average queue length is infinite; that is,

$$\frac{1}{T}\int_0^T L_Q(t)\,dt \to +\infty \text{ as } T \to \infty$$

Similarly, $L = w = w_Q = \infty$. Therefore these long-run measures of performance are meaningless for unstable queues. The quantity λ/μ is also called the offered load and is a measure of the workload imposed on the system.

Server utilization in G/G/c/∞/∞ queues

Consider a queueing system with c identical servers in parallel. If an arriving customer finds more than one server idle, the customer chooses a server without favoring any particular server. (For example, the choice of server might be made at random.) Arrivals occur at rate λ from an infinite calling population, and each server works at rate μ customers per time unit. From Equation (6.9), $L = \lambda w$, applied to the server subsystem alone, an argument similar to the one given for a single server leads to the result that, for systems in statistical equilibrium, the average number of busy servers, say L_s, is given by

$$L_s = \lambda E(S) = \frac{\lambda}{\mu} \tag{6.13}$$

Clearly, $0 \le L_s \le c$. The long-run average server utilization is defined by

$$\rho = \frac{L_s}{c} = \frac{\lambda}{c\mu} \tag{6.14}$$

and so $0 \le \rho \le 1$. The utilization ρ can be interpreted as the proportion of time an arbitrary server is busy in the long run.

The maximum service rate of the $G/G/c/\infty/\infty$ system is $c\mu$, which occurs when all servers are busy. For the system to be stable, the average arrival rate λ must be less than the maximum service rate $c\mu$; that is, the system is stable if and only if

$$\lambda < c\mu \tag{6.15}$$

or, equivalently, if the offered load λ/μ is less than the number of servers c. If $\lambda > c\mu$, then arrivals are occurring, on the average, faster than the system can handle them, all servers will be continuously busy, and the waiting line will grow in length at an average rate of $(\lambda - c\mu)$ customers per time unit. Such a system is unstable, and the long-run performance measures (L, L_Q, w, and w_Q) are again meaningless for such systems.

Notice that Condition (6.15) generalizes Condition (6.12), and the equation for utilization for stable systems, Equation (6.14), generalizes Equation (6.11).

Equations (6.13) and (6.14) can also be applied when some servers work more than others, for example, when customers favor one server over others, or when certain servers serve customers only if all other servers are busy. In this case, the L_s given by Equation (6.13) is still the average number of busy servers, but ρ, as given by Equation (6.14), cannot be applied to an individual server. Instead, ρ must be interpreted as the average utilization of all servers.

Example 6.6

Customers arrive at random to a license bureau at a rate of $\lambda = 50$ customers per hour. Currently, there are 20-clerks, each serving $\mu = 5$ customers per hour on the average. Therefore the long-run, or steady-state, average utilization of a server, given by Equation (6.14), is

$$\rho = \frac{\lambda}{c\mu} = \frac{50}{20(5)} = 0.5$$

and the average number of busy servers is

$$L_s = \frac{\lambda}{\mu} = \frac{50}{5} = 10$$

Thus, in the long run, a typical clerk is busy serving customers only 50% of the time. The office manager asks whether the number of servers can be decreased. By Equation (6.15), it follows that, for the system to be stable, it is necessary for the number of servers to satisfy

$$c > \frac{\lambda}{\mu}$$

or $c > 50/5 = 10$. Thus, possibilities for the manager to consider include $c = 11$, or $c = 12$, or $c = 13$, ... Notice that $c \geq 11$ guarantees long-run stability only in the sense that all servers, when busy, can handle the incoming work load (i.e., $c\mu > \lambda$) on average. The office manager could well desire to have more than the minimum number of servers ($c = 11$) because of other factors, such as customer delays and length of the waiting line. A stable queue can still have very long lines on average.

Server utilization and system performance

As will be illustrated here and in later sections, system performance can vary widely for a given value of utilization, ρ. Consider a $G/G/1/\infty/\infty$ queue: that is, a single-server queue with arrival rate λ, service rate μ, and utilization $\rho = \lambda/\mu < 1$.

At one extreme, consider the $D/D/1$ queue, which has deterministic arrival and service times. Then all interarrival times $\{A_1, A_2, \ldots\}$ are equal to $E(A) = 1/\lambda$, and all service times $\{S_1, S_2, \ldots\}$ are equal to $E(S) = 1/\mu$. Assuming that a customer arrives to an empty system at time 0, the system evolves in a completely deterministic and predictable fashion, as shown in Figure 6.10. Observe that $L = \rho = \lambda/\mu$, $w = E(S) = \mu^{-1}$, and $L_Q = w_Q = 0$. By varying λ and μ, server utilization can assume any value between 0 and 1, yet there is never any line whatsoever. What, then, causes lines to build, if not a high server utilization? In general, it is the variability of interarrival and service times that causes lines to fluctuate in length.

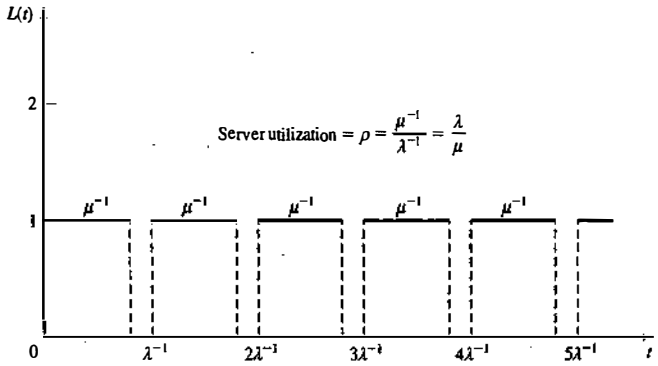


Figure 6.10 Deterministic queue ($D/D/1$).

Example 6.7

Consider a physician who schedules patients every 10 minutes and who spends S_i minutes with the ith patient, where

$$S_i = \begin{cases} 9 \text{ minutes with probability } 0.9 \\ 12 \text{ minutes with probability } 0.1 \end{cases}$$

Thus, arrivals are deterministic ($A_1 = A_2 = \cdots = \lambda^{-1} = 10$) but services are stochastic (or probabilistic), with mean and variance given by

$$E(S_i) = 9(0.9) + 12(0.1) = 9.3 \text{ minutes}$$

and

$$\begin{aligned} V(S_i) &= E(S_i^2) - [E(S_i)]^2 \\ &= 9^2(0.9) + 12^2(0.1) - (9.3)^2 \\ &= 0.81 \text{ minutes}^2 \end{aligned}$$

Here, $\rho = \lambda/\mu = E(S)/E(A) = 9.3/10 = 0.93 < 1$, the system is stable, and the physician will be busy 93% of the time in the long run. In the short run, lines will not build up as long as patients require only 9 minutes of service, but, because of the variability in the service times, 10% of the patients will require 12 minutes, which in turn will cause a temporary line to form.

Suppose the system is simulated with service times, $S_1 = 9$, $S_2 = 12$, $S_3 = 9$, $S_4 = 9$, $S_5 = 9$, ... Assuming that at time 0 a patient arrived to find the doctor idle and subsequent patients arrived precisely at times 10, 20, 30, ..., the system evolves as in Figure 6.11. The delays in queue are $W_1^Q = W_2^Q = 0$, $W_3^Q = 22 - 20 = 2$, $W_4^Q = 31 - 30 = 1$, $W_5^Q = 0$. The occurrence of a relatively long service time (here $S_2 = 12$) caused a waiting line to form temporarily. In general, because of the variability of the interarrival and service distributions, relatively small interarrival times and relatively large service times occasionally do occur, and these in turn cause lines to lengthen. Conversely, the occurrence of a large interarrival time or a small service time will tend to shorten an existing waiting line. The relationship between utilization, service and interarrival variability, and system performance will be explored in more detail in Section 6.4.

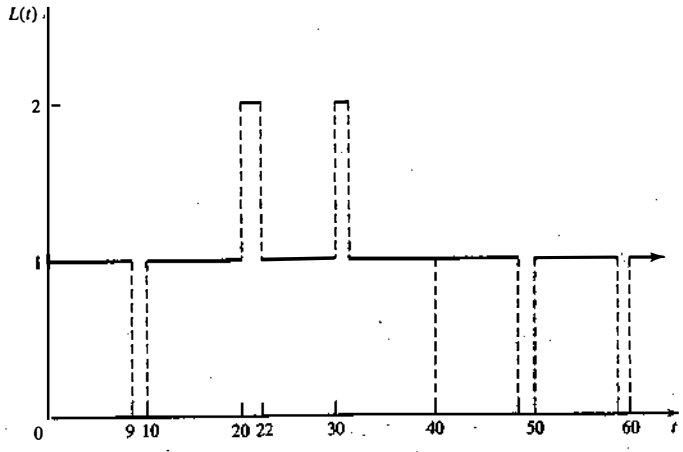


Figure 6.11 Number of patients in the doctor's office at time t.

6.3.5 Costs in Queueing Problems

In many queueing situations, costs can be associated with various aspects of the waiting line or servers. Suppose that the system incurs a cost for each customer in the queue, say at a rate of \$10 per hour per customer. If customer j spends W_j^Q hours in the queue, then $\sum_{j=1}^{N}(\$10 \cdot W_j^Q)$ is the total cost of the N customers who arrive during the simulation. Thus, the average cost per customer is

$$\sum_{j=1}^{N} \frac{\$10 \cdot W_j^Q}{N} = \$10 \cdot \hat{w}_Q$$

by Equation (6.7). If $\hat{\lambda}$ customers per hour arrive (on the average), the average cost per hour is

$$\left(\hat{\lambda}\frac{\text{customers}}{\text{hour}}\right)\left(\frac{\$10 \cdot \hat{w}_Q}{\text{customer}}\right) = \$10 \cdot \hat{\lambda}\hat{w}_Q = \$10 \cdot \hat{L}_Q/\text{hour}$$

the last equality following by Little's equation (6.8). An alternative way to derive the average cost per hour is to consider Equation (6.2). If T_i^Q is the total time over the interval $[0, T]$ that the system contains exactly i customers, then $\$10\, iT_i^Q$ is the cost incurred by the system during the time exactly i customers are present. Thus, the total cost is $\sum_{i=1}^{\infty}(\$10 \cdot iT_i^Q)$, and the average cost per hour is

$$\sum_{i=1}^{\infty} \frac{\$10 \cdot iT_i^Q}{T} = \$10 \cdot L_Q/\text{hour}$$

by Equation (6.2). In these cost expressions, $\hat{L}_Q$ may be replaced by L_Q (if the long-run number in queue is known), or by L or $\hat{L}$ (if costs are incurred while the customer is being served in addition to being delayed).

The server may also impose costs on the system. If a group of c parallel servers ($1 \le c < \infty$) have utilization ρ, and each server imposes a cost of \$5 per hour while busy, the total server cost per hour is

$$\$5 \cdot c\rho$$

because $c\rho$ is the average number of busy servers. If server cost is imposed only when the servers are idle, then the server cost per hour would be

$$\$5 \cdot c(1-\rho)$$

because $c(1 - \rho) = c - c\rho$ is the average number of idle servers. In many problems, two or more of these various costs are combined into a total cost. Such problems are illustrated by Exercises 5, 12, 17, and 20. In most cases, the objective is to minimize total costs (given certain constraints) by varying those parameters that are under management's control, such as the number of servers, the arrival rate, the service rate, and the system capacity.

6.4 STEADY-STATE BEHAVIOR OF INFINITE-POPULATION MARKOVIAN MODELS

This section presents the steady-state solution of a number of queueing models that can be solved mathematically. For the infinite-population models, the arrivals are assumed to follow a Poisson process with rate λ arrivals per time unit—that is, the interarrival times are assumed to be exponentially distributed with mean $1/\lambda$. Service

times may be exponentially distributed (M) or arbitrarily (G). The queue discipline will be FIFO. Because of the exponential-distributional assumptions on the arrival process, these models are called Markovian models.

A queueing system is said to be in statistical equilibrium, or steady state, if the probability that the system is in a given state is not time dependent—that is,

$$P(L(t) = n) = P_n(t) = P_n$$

is independent of time t. Two properties—approaching statistical equilibrium from any starting state, and remaining in statistical equilibrium once it is reached—are characteristic of many stochastic models, and, in particular, of all the systems studied in the following subsections. On the other hand, if an analyst were interested in the transient behavior of a queue over a relatively short period of time and were given some specific initial conditions (such as idle and empty), the results to be presented here would be inappropriate. A transient mathematical analysis or, more likely, a simulation model would be the chosen tool of analysis.

The mathematical models whose solutions are shown in the following subsections can be used to obtain approximate results even when the assumptions of the model do not strictly hold. These results may be considered as a rough guide to the behavior of the system. A simulation may then be used for a more refined analysis. However, it should be remembered that a mathematical analysis (when it is applicable) provides the true value of the model parameter (e.g., L), whereas a simulation analysis delivers a statistical estimate (e.g., $\hat{L}$) of the parameter. On the other hand, for complex systems, a simulation model is often a more faithful representation than a mathematical model.

For the simple models studied here, the steady-state parameter L, the time-average number of customers in the system, can be computed as

$$L = \sum_{n=0}^{\infty} nP_n \tag{6.16}$$

where $\{P_n\}$ are the steady-state probabilities of finding n customers in the system (as defined in Table 6.2). As was discussed in Section 6.3 and was expressed in Equation 6.3), L can also be interpreted as a long-run measure of performance of the system. Once L is given, the other steady-state parameters can be computed readily from Little's equation (6.9) applied to the whole system and to the queue alone:

$$\begin{aligned} w &= \frac{L}{\lambda} \\ w_Q &= w - \frac{1}{\mu} \\ L_Q &= \lambda w_Q \end{aligned} \tag{6.17}$$

where λ is the arrival rate and μ is the service rate per server.

For the $G/G/c/\infty/\infty$ queues considered in this section to have a statistical equilibrium, a necessary and sufficient condition is that $\lambda/(c\mu) < 1$, where λ is the arrival rate, μ is the service rate of one server, and c is the number of parallel servers. For these unlimited capacity, infinite-calling-population models, it shall be assumed that the theoretical server utilization, $\rho = \lambda/(c\mu)$, satisfies $\rho < 1$. For models with finite system capacity or finite calling population, the quantity $\lambda/(c\mu)$ may assume any positive value.

6.4.1 Single-Server Queues with Poisson Arrivals and Unlimited Capacity: M/G/1

Suppose that service times have mean $1/\mu$ and variance σ^2 and that there is one server. If $\rho = \lambda/\mu < 1$, then the $M/G/1$ queue has a steady-state probability distribution with steady-state characteristics, as given in Table 6.3. In general, there is no simple expression for the steady-state probabilities $P_0, P_1, P_2, \ldots$. When

Table 6.3 Steady-State Parameters of the $M/G/1$ Queue

ρ	$\frac{\lambda}{\mu}$
L	$\rho+\frac{\lambda^2(1/\mu^2+\sigma^2)}{2(1-\rho)}=\rho+\frac{\rho^2(1+\sigma^2\mu^2)}{2(1-\rho)}$
w	$\frac{1}{\mu}+\frac{\lambda(1/\mu^2+\sigma^2)}{2(1-\rho)}$
w_Q	$\frac{\lambda(1/\mu^2+\sigma^2)}{2(1-\rho)}$
L_Q	$\frac{\lambda^2(1/\mu^2+\sigma^2)}{2(1-\rho)}=\frac{\rho^2(1+\sigma^2\mu^2)}{2(1-\rho)}$
P_0	$1-\rho$

$\lambda < \mu$, the quantity $\rho = \lambda/\mu$ is the server utilization, or long-run proportion of time the server is busy. As is seen in Table 6.3, $1 - P_0 = \rho$ can also be interpreted as the steady-state probability that the system contains one or more customers. Notice also that $L - L_Q = \rho$ is the time-average number of customers being served.

Example 6.8

Widget-making machines malfunction apparently at random and then require a mechanic's attention. It is assumed that malfunctions occur according to a Poisson process, at the rate $\lambda = 1.5$ per hour. Observation over several months has found that repair times by the single mechanic take an average time of 30 minutes, with a standard deviation of 20 minutes. Thus the mean service time $1/\mu = 1/2$ hour, the service rate is $\mu = 2$ per hour and $\sigma^2 = (20)^2$ minutes^2 = 1/9 hour^2. The "customers" are the widget makers, and the appropriate model is the $M/G/1$ queue, because only the mean and variance of service times are known, not their distribution. The proportion of time the mechanic is busy is $\rho = \lambda/\mu = 1.5/2 = 0.75$, and, by Table 6.3, the steady-state time average number of broken machines is

$$L = 0.75 + \frac{(1.5)^2[(0.5)^2 + 1/9]}{2(1-0.75)}$$
$$= 0.75 + 1.625 = 2.375 \text{ machines}$$

Thus, an observer who notes the state of the repair system at arbitrary times would find an average of 2.375 broken machines (over the long run).

A closer look at the formulas in Table 6.3 reveals the source of the waiting lines and delays in an $M/G/1$ queue. For example, L_Q may be rewritten as

$$L_Q = \frac{\rho^2}{2(1-\rho)} + \frac{\lambda^2\sigma^2}{2(1-\rho)}$$

The first term involves only the ratio of the mean arrival rate, λ, to the mean service rate, μ. As shown by the second term, if λ and μ are held constant, the average length of the waiting line (L_Q) depends on the variability, σ^2, of the service times. If two systems have identical mean service times and mean interarrival times,

the one with the more variable service times (larger σ^2) will tend to have longer lines on the average. Intuitively, if service times are highly variable, there is a high probability that a large service time will occur (say, much larger than the mean service time), and, when large service times do occur, there is a higher-than-usual tendency for lines to form and delays of customers to increase. (The reader should not confuse "steady state" with low variability or short lines; a system in steady-state or statistical equilibrium can be highly variable and can have long waiting lines.)

Example 6.9

There are two workers competing for a job. Able claims an average service time that is faster than Baker's, but Baker claims to be more consistent, even if not as fast. The arrivals occur according to a Poisson process at the rate $\lambda = 2$ per hour (1/30 per minute). Able's service statistics are an average service time of 24 minutes with a standard deviation of 20 minutes. Baker's service statistics are an average service time of 25 minutes, but a standard deviation of only 2 minutes. If the average length of the queue is the criterion for hiring, which worker should be hired? For Able, $\lambda = 1/30$ per minute, $1/\mu = 24$ minutes, $\sigma^2 = 20^2 = 400$ minutes^2, $\rho = \lambda/\mu = 24/30 = 4/5$, and the average queue length is computed as

$$L_Q = \frac{(1/30)^2[24^2 + 400]}{2(1-4/5)} = 2.711 \text{ customers}$$

For Baker, $\lambda = 1/30$ per minute, $1/\mu = 25$ minutes, $\sigma^2 = 2^2 = 4$ minutes^2, $\rho = 25/30 = 5/6$, and the average queue length is

$$L_Q = \frac{(1/30)^2[25^2 + 4]}{2(1-5/6)} = 2.097 \text{ customers}$$

Although working faster on the average, Able's greater service variability results in an average queue length about 30% greater than Baker's. On the basis of average queue length, L_Q, Baker wins. On the other hand, the proportion of arrivals who would find Able idle and thus experience no delay is $P_0 = 1 - \rho = 1/5 = 20\%$, but the proportion who would find Baker idle and thus experience no delay is $P_0 = 1 - \rho = 1/6 = 16.7\%$.

One case of the $M/G/1$ queue that is of special note occurs when service times are exponential, which we describe next.

The M/M/1 queue. Suppose that service times in an $M/G/1$ queue are exponentially distributed, with mean $1/\mu$; then the variance as given by Equation (5.27) is $\sigma^2 = 1/\mu^2$. The mean and standard deviation of the exponential distribution are equal, so the $M/M/1$ queue will often be a useful approximate model when service times have standard deviations approximately equal to their means. The steady-state parameters, given in Table 6.4, may be computed by substituting $\sigma^2 = 1/\mu^2$ into the formulas in Table 6.3. Alternatively, L may be computed by Equation (6.16) from the steady-state probabilities P_n given in Table 6.4, and then w, w_Q, and L_Q may be computed from Equations (6.17). The student can show that the two expressions for each parameter are equivalent by substituting $\rho = \lambda/\mu$ into the right-hand side of each equation in Table 6.4.

Example 6.10

Suppose that the interarrival times and service times at a single-chair unisex hair-styling shop have been shown to be exponentially distributed. The values of λ and μ are 2 per hour and 3 per hour, respectively—that is, the time between arrivals averages 1/2 hour, exponentially distributed, and the service time averages 20 minutes, also exponentially distributed. The server utilization and the probabilities for zero, one, two, three, and four or more customers in the shop are computed as follows:

$$\rho = \frac{\lambda}{\mu} = \frac{2}{3}$$

$$P_0 = 1 - \frac{\lambda}{\mu} = \frac{1}{3}$$

$$P_1 = \left(\frac{1}{3}\right)\left(\frac{2}{3}\right) = \frac{2}{9}$$

$$P_2 = \left(\frac{1}{3}\right)\left(\frac{2}{3}\right)^2 = \frac{4}{27}$$

$$P_3 = \left(\frac{1}{3}\right)\left(\frac{2}{3}\right)^3 = \frac{8}{81}$$

$$P_{\geq 4} = 1 - \sum_{n=0}^{3} P_n = 1 - \frac{1}{3} - \frac{2}{9} - \frac{4}{27} - \frac{8}{81} = \frac{16}{81}$$

From the calculations, the probability that the hair stylist is busy is $1 - P_0 = \rho = 0.67$; thus, the probability that the hair stylist is idle is 0.33. The time-average number of customers in the system is given by Table 6.4 as

$$L = \frac{\lambda}{\mu - \lambda} = \frac{2}{3-2} = 2 \text{ customers}$$

The average time an arrival spends in the system can be obtained from Table 6.4 or Equation (6.17) as

$$w = \frac{L}{\lambda} = \frac{2}{2} = 1 \text{ hour}$$

The average time the customer spends in the queue can be obtained from Equation (6.17) as

$$w_Q = w - \frac{1}{\mu} = 1 - \frac{1}{3} = \frac{2}{3} \text{ hour}$$

Table 6.4 Steady-State Parameters of the M/M/1 Queue

L	$\frac{\lambda}{\mu - \lambda} = \frac{\rho}{1-\rho}$
w	$\frac{1}{\mu - \lambda} = \frac{1}{\mu(1-\rho)}$
w_Q	$\frac{\lambda}{\mu(\mu-\lambda)} = \frac{\rho}{\mu(1-\rho)}$
L_Q	$\frac{\lambda^2}{\mu(\mu-\lambda)} = \frac{\rho^2}{1-\rho}$
P_n	$\left(1-\frac{\lambda}{\mu}\right)\left(\frac{\lambda}{\mu}\right)^n = (1-\rho)\rho^n$

From Table 6.4, the time-average number in the queue is given by

$$L_Q = \frac{\lambda^2}{\mu(\mu-\lambda)} = \frac{4}{3(1)} = \frac{4}{3} \text{ customers}$$

Finally, notice that multiplying $w = w_Q + 1/\mu$ through by λ and using Little's equation (6.9) yields

$$L = L_Q + \frac{\lambda}{\mu} = \frac{4}{3} + \frac{2}{3} = 2 \text{ customers}$$

Example 6.11

For the $M/M/1$ queue with service rate $\mu = 10$ customers per hour, consider how L and w increase as the arrival rate, λ, increases from 5 to 8.64 by increments of 20%, and then to $\lambda = 10$.

λ	5.0	6.0	7.2	8.64	10.0
ρ	0.500	0.600	0.720	0.864	1.0
L	1.00	1.50	2.57	6.35	∞
w	0.20	0.25	0.36	0.73	∞

For any $M/G/1$ queue, if $\lambda/\mu \geq 1$, waiting lines tend to continually grow in length; the long-run measures of performance, L, w, w_Q, and L_Q are all infinite ($L = w = w_Q = L_Q = \infty$); and a steady-state probability distribution does not exist. As is shown here for $\lambda < \mu$, if ρ is close to 1, waiting lines and delays will tend to be long. Notice that the increase in average system time, w, and average number in system, L, is highly nonlinear as a function of ρ. For example, as λ increases by 20%, L increases first by 50% (from 1.00 to 1.50), then by 71% (to 2.57), and then by 147% (to 6.35).

Example 6.12

If arrivals are occurring at rate $\lambda = 10$ per hour, and mauagement has a choice of two servers, one who works at rate $\mu_1 = 11$ customers per hour and the second at rate $\mu_2 = 12$ customers per hour, the respective utilizations are $\rho_1 = \lambda/\mu_1 = 10/11 = 0.909$ and $\rho_2 = \lambda/\mu_2 = 10/12 = 0.833$. If the $M/M/1$ queue is used as an approximate model, then, with the first server, the average number in the system would be, by Table 6.4,

$$L_1 = \frac{\rho_1}{1-\rho_1} = 10$$

and, with the second server, the average number in the system would be

$$L_2 = \frac{\rho_2}{1-\rho_2} = 5$$

Thus, a decrease in service rate from 12 to 11 customers per hour, a mere 8.3% decrease, would result in an increase in average number in system from 5 to 10, which is a 100% increase.

The effect of utilization and service variability

For any $M/G/1$ queue, if lines are too long, they can be reduced by decreasing the server utilization ρ or by decreasing the service time variability, σ^2. These remarks hold for almost all queues, not just the $M/G/1$ queue. The utilization factor ρ can be reduced by decreasing the arrival rate λ, by increasing the service rate μ, or by increasing the number of servers, because, in general, $\rho = \lambda/(c\mu)$, where c is the number of parallel servers. The effect of additional servers will be studied in the following subsections. Figure 6.12 illustrates the effect of service variability. The mean steady-state number in the queue, L_Q, is plotted versus utilization

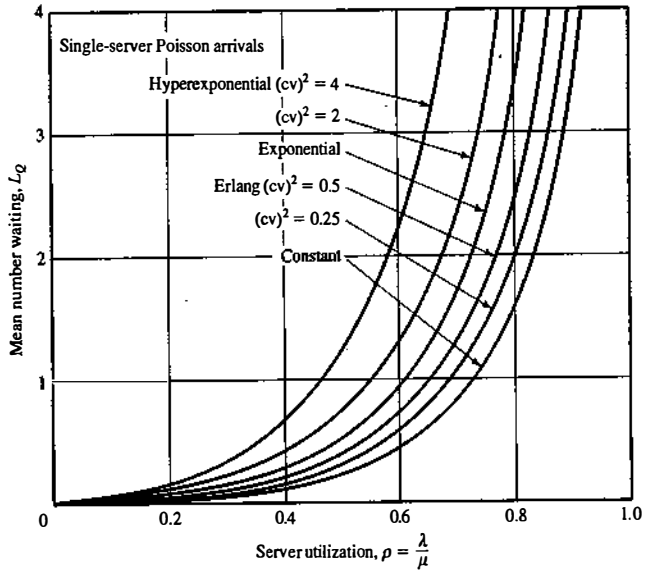


Figure 6.12 Mean number of customers waiting, L_Q, in $M/G/1$ queue having service distributions with given cv. (Adapted from Geoffrey Gordon, *System Simulation*, 2nd ed., Prentice-Hall, Englewood Cliffs, NJ, 1978.)

ρ for a number of different coefficients of variation. The coefficient of variation (cv) of a positive random variable X is defined as

$$(\text{cv})^2 = \frac{V(X)}{[E(X)]^2}$$

It is a measure of the variability of a distribution. The larger its value, the more variable is the distribution relative to its expected value. For deterministic service times, $V(X) = 0$, so cv = 0. For Erlang service times of order k, $V(X) = 1/(k\mu^2)$ and $E(X) = 1/\mu$, so $\text{cv} = 1/\sqrt{k}$. For exponential service times at service rate μ, the mean service time is $E(X) = 1/\mu$ and the variance is $V(X) = 1/\mu^2$, so cv = 1. If service times have standard deviation greater than their mean (i.e., if cv > 1), then the hyperexponential distribution, which can achieve any desired coefficient of variation greater than 1, provides a good model. One occasion where it arises is given in Exercise 16.

The formula for L_Q for any $M/G/1$ queue can be rewritten in terms of the coefficient of variation by noticing that $(\text{cv})^2 = \sigma^2/(1/\mu)^2 = \sigma^2\mu^2$. Therefore,

$$\begin{aligned} L_Q &= \frac{\rho^2(1+\sigma^2\mu^2)}{2(1-\rho)} \\ &= \frac{\rho^2(1+(\text{cv})^2)}{2(1-\rho)} \\ &= \left(\frac{\rho^2}{1-\rho}\right)\left(\frac{1+(\text{cv})^2}{2}\right) \end{aligned} \tag{6.18}$$

The first term, $\rho^2/(1-\rho)$, is L_Q for an $M/M/1$ queue. The second term, $(1 + (\text{cv})^2)/2$, corrects the $M/M/1$ formula to account for a nonexponential service-time distribution. The formula for w_Q can be obtained from the corresponding $M/M/1$ formula by applying the same correction factor.

6.4.2 Multiserver Queue: M/M/c/∞/∞

Suppose that there are c channels operating in parallel. Each of these channels has an independent and identical exponential service-time distribution, with mean $1/\mu$. The arrival process is Poisson with rate λ. Arrivals will join a single queue and enter the first available service channel. The queueing system is shown in Figure 6.13. If the number in system is $n < c$, an arrival will enter an available channel. However, when $n \geq c$, a queue will build if arrivals occur.

The offered load is defined by λ/μ. If $\lambda \geq c\mu$, the arrival rate is greater than or equal to the maximum service rate of the system (the service rate when all servers are busy); thus, the system cannot handle the load put upon it, and therefore it has no statistical equilibrium. If $\lambda > c\mu$, the waiting line grows in length at the rate $(\lambda - c\mu)$ customers per time unit, on the average. Customers are entering the system at rate λ per time unit but are leaving the system at a maximum rate of $c\mu$ per time unit.

For the $M/M/c$ queue to have statistical equilibrium, the offered load must satisfy $\lambda/\mu < c$, in which case $\lambda/(c\mu) = \rho$, the server utilization. The steady-state parameters are listed in Table 6.5. Most of the measures of performance can be expressed fairly simply in terms of P_0, the probability that the system is empty, or $\sum_{n=c}^{\infty} P_n$, the probability that all servers are busy, denoted by $P(L(\infty) \geq c)$, where $L(\infty)$ is a random variable representing the number in system in statistical equilibrium (after a very long time). Thus, $P(L(\infty) = n) = P_n$, $n = 0, 1, 2, \ldots$. The value of P_0 is necessary for computing all the measures of performance, and the equation for P_0 is somewhat more complex than in the previous cases. However, P_0 depends only on c and ρ. A good approximation to P_0 can be obtained by using Figure 6.14, where P_0 is plotted versus ρ on semilog paper for various values c. Figure 6.15 is a plot of L versus ρ for different values of c.

The results in Table 6.5 simplify to those in Table 6.4 when $c = 1$, the case of a single server. Notice that the average number of busy servers, or the average number of customers being served, is given by the simple expression $L - L_Q = \lambda/\mu = c\rho$.

Example 6.13

Many early examples of queueing theory applied to practical problems concerning tool cribs. Attendants manage the tool cribs as mechanics, assumed to be from an infinite calling population, arrive for service. Assume Poisson arrivals at rate 2 mechanics per minute and exponentially distributed service times with mean 40 seconds.

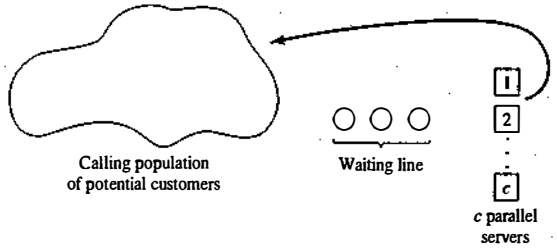


Figure 6.13 Multiserver queueing system.

Table 6.5 Steady-State parameters for the $M/M/c$ Queue

ρ	$\dfrac{\lambda}{c\mu}$
P_0	$\left\{\left[\sum_{n=0}^{c-1}\dfrac{(\lambda/\mu)^n}{n!}\right]+\left[\left(\dfrac{\lambda}{\mu}\right)^c\left(\dfrac{1}{c!}\right)\left(\dfrac{c\mu}{c\mu-\lambda}\right)\right]\right\}^{-1}$ $=\left\{\left[\sum_{n=0}^{c-1}\dfrac{(c\rho)^n}{n!}\right]+\left[(c\rho)^c\left(\dfrac{1}{c!}\right)\left(\dfrac{1}{1-\rho}\right)\right]\right\}^{-1}$
$P(L(\infty)\geq c)$	$\dfrac{(\lambda/\mu)^c P_0}{c!(1-\lambda/c\mu)}=\dfrac{(c\rho)^c P_0}{c!(1-\rho)}$
L	$c\rho+\dfrac{(c\rho)^{c+1}P_0}{c(c!)(1-\rho)^2}=c\rho+\dfrac{\rho P(L(\infty)\geq c)}{(1-\rho)}$
w	$\dfrac{L}{\lambda}$
w_Q	$w-\dfrac{1}{\mu}$
L_Q	$\lambda w_Q=\dfrac{(c\rho)^{c+1}P_0}{c(c!)(1-\rho)^2}=\dfrac{\rho P(L(\infty)\geq c)}{(1-\rho)}$
$L-L_Q$	$\dfrac{\lambda}{\mu}=c\rho$

Now, $\lambda = 2$ per minute, and $\mu = 60/40 = 3/2$ per minute. The offered load is greater than 1:

$$\frac{\lambda}{\mu}=\frac{2}{3/2}=\frac{4}{3}>1$$

so more than one server is needed if the system is to have a statistical equilibrium. The requirement for steady state is that $c > \lambda/\mu = 4/3$. Thus at least $c = 2$ attendants are needed. The quantity 4/3 is the expected number of busy servers, and for $c \geq 2$, $\rho = 4/(3c)$ is the long-run proportion of time each server is busy. (What would happen if there were only $c = 1$ server?)

Let there be $c = 2$ attendants. First, P_0 is calculated as

$$P_0=\left\{\sum_{n=0}^{1}\frac{(4/3)^n}{n!}+\left(\frac{4}{3}\right)^2\left(\frac{1}{2!}\right)\left[\frac{2(3/2)}{2(3/2)-2}\right]\right\}^{-1}$$

$$=\left\{1+\frac{4}{3}+\left(\frac{16}{9}\right)\left(\frac{1}{2}\right)(3)\right\}^{-1}=\left(\frac{15}{3}\right)^{-1}=\frac{1}{5}=0.2$$

Next, the probability that all servers are busy is computed as

$$P(L(\infty)\geq 2)=\frac{(4/3)^2}{2!(1-2/3)}\left(\frac{1}{5}\right)=\left(\frac{8}{3}\right)\left(\frac{1}{5}\right)=\frac{8}{15}=0.533$$

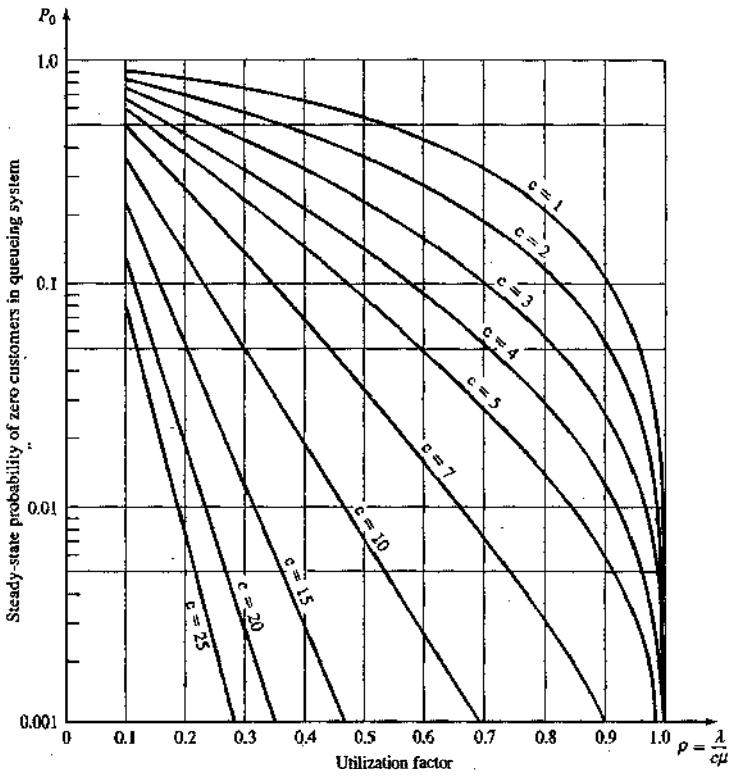


Figure 6.14 Values of P_0 for $M/M/c/\infty$ model. (From F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 5th ed., 1990, p. 616. Adapted with permission of McGraw-Hill, Inc., New York.)

Thus, the time-average length of the waiting line of mechanics is

$$L_Q=\frac{(2/3)(8/15)}{1-2/3}=1.07 \text{ mechanics}$$

and the time-average number in system is given by

$$L=L_Q+\frac{\lambda}{\mu}=\frac{16}{15}+\frac{4}{3}=\frac{12}{5}=2.4 \text{ mechanics}$$

From Little's relationships, the average time a mechanic spends at the tool crib is

$$w=\frac{L}{\lambda}=\frac{2.4}{2}=1.2 \text{ minutes}$$

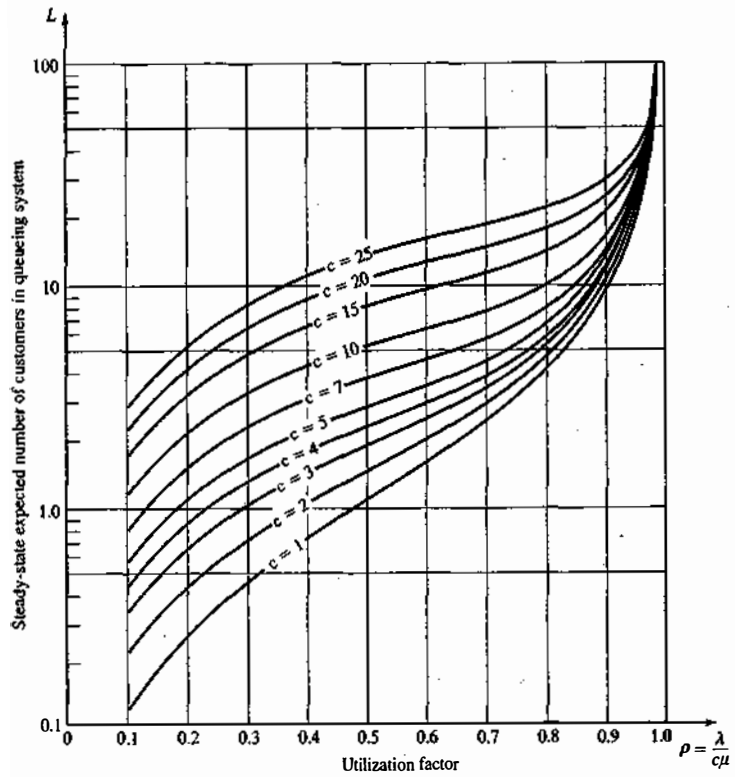


Figure 6.15 Values of L for $M/M/c/\infty$ model. (From F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 5th ed., 1990, p. 617. Adapted with permission of McGraw-Hill, Inc., New York.)

and the average time spent waiting for an attendant is

$$w_Q = w - \frac{1}{\mu} = 1.2 - \frac{2}{3} = 0.533 \text{ minute}$$

Example 6.14

Using the data of Example 6.13, compute P_0 and L from Figures 6.14 and 6.15. First, compute

$$\rho = \frac{\lambda}{c\mu} = \frac{2}{2(3/2)} = \frac{2}{3} = 0.667$$

Entering the utilization factor 0.667 on the horizontal axis of Figure 6.14 gives the value 0.2 for P_0 on the vertical axis. Similarly, the value $L = 2.4$ is read from the vertical axis of Figure 6.15.

An Approximation for the M/G/c/∞ Queue

Recall that formulas for L_Q and w_Q for the $M/G/1$ queue can be obtained from the corresponding $M/M/1$ formulas by multiplying them by the correction factor $(1 + (cv)^2)/2$, as in Equation (6.18). *Approximate* formulas for the $M/G/c$ queue can be obtained by applying the same correction factor to the $M/M/c$ formulas for L_Q and w_Q (no exact formula exists for $1 < c < \infty$). The nearer the cv is to 1, the better the approximation.

Example 6.15

Recall Example 6.13. Suppose that the service times for the mechanics at the tool crib are not exponentially distributed, but are known to have a standard deviation of 30 seconds. Then we have an $M/G/c$ model, rather than an $M/M/c$. The mean service time is 40 seconds, so the coefficient of variation of the service time is

$$\text{cv} = \frac{30}{40} = \frac{3}{4} < 1$$

Therefore, the accuracy of L_Q and w_Q can be improved by the correction factor

$$\frac{1+(\text{cv})^2}{2} = \frac{1+(3/4)^2}{2} = \frac{25}{32} = 0.78$$

For example, when there are $c = 2$ attendants,

$$L_Q = (0.78)(1.07) = 0.83 \text{ mechanics}$$

Notice that, because the coefficient of variation of the service time is less than 1, the congestion in the system, as measured by L_Q, is less than in the corresponding $M/M/2$ model.

The correction factor applies only to the formulas for L_Q and w_Q. Little's formula can then be used to calculate L and w. Unfortunately, there is no general method for correcting the steady-state probabilities, P_n.

When the Number of Servers is Infinite (M/G/∞/∞)

There are at least three situations in which it is appropriate to treat the number of servers as infinite:

1. when each customer is its own server—in other words, in a self-service system;
2. when service capacity far exceeds service demand, as in a so-called ample-server system; and
3. when we want to know how many servers are required so that customers will rarely be delayed.

The steady-state parameters for the $M/G/\infty$ queue are listed in Table 6.6. In the table, λ is the arrival rate of the Poisson arrival process, and $1/\mu$ is the expected service time of the general service-time distribution (including exponential, constant, or any other).

Example 6.16

Prior to introducing their new, subscriber-only, on-line computer information service, The Connection must plan their system capacity in terms of the number of users that can be logged in simultaneously. If the service is successful, customers are expected to log on at a rate of $\lambda = 500$ per hour, according to a Poisson process, and stay connected for an average of $1/\mu = 180$ minutes (or 3 hours). In the real system, there will be an upper limit on simultaneous users, but, for planning purposes, The Connection can pretend that the number of simultaneous users is infinite. An $M/G/\infty$ model of the system implies that the expected number of simultaneous users is $L = \lambda/\mu = 500(3) = 1500$, so a capacity greater than 1500 is certainly required. To ensure providing adequate capacity 95% of the time, The Connection could allow the number of simultaneous users to be the smallest value c such that

Table 6.6 Steady-State Parameters for the $M/G/\infty$ Queue

P_0	$e^{-\lambda/\mu}$
w	$\frac{1}{\mu}$
w_Q	0
L	$\frac{\lambda}{\mu}$
L_Q	0
P_n	$\frac{e^{-\lambda/\mu}(\lambda/\mu)^n}{n!}$, $n = 0, 1, \ldots$

$$P(L(\infty) \le c) = \sum_{n=0}^{c} P_n = \sum_{n=0}^{c} \frac{e^{-1500}(1500)^n}{n!} \ge 0.95$$

The capacity $c = 1564$ simultaneous users satisfies this requirement.

6.4.3 Multiserver Queues with Poisson Arrivals and Limited Capacity: M/M/c/N/∞

Suppose that service times are exponentially distributed at rate μ, that there are c servers, and that the total system capacity is $N \ge c$ customers. If an arrival occurs when the system is full, that arrival is turned away and does not enter the system. As in the preceding section, suppose that arrivals occur randomly according to a Poisson process with rate λ arrivals per time unit. For any values of λ and μ such that $\rho \ne 1$, the $M/M/c/N$ queue has a statistical equilibrium with steady-state characteristics as given in Table 6.7 (formulas for the case $\rho = 1$ can be found in Hillier and Lieberman [2005]).

Table 6.7 Steady-State Parameters for the $M/M/c/N$ Queue (N = System Capacity, $a = \lambda/\mu$, $\rho = \lambda/(c\mu)$)

P_0	$\left[1 + \sum_{n=1}^{c} \frac{a^n}{n!} + \frac{a^c}{c!} \sum_{n=c+1}^{N} \rho^{n-c}\right]^{-1}$
P_N	$\frac{a^N}{c!c^{N-c}} P_0$
L_Q	$\frac{P_0 a^c \rho}{c!(1-\rho)^2} [1 - \rho^{N-c} - (N-c)\rho^{N-c}(1-\rho)]$
λ_e	$\lambda(1 - P_N)$
w_Q	$\frac{L_Q}{\lambda_e}$
w	$w_Q + \frac{1}{\mu}$
L	$\lambda_e w$

The effective arrival rate, λ_e, is defined as the mean number of arrivals per time unit who enter and remain in the system. For all systems, $\lambda_e \le \lambda$; for the unlimited-capacity systems, $\lambda_e = \lambda$; but, for systems such as the present one, which turn customers away when full, $\lambda_e < \lambda$. The effective arrival rate is computed by

$$\lambda_e = \lambda (1 - P_N)$$

because $1 - P_N$ is the probability that a customer, upon arrival, will find space and be able to enter the system. When one is using Little's equations (6.17) to compute mean time spent in system w and in queue w_Q, λ must be replaced by λ_e.

Example 6.17

The unisex hair-styling shop described in Example 6.17 can hold only three customers: one in service, and two waiting. Additional customers are turned away when the system is full. The offered load is as previously determined, namely $\lambda/\mu = 2/3$.

In order to calculate the performance measures, first compute P_0:

$$P_0 = \left[1 + \frac{2}{3} + \frac{2}{3}\sum_{n=2}^{3}\left(\frac{2}{3}\right)^{n-1}\right]^{-1} = 0.415$$

The probability that there are three customers in the system (the system is full) is

$$P_N = P_3 = \frac{(2/3)^3}{1!1^2} P_0 = \frac{8}{65} = 0.123$$

Then, the average length of the queue (customers waiting for a haircut) is given by

$$L_Q = \frac{(27/65)(2/3)(2/3)}{(1-2/3)^2}[1 - (2/3)^2 - 2(2/3)^2(1-2/3)] = 0.431 \text{ customer}$$

Now, the effective arrival rate, λ_e, is given by

$$\lambda_e = 2\left(1 - \frac{8}{65}\right) = \frac{114}{65} = 1.754 \text{ customers per hour}$$

Therefore, from Little's equation, the expected time spent waiting in queue is

$$w_Q = \frac{L_Q}{\lambda_e} = \frac{28}{114} = 0.246 \text{ hour}$$

and the expected total time in the shop is

$$w = w_Q + \frac{1}{\mu} = \frac{66}{114} = 0.579 \text{ hour}$$

One last application of Little's equation gives the expected number of customers in the shop (in queue and getting a haircut) as

$$L = \lambda_e w = \frac{66}{65} = 1.015 \text{ customers}$$

Notice that $1 - P_0 = 0.585$ is the average number of customers being served or, equivalently, the probability that the single server is busy. Thus, the server utilization, or proportion of time the server is busy in the long run, is given by

$$1 - P_0 = \frac{\lambda_e}{\mu} = 0.585$$

The reader should compare these results to those of the unisex hair-styling shop before the capacity constraint was placed on the system. Specifically, in systems with limited capacity, the offered load λ/μ can assume any positive value and no longer equals the server utilization $\rho = \lambda_e/\mu$. Notice that server utilization decreases from 67% to 58.5% when the system imposes a capacity constraint.

6.5 STEADY-STATE BEHAVIOR OF FINITE-POPULATION MODELS (M/M/C/K/K)

In many practical problems, the assumption of an infinite calling population leads to invalid results because the calling population is, in fact, small. When the calling population is small, the presence of one or more customers in the system has a strong effect on the distribution of future arrivals, and the use of an infinite-population model can be misleading. Typical examples include a small group of machines that break down from time to time and require repair, or a small group of mechanics who line up at a counter for parts or tools. In the extreme case, if all the machines are broken, no new "arrivals" (breakdowns) of machines can occur; similarly, if all the mechanics are in line, no arrival is possible to the tool and parts counter. Contrast this to the infinite-population models, in which the arrival rate, λ, of customers to the system is assumed to be independent of the state of the system.

Consider a finite-calling-population model with K customers. The time between the end of one service visit and the next call for service for each member of the population is assumed to be exponentially distributed, with mean $1/\lambda$ time units; service times are also exponentially distributed, with mean $1/\mu$ time units; there are c parallel servers, and system capacity is K, so that all arrivals remain for service. Such a system is depicted in Figure 6.16.

The steady-state parameters for this model are listed in Table 6.8. An electronic spreadsheet or a symbolic calculation program is useful for evaluating these complex formulas. For example, Figure 6.17 is a procedure written for the symbolic calculation program Maple to calculate the steady-state probabilities for the $M/M/c/K/K$ queue. Another approach is to use precomputed queueing tables, such as those found in Banks and Heikes [1984], Hillier and Yu [1981], Peck and Hazelwood [1958] or Descloux [1962].

The effective arrival rate λ_e has several valid interpretations:

λ_e = long-run effective arrival rate of customers to the queue
= long-run effective arrival rate of customers entering service
= long-run rate at which customers exit from service
= long-run rate at which customers enter the calling population (and begin a new runtime)
= long-run rate at which customers exist from the calling population

Example 6.18

There are two workers who are responsible for 10 milling machines. The machines run on the average for 20 minutes, then require an average 5-minute service period, both times exponentially distributed. Therefore, $\lambda = 1/20$ and $\mu = 1/5$. Compute the various measures of performance for this system.

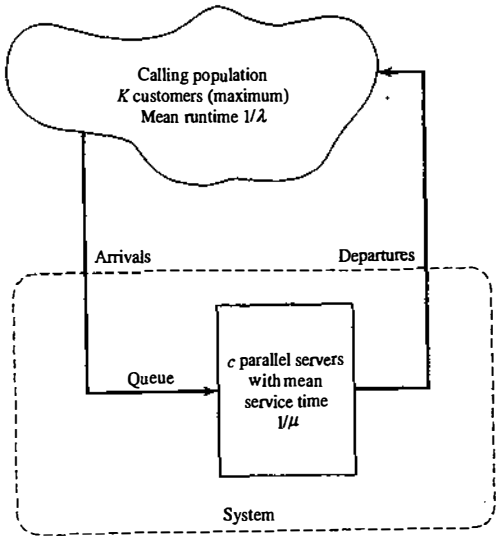


Figure 6.16 Finite-population queueing model.

Table 6.8 Steady-State Parameters for the M/M/c/K/K Queue

P_0	$\left[\sum_{n=0}^{c-1}\binom{K}{n}\left(\frac{\lambda}{\mu}\right)^n + \sum_{n=c}^{K}\frac{K!}{(K-n)!c!c^{n-c}}\left(\frac{\lambda}{\mu}\right)^n\right]^{-1}$
P_n	$\begin{cases}\binom{K}{n}\left(\frac{\lambda}{\mu}\right)^n P_0, & n = 0, 1, \ldots, c-1 \\ \frac{K!}{(K-n)!c!c^{n-c}}\left(\frac{\lambda}{\mu}\right)^n P_0, & n = c, c+1, \ldots, K\end{cases}$
L	$\sum_{n=c}^{K} nP_n$
L_Q	$\sum_{n=c+1}^{K}(n-c)P_n$
λ_e	$\sum_{n=0}^{K}(K-n)\lambda P_n$
w	L/λ_e
w_Q	L_Q/λ_e
ρ	$\frac{L-L_Q}{c} = \frac{\lambda_e}{c\mu}$

```
mmcKK := proc(lambda, mu, c, K)
         # return steady-state probabilities for M/M/c/K/K queue
         # notice that p[n+1] is P_n, n=0,..,K
         local crho, Kfac, cfac, p, n;
         p : = vector(K+1,0);
         crho := lambda/mu;
         Kfac := K!;
         cfac := c!;
         p[1] := sum((Kfac/(n!*(K-n)!))*crho^n, n=0..c-1) + sum((Kfac/(c^(n-c)*
              (K-n)!*cfac))*crho^n, n=c..K);
         p[1] := 1/p[1];
         for n from 1 to c-1
         do
            p[n+1] := p[1]*(Kfac/(n!*(K-n)!))*crho^n;
         od;
         for n from c to K
         do
            p[n+1] := p[1]*(Kfac/(c^(n-c)*(K-n)!*cfac))*crho^n;
         od;
         RETURN(evalm(p));
         end;
```

Figure 6.17 Maple procedure to calculate P_n for the $M/M/c/K/K$ queue.

All of the performance measures depend on P_0, which is

$$\left[\sum_{n=0}^{2-1}\binom{10}{n}\left(\frac{5}{20}\right)^n+\sum_{n=c}^{10}\frac{10!}{(10-n)!2!2^{n-2}}\left(\frac{5}{20}\right)^n\right]^{-1}=0.065$$

From P_0, we can obtain the other P_n, from which we can compute the average number of machines waiting for service,

$$L_Q=\sum_{n=3}^{10}(n-2)P_n=1.46 \text{ machines}$$

the effective arrival rate,

$$\lambda_e=\sum_{n=0}^{10}(10-n)\left(\frac{1}{20}\right)P_n=0.342 \text{ machines/minute}$$

and the average waiting time in the queue,

$$w_Q=L_Q/\lambda_e=4.27 \text{ minutes}$$

Similarly, we can compute the expected number of machines being serviced or waiting to be serviced,

$$L=\sum_{n=0}^{10}nP_n=3.17 \text{ machines}$$

The average number of machines being serviced is given by

$$L-L_Q=3.17-1.46=1.71 \text{ machines}$$

Each machine must be either running, waiting to be serviced, or in service, so the average number of running machines is given by

$$K-L=10-3.17=6.83 \text{ machines}$$

A question frequently asked is this: What will happen if the number of servers is increased or decreased? If the number of workers in this example increases to three ($c=3$), then the time-average number of running machines increases to

$$K-L=7.74 \text{ machines}$$

an increase of 0.91 machine, on the average.

Conversely, what happens if the number of servers decreases to one? Then the time-average number of running machines decreases to

$$K-L=3.98 \text{ machines}$$

The decrease from two servers to one has resulted in a drop of nearly three machines running, on the average. Examples 12, and 20 asks the reader to determine the optimal number of servers.

Example 6.18 illustrates several general relationships that have been found to hold for almost all queues. If the number of servers is decreased, delays, server utilization, and the probability of an arrival having to wait to begin service all increase.

6.6 NETWORKS OF QUEUES

In this chapter, we have emphasized the study of single queues of the *G/G/c/N/K* type. However, many systems are naturally modeled as networks of single queues in which customers departing from one queue may be routed to another. Example 6.1 (see, in particular, Figure 6.3) and Example 6.2 (see Figure 6.5) are illustrations.

The study of mathematical models of networks of queues is beyond the scope of this chapter; see, for instance, Gross and Harris [1997], Nelson [1995], and Kleinrock [1976]. However, a few fundamental principles are very useful for rough-cut modeling, perhaps prior to a simulation study. The following results assume a stable system with infinite calling population and no limit on system capacity:

1. Provided that no customers are created or destroyed in the queue, then the departure rate out of a queue is the same as the arrival rate into the queue, over the long run.
2. If customers arrive to queue i at rate λ_i, and a fraction $0 \le p_{ij} \le 1$ of them are routed to queue j upon departure, then the arrival rate from queue i to queue j is $\lambda_i p_{ij}$ over the long run.
3. The overall arrival rate into queue j, λ_j, is the sum of the arrival rate from all sources. If customers arrive from outside the network at rate a_j, then

$$\lambda_j=a_j+\sum_{\text{all } i}\lambda_i p_{ij}$$

4. If queue j has $c_j < \infty$ parallel servers, each working at rate μ_j, then the long-run utilization of each server is

$$\rho_j=\frac{\lambda_j}{c_j\mu_j}$$

and $\rho_j < 1$ is required for the queue to be stable.

5. If, for each queue j, arrivals from outside the network form a Poisson process with rate a_j, and if there are c_j identical servicers delivering exponentially distributed service times with mean $1/\mu_j$ (where c_j may be ∞), then, in steady state, queue j behaves like an $M/M/c_j$ queue with arrival rate $\lambda_j = a_j + \sum_{\text{all } i} \lambda_i p_{ij}$.

Example 6.19

Consider again the discount store described in Example 6.1 and shown in Figure 6.3. Suppose that customers arrive at the rate 80 per hour and that, of those arrivals, 40% choose self-service; then, the arrival rate to service center 1 is $\lambda_1 = (80)(0.40) = 32$ per hour, and the arrival rate to service center 2 is $\lambda_2 = (80)(0.6) = 48$ per hour. Suppose that each of the $c_2 = 3$ clerks at service center 2 works at the rate $\mu_2 = 20$ customers per hour. Then the long-run utilization of the clerks is

$$\rho_2 = \frac{48}{(3)(20)} = 0.8$$

All customers must see the cashier at service center 3. The overall arrival rate to service center 3 is $\lambda_3 = \lambda_1 + \lambda_2 = 80$ per hour, regardless of the service rate at service center 1, because, over the long run, the departure rate out of each service center must be equal to the arrival rate into it. If the cashier works at rate $\mu_3 = 90$ per hour, then the utilization of the cashier is

$$\rho_3 = \frac{80}{90} = 0.89$$

Example 6.20

At a Driver's License branch office, drivers arrive at the rate 50 per hour. All arrivals must first check in with one of two clerks, with the average check-in time being 2 minutes. After check in, 15% of the drivers need to take a written test that lasts approximately 20 minutes. All arrivals must wait to have their picture taken and their license produced; this station can process about 60 drivers per hour. The branch manager wants to know whether it is adding a check-in clerk or adding a new photo station that will lead to a greater reduction in customer delay.

To solve the problem, let the check-in clerks be queue 1 (with $c_1 = 2$ servers, each working at rate $\mu_1 = 30$ drivers per hour), let the testing station be queue 2 (with $c_2 = \infty$ servers, because any number of people can be taking the written test simultaneously, and service rate $\mu_2 = 3$ drivers per hour), and let the photo station be queue 3 (with $c_3 = 1$ server working at rate $\mu_3 = 60$ drivers per hour). The arrival rates to each queue are as follows:

$$\lambda_1 = a_1 + \sum_{i=1}^{3} p_{i1}\lambda_i = 50 \text{ drivers per hour}$$

$$\lambda_2 = a_2 + \sum_{i=1}^{3} p_{i2}\lambda_i = (0.15)\lambda_1 \text{ drivers per hour}$$

$$\lambda_3 = a_3 + \sum_{i=1}^{3} p_{i3}\lambda_i = (1)\lambda_2 + (0.85)\lambda_1 \text{ drivers per hour}$$

Notice that arrivals from outside the network occur only at queue 1, so $a_1 = 50$ and $a_2 = a_3 = 0$. Solving this system of equations gives $\lambda_1 = \lambda_3 = 50$ and $\lambda_2 = 7.5$.

If we approximate the arrival process as Poisson, and the service times at each queue as exponentially distributed, then the check-in clerks can be approximated as an $M/M/c_1$ queue, the testing station as an $M/M/\infty$ queue, and the photo station as an $M/M/c_3$ queue. Thus, under the current set-up, the check-in station is an $M/M/2$; using the formulas in Table 6.5 gives $w_Q = 0.0758$ hours. If we add a clerk, so that the model is $M/M/3$, the waiting time in queue drops to 0.0075 hours, a savings of 0.0683 hours or about 4.1 minutes.

The current photo station can be modeled as an $M/M/1$ queue, giving $w_Q = 0.0833$ hours; adding a second photo station ($M/M/2$) causes the time in queue to drop to 0.0035 hours, a savings of 0.0798 hours, or about 4.8 minutes. Therefore, a second photo station offers a slightly greater reduction in waiting time than does adding a third clerk.

If desired, the testing station can be analyzed by using the results for an $M/M/\infty$ queue in Table 6.6. For instance, the expected number of people taking the test at any time is $L = \lambda_2/\mu_2 = 7.5/3 = 2.5$.

6.7 SUMMARY

Queueing models have found widespread use in the analysis of service facilities, production and material-handling systems, telephone and communications systems, and many other situations where congestion or competition for scarce resources can occur. This chapter has introduced the basic concepts of queueing models and shown how simulation, and in some cases a mathematical analysis, can be used to estimate the performance measures of a system.

A simulation may be used to generate one or more artificial histories of a complex system. This simulation-generated data may, in turn, be used to estimate desired performance measures of the system. Commonly used performance measures, including L, L_Q, w, w_Q, ρ, and λ_e were introduced, and formulas were given for their estimation from data.

When simulating any system that evolves over time, the analyst must decide whether transient behavior or steady-state performance is to be studied. Simple formulas exist for the steady-state behavior of some queues, but estimating steady-state performance measures from simulation-generated data requires recognizing and dealing with the possibly deleterious effect of the initial conditions on the estimators of steady-state performance. These estimators could be severely biased (either high or low), if the initial conditions are unrepresentative of steady state or if simulation run length is too short. These estimation problems are discussed at greater length in Chapter 11.

Whether the analyst is interested in transient or in steady-state performance of a system, it should be recognized that the estimates obtained from a simulation of a stochastic queue are exactly that—estimates. Every such estimate contains random error, and a proper statistical analysis is required to assess the accuracy of the estimate. Methods for conducting such a statistical analysis are discussed in Chapters 11 and 12.

In the last three sections of this chapter, it was shown that a number of simple models can be solved mathematically. Although the assumptions behind such models might not be met exactly in a practical application, these models can still be useful in providing a rough estimate of a performance measure. In many cases, models with exponentially distributed interarrival and service times will provide a conservative estimate of system behavior. For example, if the model predicts that average waiting time, w, will be 12.7 minutes, then average waiting time in the real system is likely to be less than 12.7 minutes. The conservative nature of exponential models arises because (a) performance measures, such as w and L, are generally increasing functions of the variance of interarrival times and service times (recall the $M/G/1$ queue), and (b) the exponential distribution is fairly highly variable, having its standard deviation always equal to its mean. Thus, if the arrival process or service mechanism of the real system is less variable than exponentially distributed interarrival or service times, it is likely that the average number in the system, L, and the average time spent in system, w, will be less than what is predicted by the exponential model. Of course, if the interarrival and service times are *more* variable than exponential random variables, then the M/M queueing models could underestimate congestion.

An important application of mathematical queueing models is determining the minimum number of servers needed at a work station or service center. Quite often, if the arrival rate λ and the service rate μ are known or can be estimated, then the simple inequality $\lambda/(c\mu) < 1$ can be used to provide an initial estimate for the number of servers, c, at a work station. For a large system with many work stations, it could be quite time consuming to have to simulate every possibility $(c_1, c_2, \ldots)$ for the number of servers, c_i, at work station i. Thus, a bit of mathematical analysis rough estimates could save a great deal of computer time and analyst's time.

Finally, the qualitative behavior of the simple exponential models of queueing carries over to more complex systems. In general, it is the variability of service times and the variability of the arrival process that causes waiting lines to build up and congestion to occur. For most systems, if the arrival rate increases, or if the service rate decreases, or if the variance of service times or interarrival times increases, then the system will become more congested. Congestion can be decreased by adding more servers or by reducing the mean value and variability of service times. Simple queueing models can be a great aid in quantifying these relationships and in evaluating alternative system designs.

REFERENCES

BANKS, J., AND R. G. HEIKES [1984], *Handbook of Tables and Graphs for the Industrial Engineer and Manager*, Reston, Reston, VA.

COOPER, R. B. [1990], *Introduction to Queueing Theory*, 3d ed., George Washington University, Washington, DC.

DESCLOUX, A. [1962], *Delay Tables for Finite- and Infinite-Source Systems*, McGraw–Hill, New York.

GROSS, D., AND C. HARRIS [1997], *Fundamentals of Queueing Theory*, 3d ed., Wiley, New York.

HALL, R. W. [1991], *Queueing Methods: For Services and Manufacturing*, Prentice Hall, Englewood Cliffs, NJ.

HILLIER, F. S., AND G. J. LIEBERMAN [2005], *Introduction to Operations Research*, 8th ed., McGraw–Hill, New York.

HILLIER, F. S., AND O. S. YU [1981], *Queueing Tables and Graphs*, Elsevier North-Holland, New York.

KENDALL, D. G. [1953], "Stochastic Processes Occurring in the Theory of Queues and Their Analysis by the Method of Imbedded Markov Chains," *Annals of Mathematical Statistics*, Vol. 24, pp. 338–354.

KLEINROCK, L. [1976], *Queueing Systems, Vol 2: Computer Applications*, Wiley, New York.

LITTLE, J. D. C. [1961], "A Proof for the Queueing Formula $L = \lambda w$," *Operations Research*, Vol. 16, pp. 651–665.

NELSON, B. L. [1995], *Stochastic Modeling: Analysis & Simulation*, Dover Publications, Mineola, NY.

PECK, L. G., AND R. N. HAZELWOOD [1958], *Finite Queueing Tables*, Wiley, New York.

WAGNER, H. M. [1975], *Principles of Operations Research*, 2d ed., Prentice-Hall, Englewood Cliffs, NJ.

WINSTON, W. L. [1997], *Operations Research: Applications and Algorithms*, Duxbury Press, Pacific Grove, CA.

EXERCISES

1. Identify the calling population, customer, and server in the following queueing situations:
   (a) university library
   (b) bank teller counter
   (c) Internet router
   (d) police station
   (e) assembly line

2. A two-runway (one runway for landing, one runway for taking off) airport is being designed for propeller-driven aircraft. The time to land an airplane is known to be exponentially distributed, with a mean of 1-1/2 minutes. If airplane arrivals are assumed to occur at random, what arrival rate can be tolerated if the average wait in the sky is not to exceed 3 minutes?

3. If customers arrive for service according to Poisson distribution with a mean of 5 per day, how fast the average service time (assume exponential) must be to keep average number in the system less than 4?

4. Give some examples from real-life situations for balking and reneging.

5. Trucks arrive at a facility to be unloaded in a pattern, which can be characterized by the Poisson distribution. The average rate of arrivals is 36 per hour, and the level of service is exponentially distributed with a mean service rate of 39 trucks per hour. Compute all the relevant statistics for the system. The drivers make Rs. 9 each hour and do not unload the trucks. How much expense, on the average, is incurred by the trucking company for idle time on the part of each driver for each visit to the facility?

6. Patients arrive for a physical examination according to a Poisson process at the rate 1 per hour. The physical examination requires three stages, each one independently exponentially distributed, with a service time of 15 minutes. A patient must go through all three stages before the next patient is admitted to the treatment facility. Compute the average number of delayed patients, L_Q, for this system. (Hint: The variance of the sum of independent random variables is the sum of the variance.)

7. Suppose that mechanics arrive randomly at a tool crib according to a Poisson process with rate $\lambda = 10$ per hour. It is known that the single tool clerk serves a mechanic in 4 minutes on the average, with a standard deviation of approximately 2 minutes. Suppose that mechanics make $15.00 per hour. Estimate the steady-state average cost per hour of mechanics waiting for tools.

8. The arrival of customers at a teller counter follows Poisson with a mean of 45 per hour and teller's service time follows exponential with a mean of 1 minute. Determine the following:
   (a) Probability of having 0 customer in the system, 5 customers in the system, and 10 customers in the system.
   (b) Determine L_Q, L, W_Q, and W.

9. A machine shop repairs small electric motors, which arrive according to a Poisson process at the rate 12 per week (5-day, 40-hour workweek). An analysis of past data indicates that engines can be repaired, on the average, in 2.5 hours, with a variance of 1 hour^2. How many working hours should a customer expect to leave a motor at the repair shop (not knowing the status of the system)? If the variance of the repair time could be controlled, what variance would reduce the expected waiting time to 6.5 hours?

10. Arrivals to a self-service gasoline pump occur in a Poisson fashion at the rate 12 per hour. Service time has a distribution that averages 4 minutes, with a standard deviation of 1-1/3 minutes. What is the expected number of vehicles in the system?

11. Classic Car Care has one worker who washes cars in a four-step method—soap, rinse, dry, vacuum. The time to complete each step is exponentially distributed, with mean 9 minutes. Every car goes through every step before another car begins the process. On the average, one car every 45 minutes arrives for a wash job, according to a Poisson process. What is the average time a car waits to begin the wash job? What is the average number of cars in the car wash system? What is the average time required to wash a car?

12. Machines arrive for repair at the rate of six per hour following Poisson. The mechanics mean repair time is 15 minutes, which follows exponential distribution. The down time cost for the broken down machines per hour is Rs. 300. Mechanics are paid Rs. 60 per hour. Determine the optimal number of mechanics to be employed to minimize the total cost.

13. Given the following information for a finite calling population problem with exponentially distributed runtimes and service times:

$$K = 10$$
$$\frac{1}{\mu} = 15$$
$$\frac{1}{\lambda} = 82$$
$$c = 2$$

Compute L_Q and w_Q. Find the value of λ such that $L_Q = L/2$.

14. Suppose that Figure 6.6 represents the number in system for a last-in–first-out (LIFO) single-server system. Customers are not preempted (i.e., kicked out of service), but, upon service completion, the most recent arrival next begins service. For this LIFO system, apportion the total area under $L(t)$ to each individual customer, as was done in Figure 6.8 for the FIFO system. Using the figure, show that Equations (6.10) and (6.8) hold for the single-server LIFO system.

15. Repeat Exercise 14, but assuming that

(a) Figure 6.6 represents a FIFO system with $c = 2$ servers;
(b) Figure 6.6 represents a LIFO system with c = 2 servers:

16. Consider a *M*/*G*/1 queue with the following type of service distribution: Customers request one of two types of service, in the proportions p and $1-p$. Type i service is exponentially distributed at rate μ_i, $i = 1, 2$. Let X_i denote a type-i service time and X an arbitrary service time. Then $E(X_i) = 1/\mu_i$, $V(X_i) = 1/\mu_i^2$ and

$$X = \begin{cases} X_1 \text{ with probability } p \\ X_2 \text{ with probability } (1-p) \end{cases}$$

The random variable X is said to have a hyperexponential distribution with parameters (μ_1, μ_2, p).

(a) Show that $E(X) = p/\mu_1 + (1-p)/\mu_2$ and $E(X^2) = 2p/\mu_1^2 + 2(1-p)/\mu_2^2$.
(b) Use $V(X) = E(X^2) - [E(X)]^2$ to show $V(X) = 2p/\mu_1^2 + 2(1-p)/\mu_2^2 - [p/\mu_1 + (1-p)/\mu_2]^2$.
(c) For any hyperexponential random variable, if $\mu_1 \neq \mu_2$ and $0 < p < 1$, show that its coefficient of variation is greater than 1—that is, $(cv)^2 = V(X)/[E(X)]^2 > 1$. Thus, the hyperexponential distribution provides a family of statistical models for service times that are more variable than exponentially distributed service times. *Hint*: The algebraic expression for $(cv)^2$, by using parts (a) and (b), can be manipulated into the form $(cv)^2 = 2p(1-p)(1/\mu_1 - 1/\mu_2)^2/[E(X)]^2 + 1$.
(d) Many choices of μ_1, μ_2, and p lead to the same overall mean $E(X)$ and $(cv)^2$. If a distribution with mean $E(X) = 1$ and coefficient of variation $cv = 2$ is desired, find values of μ_1, μ_2, and p to achieve this. *Hint*: Choose $p = 1/4$ arbitrarily; then solve the following equations for μ_1 and μ_2.

$$\frac{1}{4\mu_1} + \frac{3}{4\mu_2} = 1$$
$$\frac{3}{8}\left(\frac{1}{\mu_1} - \frac{1}{\mu_2}\right)^2 + 1 = 4$$

17. Orders are expected to arrive at a machining center according to Poisson process at a mean rate of 30 per hour. The management has an option of two machines M1 (fast but expensive) and M2 (slow inexpensive). Both machines would have an exponential distribution for machining times with M1 having a mean of 1.2 minutes and M2 having a mean of 1.5 minutes. The profit per year is given by Rs. 72,000/*W*, where *W* is the expected waiting time (in minutes) for the orders in the system. Determine the upper bound on the difference in the average yearly cost that would justify buying M1 rather than M2.

18. In Example 6.18, increase the number of machines by 2, then compare the systems with $c = 1$, $c = 2$, and $c = 3$ servers on the basis of server utilization ρ (the proportion of time a typical server is busy).

19. Vehicles pass through a toll gate at a rate of 90 per hour. The average time to pass through the gate is 36 seconds. The arrival rate and service rate follow Poisson distribution. There is a complaint that the vehicles wait for a long duration. The authorities are willing to install one more gate to reduce the average time to pass through to 30 seconds, if the idle time of the toll gate is less than 10% and the present average queue length at the gate is more than five vehicles. Check whether the installation of the second gate is justified.

20. The arrival of employees at a tool crib can be described by a Poisson distribution. Service times are exponentially distributed. The rate of arrival averages 45 machinists per hour, while an attendant can serve an average of 50 men per hour. The machinists are paid Rs. 24 per hour, while the attendants are paid Rs. 15 per hour. Find the optimum number of attendants to place in the crib, assuming 8 hours and 200 days per year.

21. This problem is based on Case 8.1 in Nelson [1995]. A large consumer shopping mall is to be constructed. During busy times, the arrival rate of cars is expected to be 1000 per hour, and studies at other malls suggest that customers will spend 3 hours, on average, shopping. The mall designers would like to have sufficient parking so that there are enough spaces 99.9% of the time. How many spaces should they have? Hint: Model the system as an *M*/*G*/∞ queue where the spaces are servers, and find out how many spaces are adequate with probability 0.999.

22. In Example 6.19, suppose that the overall arrival rate is expected to increase to 160 per hour. If the service rates do not change, how many clerks will be needed at service centers 2 and 3, just to keep up with the customer load?

23. A small copy shop has a self-service copier. Currently there is room for only 4 people to line up for the machine (including the person using the machine); when there are more than 4 people, then the additional people must line up outside the shop. The owners would like to avoid having people line up outside the shop, as much as possible. For that reason, they are thinking about adding a second self-service copier. Self-service customers have been observed to arrive at the rate 24 per hour, and they use the machine 2 minutes, on average. Assess the impact of adding another copier. Carefully state any assumptions or approximations you make.

24. In an *N* machine one operator environment, five automatic machines are attended by one operator. Every time a machine completes a batch, the operator must reset it before a new batch is started. The time to complete a batch run is exponential with a mean of 45 minutes. The setup time is also exponential with a mean of 8 minutes. Determine

(a) the average number of machines that are waiting for set up.
(b) the probability that all the machines are working.
(c) the average time a machine is down.

25. Search the web and find applications of queueing theory in production activities.

26. Study the effect of *pooling servers* (having multiple servers draw from a single queue, rather than each having its own queue) by comparing the performance measures for two M/M/1 queues, each with arrival rate λ and service rate μ, to an M/M/2 queue with arrival rate 2λ and service rate μ for each server.

27. A repair and inspection facility consists of two stations: a repair station with two technicians, and an inspection station with 1 inspector. Each repair technician works at the rate 3 items per hour; the inspector can inspect 8 items per hour. Approximately 10% of all items fail inspection and are sent back to the repair station. (This percentage holds even for items that have been repaired two or more times.) If items arrive at the rate 5 per hour, what is the long-run expected delay that items experience at each of the two stations, assuming a Poisson arrival process and exponentially distributed service times? What is the maximum arrival rate that the system can handle without adding personnel?

Part III

Random Numbers

7

Random-Number Generation

Random numbers are a necessary basic ingredient in the simulation of almost all discrete systems. Most computer languages have a subroutine, object, or function that will generate a random number. Similarly, simulation languages generate random numbers that are used to generate event times and other random variables. In this chapter, the generation of random numbers and their subsequent testing for randomness is described. Chapter 8 shows how random numbers are used to generate a random variable with any desired probability distribution.

7.1 PROPERTIES OF RANDOM NUMBERS

A sequence of random numbers, $R_1, R_2, \ldots$, must have two important statistical properties: uniformity and independence. Each random number R_i must be an independent sample drawn from a continuous uniform distribution between zero and 1—that is, the pdf is given by

$$f(x) = \begin{cases} 1, & 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$$

This density function is shown in Figure 7.1. The expected value of each R_i is given by

$$E(R) = \int_0^1 x\,dx = \frac{x^2}{2}\Big|_0^1 = \frac{1}{2}$$

and the variance is given by

$$V(R) = \int_0^1 x^2\,dx - \left[E(R)\right]^2 = \frac{x^3}{3}\Big|_0^1 - \left(\frac{1}{2}\right)^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}$$

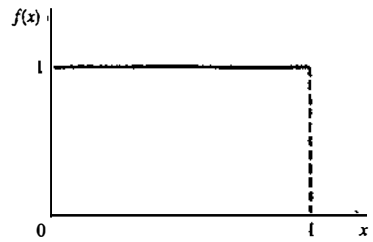


Figure 7.1 pdf for random numbers.

Some consequences of the uniformity and independence properties are the following:

1. If the interval [0, 1] is divided into n classes, or subintervals of equal length, the expected number of observations in each interval is N/n, where N is the total number of observations.
2. The probability of observing a value in a particular interval is independent of the previous values drawn.

7.2 GENERATION OF PSEUDO-RANDOM NUMBERS

Notice that the title of this section has the word "pseudo" in it. "Pseudo" means false, so false random numbers are being generated! In this instance, "pseudo" is used to imply that the very act of generating random numbers by a known method removes the potential for true randomness. If the method is known, the set of random numbers can be replicated. Then an argument can be made that the numbers are not truly random. The goal of any generation scheme, however, is to produce a sequence of numbers between 0 and 1 that simulates, or imitates, the ideal properties of uniform distribution and independence as closely as possible.

To be sure, in the generation of pseudo-random numbers, certain problems or errors can occur. These errors, or departures from ideal randomness, are all related to the properties stated previously. Some examples of such departures include the following:

1. The generated numbers might not be uniformly distributed.
2. The generated numbers might be discrete-valued instead of continuous-valued.
3. The mean of the generated numbers might be too high or too low.
4. The variance of the generated numbers might be too high or too low.
5. There might be dependence. The following are examples:
   (a) autocorrelation between numbers;
   (b) numbers successively higher or lower than adjacent numbers;
   (c) several numbers above the mean followed by several numbers below the mean.

Departures from uniformity and independence for a particular generation scheme often can be detected by such tests as those described in Section 7.4. If such departures are detected, the generation scheme should be dropped in favor of an acceptable generator. Generators that pass the tests in Section 7.4 and tests even more stringent have been developed; thus, there is no excuse for using a generator that has been found to be defective.

Usually, random numbers are generated by a digital computer, as part of the simulation. There are numerous methods that can be used to generate the values. Before we describe some of these methods, or *routines*, there are a number of important considerations that we should mention:

1. The routine should be fast. Individual computations are inexpensive, but simulation could require many millions of random numbers. The total cost can be managed by selecting a computationally efficient method of random-number generation.
2. The routine should be portable to different computers—and, ideally, to different programming languages. This is desirable so that the simulation program will produce the same results wherever it is executed.
3. The routine should have a sufficiently long cycle. The cycle length, or period, represents the length of the random number sequence before previous numbers begin to repeat themselves in an earlier order. Thus, if 10,000 events are to be generated, the period should be many times that long.
   A special case of cycling is degenerating. A routine degenerates when the same random numbers appear repeatedly. Such an occurrence is certainly unacceptable. This can happen rapidly with some methods.
4. The random numbers should be replicable. Given the starting point (or conditions) it should be possible to generate the same set of random numbers, completely independent of the system that is being simulated. This is helpful for debugging purposes and is a means of facilitating comparisons between systems (see Chapter 12). For the same reasons, it should be possible to easily specify different starting points, widely separated, within the sequence.
5. Most important, the generated random numbers should closely approximate the ideal statistical properties of uniformity and independence.

Inventing techniques that seem to generate random numbers is easy; inventing techniques that really do produce sequences that appear to be independent, uniformly distributed random numbers is incredibly difficult. There is now a vast literature and rich theory on the topic, and many hours of testing have been devoted to establishing the properties of various generators. Even when a technique is known to be theoretically sound, it is seldom easy to implement it in a way that will be fast and portable. The goal of this chapter is to make the reader aware of the central issues in random-number generation, to enhance understanding and to show some of the techniques that are used by those working in this area.

7.3 TECHNIQUES FOR GENERATING RANDOM NUMBERS

The linear congruential method of Section 7.3.1 is the most widely used technique for generating random numbers, so we describe it in detail. We also report an extension of this method that yields sequences with a longer period. Many other methods have been proposed, and they are reviewed in Bratley, Fox, and Schrage [1996], Law and Kelton [2000], and Ripley [1987].

7.3.1 Linear Congruential Method

The linear congruential method, initially proposed by Lehmer [1951], produces a sequence of integers, $X_1, X_2, \ldots$ between zero and $m-1$ by following a recursive relationship:

$$X_{i+1} = (aX_i + c) \bmod m, \quad i = 0, 1, 2, \ldots \tag{7.1}$$

The initial value X_0 is called the seed, a is called the multiplier, c is the increment, and m is the modulus. If $c \neq 0$ in Equation (7.1), then the form is called the *mixed congruential method*. When $c = 0$, the form is known as the *multiplicative congruential method*. The selection of the values for a, c, m, and X_0 drastically

affects the statistical properties and the cycle length. Variations of Equation (7.1) are quite common in the computer generation of random numbers. An example will illustrate how this technique operates.

Example 7.1

Use the linear congruential method to generate a sequence of random numbers with $X_0 = 27$, $a = 17$, $c = 43$, and $m = 100$. Here, the integer values generated will all be between zero and 99 because of the value of the modulus. Also, notice that random integers are being generated rather than random numbers. These random integers should appear to be uniformly distributed on the integers zero to 99. Random numbers between zero and 1 can be generated by

$$R_i = \frac{X_i}{m}, \quad i = 1, 2, \ldots \tag{7.2}$$

The sequence of X_i and subsequent R_i values is computed as follows:

$$X_0 = 27$$

$$X_1 = (17 \cdot 27 + 43) \bmod 100 = 502 \bmod 100 = 2$$

$$R_1 = \frac{2}{100} = 0.02$$

$$X_2 = (17 \cdot 2 + 43) \bmod 100 = 77 \bmod 100 = 77$$

$$R_2 = \frac{77}{100} = 0.77$$

$$X_3 = (17 \cdot 77 + 43) \bmod 100 = 1352 \bmod 100 = 52$$

$$R_3 = \frac{52}{100} = 0.52$$

$$\vdots$$

Recall that $a = b \bmod m$ provided that $(b - a)$ is divisible by m with no remainder. Thus, $X_1 = 502 \bmod 100$, but 502/100 equals 5 with a remainder of 2, so that $X_1 = 2$. In other words, $(502 - 2)$ is evenly divisible by $m = 100$, so $X_1 = 502$ "reduces" to $X_1 = 2 \bmod 100$. (A shortcut for the modulo, or reduction operation for the case $m = 10^b$, a power of 10, is illustrated in Example 7.3.)

The ultimate test of the linear congruential method, as of any generation scheme, is how closely the generated numbers $R_1, R_2, \ldots$ approximate uniformity and independence. There are, however, several secondary properties that must be considered. These include *maximum density* and *maximum period.*

First, notice that the numbers generated from Equation (7.2) assume values only from the set $I = \{0, 1/m, 2/m, \ldots, (m-1)/m\}$, because each X_i is an integer in the set $\{0, 1, 2, \ldots, m-1\}$. Thus, each R_i is discrete on I, instead of continuous on the interval [0, 1]. This approximation appears to be of little consequence if the modulus m is a very large integer. (Values such as $m = 2^{31} - 1$ and $m = 2^{48}$ are in common use in generators appearing in many simulation languages.) By maximum density is meant that the values assumed by R_i, $i = 1, 2, \ldots$, leave no large gaps on [0, 1].

Second, to help achieve maximum density, and to avoid cycling (i.e., recurrence of the same sequence of generated numbers) in practical applications, the generator should have the largest possible period. Maximal period can be achieved by the proper choice of a, c, m, and X_0 [Fishman, 1978; Law and Kelton, 2000].

- For m a power of 2, say $m = 2^b$, and $c \neq 0$, the longest possible period is $P = m = 2^b$, which is achieved whenever c is relatively prime to m (that is, the greatest common factor of c and m is 1) and $a = 1 + 4k$, where k is an integer.

- For m a power of 2, say $m = 2^b$, and $c = 0$, the longest possible period is $P = m/4 = 2^{b-2}$, which is achieved if the seed X_0 is odd and if the multiplier, a, is given by $a = 3 + 8k$ or $a = 5 + 8k$, for some $k = 0, 1, \ldots$.
- For m a prime number and $c = 0$, the longest possible period is $P = m - 1$, which is achieved whenever the multiplier, a, has the property that the smallest integer k such that $a^k - 1$ is divisible by m is $k = m - 1$.

Example 7.2

Using the multiplicative congruential method, find the period of the generator for $a = 13$, $m = 2^6 = 64$ and $X_0 = 1$, 2, 3, and 4. The solution is given in Table 7.1. When the seed is 1 or 3, the sequence has period 16. However, a period of length eight is achieved when the seed is 2 and a period of length four occurs when the seed is 4.

In Example 7.2, $m = 2^6 = 64$ and $c = 0$. The maximal period is therefore $P = m/4 = 16$. Notice that this period is achieved by using odd seeds, $X_0 = 1$ and $X_0 = 3$; even seeds, $X_0 = 2$ and $X_0 = 4$, yield the periods eight and four, respectively, both less than the maximum. Notice that $a = 13$ is of the form $5 + 8k$ with $k = 1$, as is required to achieve maximal period.

When $X_0 = 1$, the generated sequence assumes values from the set {1, 5, 9, 13, ..., 53, 57, 61}. The "gaps" in the sequence of generated random numbers, R_i, are quite large (i.e., the gap is 5/64 − 1/64 or 0.0625). Such a gap gives rise to concern about the density of the generated sequence.

The generator in Example 7.2 is not viable for any application—its period is too short, and its density is insufficient. However, the example shows the importance of properly choosing a, c, m, and X_0.

Speed and efficiency in using the generator on a digital computer is also a selection consideration. Speed and efficiency are aided by use of a modulus, m, which is either a power of 2 or close to a power of 2. Since most digital computers use a binary representation of numbers, the modulo, or remaindering, operation of Equation (7.1) can be conducted efficiently when the modulo is a power of 2 (i.e., $m = 2^b$). After ordinary arithmetic yields a value for $aX_i + c$, X_{i+1} is obtained by dropping the leftmost binary digits in $aX_i + c$ and

Table 7.1 Period Determination Using Various Seeds

i	X_i	X_i	X_i	X_i
0	1	2	3	4
1	13	26	39	52
2	41	18	59	36
3	21	42	63	20
4	17	34	51	4
5	29	58	23	
6	57	50	43	
7	37	10	47	
8	33	2	35	
9	45		7	
10	9		27	
11	53		31	
12	49		19	
13	61		55	
14	25		11	
15	5		15	
16	1		3	

then using only the b rightmost binary digits. The following example illustrates, by analogy, this operation using $m = 10^b$, because most human beings think in decimal representation.

Example 7.3

Let $m = 10^2 = 100$, $a = 19$, $c = 0$, and $X_0 = 63$, and generate a sequence of random integers using Equation (7.1).

$$X_0 = 63$$
$$X_1 = (19)(63) \bmod 100 = 1197 \bmod 100 = 97$$
$$X_2 = (19)(97) \bmod 100 = 1843 \bmod 100 = 43$$
$$X_3 = (19)(43) \bmod 100 = 817 \bmod 100 = 17$$
$$\vdots$$

When m is a power of 10, say $m = 10^b$, the modulo operation is accomplished by saving the b rightmost (decimal) digits. By analogy, the modulo operation is most efficient for binary computers when $m = 2^b$ for some $b > 0$.

Example 7.4

The last example in this section is in actual use. It has been extensively tested (Learmonth and Lewis, 1973; Lewis *et al.*, 1969]. The values for a, c, and m have been selected to ensure that the characteristics desired in a generator are most likely to be achieved. By changing X_0, the user can control the repeatability of the stream.

Let $a = 7^5 = 16{,}807$; $m = 2^{31} - 1 = 2{,}147{,}483{,}647$ (a prime number); and $c = 0$. These choices satisfy the conditions that insure a period of $P = m - 1$ (well over 2 billion). Further, specify the seed $X_0 = 123{,}457$. The first few numbers generated are as follows:

$$X_1 = 7^5(123{,}457) \bmod (2^{31} - 1) = 2{,}074{,}941{,}799 \bmod (2^{31} - 1)$$
$$X_1 = 2{,}074{,}941{,}799$$
$$R_1 = \frac{X_1}{2^{31}} = 0.9662$$
$$X_2 = 7^5(2{,}074{,}941{,}799) \bmod (2^{31} - 1) = 559{,}872{,}160$$
$$R_2 = \frac{X_2}{2^{31}} = 0.2607$$
$$X_3 = 7^5(559{,}872{,}160) \bmod (2^{31} - 1) = 1{,}645{,}535{,}613$$
$$R_3 = \frac{X_3}{2^{31}} = 0.7662$$
$$\vdots$$

Notice that this routine divides by $m + 1$ instead of m; however, for such a large value of m, the effect is negligible.

7.3.2 Combined Linear Congruential Generators

As computing power has increased, the complexity of the systems that we are able to simulate has also increased. A random-number generator with period $2^{31} - 1 \approx 2 \times 10^9$, such as the popular generator described in Example 7.4, is no longer adequate for all applications. Examples include the simulation of highly reliable systems, in which hundreds of thousands of elementary events must be simulated to observe even a single failure

event, and the simulation of complex computer networks, in which thousands of users are executing hundreds of programs. An area of current research is the deriving of generators with substantially longer periods.

One fruitful approach is to combine two or more multiplicative congruential generators in such a way that the combined generator has good statistical properties and a longer period. The following result from L'Ecuyer [1988] suggests how this can be done:

If $W_{i,1}, W_{i,2}, \ldots, W_{i,k}$ are any independent, discrete-valued random variables (not necessarily identically distributed), but one of them, say $W_{i,1}$, is uniformly distributed on the integers from 0 to $m_1 - 2$, then

$$W_i = \left(\sum_{j=1}^{k} W_{i,j} \right) \bmod m_1 - 1$$

is uniformly distributed on the integers from 0 to $m_1 - 2$.

To see how this result can be used to form combined generators, let $X_{i,1}, X_{i,2}, \ldots, X_{i,k}$ be the ith output from k different multiplicative congruential generators, where the jth generator has prime modulus m_j and the multiplier a_j is chosen so that the period is $m_j - 1$. Then the jth generator is producing integers $X_{i,j}$ that are approximately uniformly distributed on the integers from 1 to $m_j - 1$, and $W_{i,j} = X_{i,j} - 1$ is approximately uniformly distributed on the integers from 0 to $m_j - 2$. L'Ecuyer [1988] therefore suggests combined generators of the form

$$X_i = \left(\sum_{j=1}^{k} (-1)^{j-1} X_{i,j} \right) \bmod m_1 - 1$$

with

$$R_i = \begin{cases} \dfrac{X_i}{m_1}, & X_i > 0 \\ \dfrac{m_1 - 1}{m_1}, & X_i = 0 \end{cases}$$

Notice that the "$(-1)^{j-1}$" coefficient implicitly performs the subtraction $X_{i,1} - 1$; for example, if $k = 2$ then $(-1)^0(X_{i,1} - 1) - (-1)^1(X_{i,2} - 1) = \sum_{j=1}^{2} (-1)^{j-1} X_{i,j}$.

The maximum possible period for such a generator is

$$p = \frac{(m_1 - 1)(m_2 - 1) \cdots (m_k - 1)}{2^{k-1}}$$

which is achieved by the generator described in the next example.

Example 7.5

For 32-bit computers, L'Ecuyer [1988] suggests combining $k = 2$ generators with $m_1 = 2{,}147{,}483{,}563$, $a_1 = 40{,}014$, $m_2 = 2{,}147{,}483{,}399$ and $a_2 = 40{,}692$. This leads to the following algorithm:

1. Select seed $X_{1,0}$ in the range [1,2,14,74,83,562] for the first generator, and seed $X_{2,0}$ in the range [1,2,14,74,83,398] for the second.
   Set $j = 0$.
2. Evaluate each individual generator.

$$X_{1,j+1} = 40{,}014\, X_{1,j} \bmod 2{,}147{,}483{,}563$$
$$X_{2,j+1} = 40{,}692\, X_{2,j} \bmod 2{,}147{,}483{,}399$$

3. Set

$$X_{j+1} = (X_{1,j+1} - X_{2,j+1}) \bmod 2{,}147{,}483{,}562$$

4. Return

$$R_{j+1} = \begin{cases} \dfrac{X_{j+1}}{2{,}147{,}483{,}563}, & X_{j+1} > 0 \\ \dfrac{2{,}147{,}483{,}562}{2{,}147{,}483{,}563}, & X_{j+1} = 0 \end{cases}$$

5. Set $j = j + 1$ and go to step 2.

This combined generator has period $(m_1 - 1)(m_2 - 1)/2 \approx 2 \times 10^{18}$. Perhaps surprisingly, even such a long period might not be adequate for all applications. See L'Ecuyer [1996, 1999] and L'Ecuyer *et al.* [2002] for combined generators with periods as long as $2^{191} \approx 3 \times 10^{57}$.

7.3.3 Random-Number Streams

The seed for a linear congruential random-number generator (seeds, in the case of a combined linear congruential generator) is the integer value X_0 that initializes the random-number sequence. Since the sequence of integers $X_0, X_1, \ldots, X_P, X_0, X_1, \ldots$ produced by a generator repeats, any value in the sequence could be used to "seed" the generator.

For a linear congruential generator, a random-number *stream* is nothing more than a convenient way to refer to a starting seed taken from the sequence $X_0, X_1, \ldots, X_P$ (for a combined generator, starting seeds for all of the basic generators are required); typically these starting seeds are far apart in the sequence. For instance, if the streams are b values apart, then stream i could be defined by starting seed

$$S_i = X_{b(i-1)}$$

for $i = 1, 2, \ldots, \lfloor P/b \rfloor$. Values of $b = 100{,}000$ were common in older generators, but values as large as $b = 10^{37}$ are in use in modern combined linear congruential generators. (See, for instance, L'Ecuyer *et al.* [2002] for the implementation of such a generator.) Thus, a single random-number generator with k streams acts like k distinct virtual random-number generators, provided that the current value of seed for each stream is maintained. Exercise 21 illustrates one way to create streams that are widely separated in the random-number sequence.

In Chapter 12, we will consider the problem of comparing two or more alternative systems via simulation, and we will show that there are advantages to dedicating portions of the pseudorandom number sequence to the same purpose in each of the simulated systems. For instance, in comparing the efficiency of several queueing systems, a fairer comparison will be achieved if all of the simulated systems experience exactly the same sequence of customer arrivals. Such synchronization can be achieved by assigning a specific stream to generate arrivals in each of the queueing simulations. If the starting seeds for the streams are spaced far enough apart, then this has the same effect as having a distinct random-number generator whose only purpose is to generate customer arrivals.

7.4 TESTS FOR RANDOM NUMBERS

The desirable properties of random numbers—uniformity and independence—were discussed in Section 7.1. To check on whether these desirable properties have been achieved, a number of tests can be performed.

(Fortunately, the appropriate tests have already been conducted for most commercial simulation software.) The tests can be placed in two categories, according to the properties of interest: uniformity, and independence. A brief description of two types of tests is given in this chapter:

1. *Frequency test.* Uses the Kolmogorov–Smirnov or the chi-square test to compare the distribution of the set of numbers generated to a uniform distribution.
2. *Autocorrelation test.* Tests the correlation between numbers and compares the sample correlation to the expected correlation, zero.

In testing for uniformity, the hypotheses are as follows:

$$H_0: \; R_i \sim U[0, 1]$$
$$H_1: \; R_i \not\sim U[0, 1]$$

The null hypothesis, H_0, reads that the numbers are distributed uniformly on the interval [0, 1]. Failure to reject the null hypothesis means that evidence of nonuniformity has not been detected by this test. This does not imply that further testing of the generator for uniformity is unnecessary.

In testing for independence, the hypotheses are as follows:

$$H_0: \; R_i \sim \text{independently}$$
$$H_1: \; R_i \not\sim \text{independently}$$

This null hypothesis, H_0, reads that the numbers are independent. Failure to reject the null hypothesis means that evidence of dependence has not been detected by this test. This does not imply that further testing of the generator for independence is unnecessary.

For each test, a level of significance α must be stated. The level α is the probability of rejecting the null hypothesis when the null hypothesis is true:

$$\alpha = P(\text{reject } H_0 \,|\, H_0 \text{ true})$$

The decision maker sets the value of a for any test. Frequently, α is set to 0.01 or 0.05.

If several tests are conducted on the same set of numbers, the probability of rejecting the null hypothesis on at least one test, by chance alone [i.e., making a Type I (α) error], increases. Say that $\alpha = 0.05$ and that five different tests are conducted on a sequence of numbers. The probability of rejecting the null hypothesis on at least one test, by chance alone, could be as large as 0.25.

Similarly, if one test is conducted on many sets of numbers from a generator, the probability of rejecting the null hypothesis on at least one test by chance alone [i.e., making a Type I (α) error], increases as more sets of numbers are tested. For instance, if 100 sets of numbers were subjected to the test, with $\alpha = 0.05$, it would be expected that five of those tests would be rejected by chance alone. If the number of rejections in 100 tests is close to 100α, then there is no compelling reason to discard the generator. The concept discussed in this and the preceding paragraph is discussed further at the conclusion of Example 7.8.

If one of the well-known simulation languages or random-number generators is used, it is probably unnecessary to apply the tests just mentioned and described in Sections 7.4.1 and 7.4.2. However, random-number generators frequently are added to software that is not specifically developed for simulation, such as spreadsheet programs, symbolic/numerical calculators, and programming languages. If the generator that is at hand is not explicitly known or documented, then the tests in this chapter should be applied to many samples of numbers from the generator. Some additional tests that are commonly used, but are not covered here, are Good's serial test for sampling numbers [1953, 1967], the median-spectrum test [Cox and Lewis, 1966; Durbin, 1967], the runs test [Law and Kelton 2000] and a variance heterogeneity test [Cox

and Lewis, 1966]. Even if a set of numbers passes all the tests, there is no guarantee of randomness; it is always possible that some underlying pattern has gone undetected.

In this book, we emphasize empirical tests that are applied to actual sequences of numbers produced by a generator. Because of the extremely long period of modern pseudo-random-number generators, as described in Section 7.3.2, it is no longer possible to apply these tests to a significant portion of the period of such generators. The tests can be used as a check if one encounters a generator with completely unknown properties (perhaps one that is undocumented and buried deep in a software package), but they cannot be used to establish the quality of a generator throughout its period. Fortunately, there are also families of theoretical tests that evaluate the choices for m, a, and c without actually generating any numbers, the most common being the spectral test. Many of these tests assess how k-tuples of random numbers fill up a k-dimensional unit cube. These tests are beyond the scope of this book; see, for instance, Ripley [1987].

In the examples of tests that follow, the hypotheses are not restated. The hypotheses are as indicated in the foregoing paragraphs. Although few simulation analysts will need to perform these tests, every simulation user should be aware of the qualities of a good random-number generator.

7.4.1 Frequency Tests

A basic test that should always be performed to validate a new generator is the test of uniformity. Two different methods of testing are available. They are the Kolmogorov-Smirnov and the chi-square test. Both of these tests measure the degree of agreement between the distribution of a sample of generated random numbers and the theoretical uniform distribution. Both tests are based on the null hypothesis of no significant difference between the sample distribution and the theoretical distribution.

1. *The Kolmogorov–Smirnov test.* This test compares the continuous cdf, $F(x)$, of the uniform distribution with the empirical cdf, $S_N(x)$, of the sample of N observations. By definition,

$$F(x) = x, \quad 0 \le x \le 1$$

If the sample from the random-number generator is $R_1, R_2, \ldots, R_N$, then the empirical cdf, $S_N(x)$, is defined by

$$S_N(x) = \frac{\text{number of } R_1, R_2, \ldots, R_N \text{ which are} \le x}{N}$$

As N becomes larger, $S_N(x)$ should become a better approximation to $F(x)$, provided that the null hypothesis is true.

In Section 5.6, empirical distributions were described. The cdf of an empirical distribution is a step function with jumps at each observed value. This behavior was illustrated by Example 5.35.

The Kolmogorov-Smirnov test is based on the largest absolute deviation between $F(x)$ and $S_N(x)$ over the range of the random variable—that is, it is based on the statistic

$$D = \max|F(x) - S_N(x)| \tag{7.3}$$

The sampling distribution of D is known; it is tabulated as a function of N in Table A.8. For testing against a uniform cdf, the test procedure follows these steps:

Step 1. Rank the data from smallest to largest. Let $R_{(i)}$ denote the ith smallest observation, so that

$$R_{(1)} \le R_{(2)} \le \cdots \le R_{(N)}$$

Step 2. Compute

$$D^+ = \max_{1 \le i \le N} \left\{ \frac{i}{N} - R_{(i)} \right\}$$

$$D^- = \max_{1 \le i \le N} \left\{ R_{(i)} - \frac{i-1}{N} \right\}$$

Step 3. Compute $D = \max(D^+, D^-)$.

Step 4. Locate in Table A.8 the critical value, D_α, for the specified significance level α and the given sample size N.

Step 5. If the sample statistic D is greater than the critical value, D_α, the null hypothesis that the data are a sample from a uniform distribution is rejected. If $D \le D_\alpha$, conclude that no difference has been detected between the true distribution of $\{R_1, R_2, \ldots, R_N\}$ and the uniform distribution.

Example 7.6

Suppose that the five numbers 0.44, 0.81, 0.14, 0.05, 0.93 were generated, and it is desired to perform a test for uniformity by using the Kolmogorov–Smirnov test with the level of significance $\alpha = 0.05$. First, the numbers must be ranked from smallest to largest. The calculations can be facilitated by use of Table 7.2. The top row lists the numbers from smallest ($R_{(1)}$) to largest ($R_{(5)}$). The computations for D^+, namely $i/N - R_{(i)}$, and for D^-, namely $R_{(i)} - (i-1)/N$, are easily accomplished by using Table 7.2. The statistics are computed as $D^+ = 0.26$ and $D^- = 0.21$. Therefore, $D = \max\{0.26, 0.21\} = 0.26$. The critical value of D, obtained from Table A.8 for $\alpha = 0.05$ and $N = 5$, is 0.565. Since the computed value, 0.26, is less than the tabulated critical value, 0.565, the hypothesis that the distribution of the generated numbers is the uniform distribution is not rejected.

The calculations in Table 7.2 are illustrated in Figure 7.2, where the empirical cdf, $S_N(x)$, is compared to the uniform cdf, $F(x)$. It can be seen that D^+ is the largest deviation of $S_N(x)$ above $F(x)$, and that D^- is the largest deviation of $S_N(x)$ below $F(x)$. For example, at $R_{(3)}$, the value of D^+ is given by $3/5 - R_{(3)} = 0.60 - 0.44 = 0.16$, and that of D^- is given by $R_{(3)} - 2/5 = 0.44 - 0.40 = 0.04$. Although the test statistic D is defined by Equation (7.3) as the maximum deviation over all x, it can be seen from Figure 7.2 that the maximum deviation will always occur at one of the jump points $R_{(1)}, R_{(2)}, \ldots$; thus, the deviation at other values of x need not be considered.

2. *The chi-square test.* The chi-square test uses the sample statistic

$$\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

where O_i is the observed number in the ith class, E_i is the expected number in the ith class, and n is the number of classes. For the uniform distribution, E_i, the expected number in each class is given by

$$E_i = \frac{N}{n}$$

Table 7.2 Calculations for Kolmogorov–Smirnov Test

$R_{(i)}$	0.05	0.14	0.44	0.81	0.93
i/N	0.20	0.40	0.60	0.80	1.00
$i/N - R_{(i)}$	0.15	0.26	0.16	—	0.07
$R_{(i)} - (i-1)/N$	0.05	—	0.04	0.21	0.13

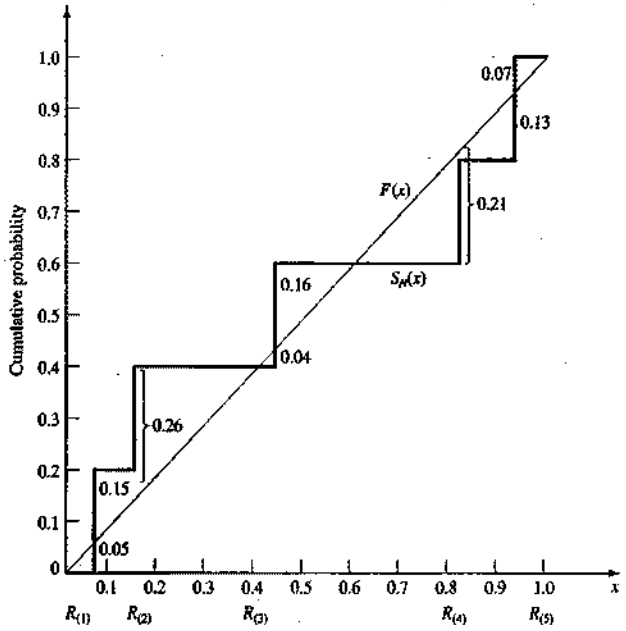


Figure 7.2 Comparison of $F(x)$ and $S_N(x)$.

for equally spaced classes, where N is the total number of observations. It can be shown that the sampling distribution of χ_0^2 is approximately the chi-square distribution with $n - 1$ degrees of freedom.

Example 7.7

Use the chi-square test with $\alpha = 0.05$ to test for whether the data shown next are uniformly distributed. Table 7.3 contains the essential computations. The test uses $n = 10$ intervals of equal length, namely [0, 0.1), [0.1, 0.2), ..., [0.9, 1.0). The value of χ_0^2 is 3.4. This is compared with the critical value $\chi^2_{0.05,9} = 16.9$ from Table A.6. Since χ_0^2 is much smaller than the tabulated value of $\chi^2_{0.05,9}$, the null hypothesis of a uniform distribution is not rejected.

0.34	0.90	0.25	0.89	0.87	0.44	0.12	0.21	0.46	0.67
0.83	0.76	0.79	0.64	0.70	0.81	0.94	0.74	0.22	0.74
0.96	0.99	0.77	0.67	0.56	0.41	0.52	0.73	0.99	0.02
0.47	0.30	0.17	0.82	0.56	0.05	0.45	0.31	0.78	0.05
0.79	0.71	0.23	0.19	0.82	0.93	0.65	0.37	0.39	0.42
0.99	0.17	0.99	0.46	0.05	0.66	0.10	0.42	0.18	0.49
0.37	0.51	0.54	0.01	0.81	0.28	0.69	0.34	0.75	0.49
0.72	0.43	0.56	0.97	0.30	0.94	0.96	0.58	0.73	0.05
0.06	0.39	0.84	0.24	0.40	0.64	0.40	0.19	0.79	0.62
0.18	0.26	0.97	0.88	0.64	0.47	0.60	0.11	0.29	0.78

Different authors have offered considerations concerning the application of the χ^2 test. In the application to a data set the size of that in Example 7.7, the considerations do not apply—that is, if 100 values are in the

Table 7.3 Computations for Chi-Square Test

Interval	O_i	E_i	$O_i - E_i$	$(O_i - E_i)^2$	$\frac{(O_i - E_i)^2}{E_i}$
1	8	10	−2	4	0.4
2	8	10	−2	4	0.4
3	10	10	0	0	0.0
4	9	10	−1	1	0.1
5	12	10	2	4	0.4
6	8	10	−2	4	0.4
7	10	10	0	0	0.0
8	14	10	4	16	1.6
9	10	10	0	0	0.0
10	11	10	1	1	0.1
	100	100	0		3.4

sample and from 5 to 10 intervals of equal length are used, the test will be acceptable. In general, it is recommended that n and N be chosen so that each $E_i \geq 5$.

Both the Kolmogorov–Smirnov test and the chi-square test are acceptable for testing the uniformity of a sample of data, provided that the sample size is large. However, the Kolmogorov–Smirnov test is the more powerful of the two and is recommended. Furthermore, the Kolmogorov–Smirnov test can be applied to small sample sizes, whereas the chi-square is valid only for large samples, say $N \geq 50$.

Imagine a set of 100 numbers which are being tested for independence, one where the first 10 values are in the range 0.01–0.10, the second 10 values are in the range 0.11–0.20, and so on. This set of numbers would pass the frequency tests with ease, but the ordering of the numbers produced by the generator would not be random. The test in the next section of this chapter is concerned with the independence of random numbers that are generated.

7.4.2 Tests for Autocorrelation

The tests for autocorrelation are concerned with the dependence between numbers in a sequence. As an example, consider the following sequence of numbers:

0.12	0.01	0.23	0.28	0.89	0.31	0.64	0.28	0.83	0.93
0.99	0.15	0.33	0.35	0.91	0.41	0.60	0.27	0.75	0.88
0.68	0.49	0.05	0.43	0.95	0.58	0.19	0.36	0.69	0.87

From a visual inspection, these numbers appear random, and they would probably pass all the tests presented to this point. However, an examination of the 5th, 10th, 15th (every five numbers beginning with the fifth), and so on, indicates a very large number in that position. Now, 30 numbers is a rather small sample size on which to reject a random number generator, but the notion is that numbers in the sequence might be related. In this particular section, a method for discovering whether such a relationship exists is described. The relationship would not have to be all high numbers. It is possible to have all low numbers in the locations being examined, or the numbers could alternate from very high to very low.

The test to be described shortly requires the computation of the autocorrelation between every m numbers (m is also known as the lag), starting with the ith number. Thus, the autocorrelation ρ_{im} between the following numbers would be of interest: R_i, R_{i+m}, R_{i+2m}, ..., $R_{i+(M+1)m}$. The value M is the largest integer such

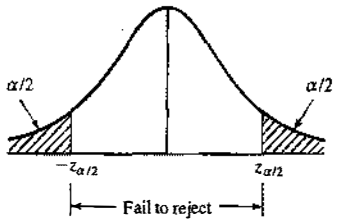


Figure 7.3 Failure to reject hypothesis.

that $i + (M + 1)m \le N$, where N is the total number of values in the sequence. (Thus, a subsequence of length $M + 2$ is being tested.)

A nonzero autocorrelation implies a lack of independence, so the following two-tailed test is appropriate:

$$H_0: \quad \rho_{im} = 0$$
$$H_1: \quad \rho_{im} \neq 0$$

For large values of M, the distribution of the estimator of ρ_{im}, denoted $\hat{\rho}_{im}$, is approximately normal if the values $R_i, R_{i+m}, R_{i+2m}, \ldots, R_{i+(M+1)m}$ are uncorrelated. Then the test statistic can be formed as follows:

$$Z_0 = \frac{\hat{\rho}_{im}}{\sigma_{\hat{\rho}_{im}}}$$

which is distributed normally with a mean of zero and a variance of 1, under the assumption of independence, for large M.

The formula for $\hat{\rho}_{im}$, in a slightly different form, and the standard deviation of the estimator, $\sigma_{\hat{\rho}_{im}}$, are given by Schmidt and Taylor [1970] as follows:

$$\hat{\rho}_{im} = \frac{1}{M+1}\left[\sum_{k=0}^{M} R_{i+km} R_{i+(k+1)m}\right] - 0.25$$

and

$$\sigma_{\hat{\rho}_{im}} = \frac{\sqrt{13M+7}}{12(M+1)}$$

After computing Z_0, do not reject the null hypothesis of independence if $-z_{\alpha/2} \le Z_0 \le z_{\alpha/2}$ where α is the level of significance and $z_{\alpha/2}$ is obtained from Table A.3. Figure 7.3 illustrates this test.

If $\rho_{im} > 0$, the subsequence is said to exhibit positive autocorrelation. In this case, successive values at lag m have a higher probability than expected of being close in value (i.e., high random numbers in the subsequence followed by high, and low followed by low). On the other hand, if $\rho_{im} < 0$, the subsequence is exhibiting negative autocorrelation, which means that low random numbers tend to be followed by high ones, and vice versa. The desired property, independence (which implies zero autocorrelation), means that there is no discernible relationship of the nature discussed here between successive random numbers at lag m.

Example 7.8

Test for whether the 3rd, 8th, 13th, and so on, numbers in the sequence at the beginning of this section are autocorrelated using $\alpha = 0.05$. Here, $i = 3$ (beginning with the third number), $m = 5$ (every five numbers), $N = 30$ (30 numbers in the sequence), and $M = 4$ (largest integer such that $3 + (M + 1)5 \le 30$). Then,

$$\hat{\rho}_{35} = \frac{1}{4+1}[(0.23)(0.28) + (0.28)(0.33) + (0.33)(0.27) + (0.27)(0.05) + (0.05)(0.36)] - 0.25$$
$$= -0.1945$$

and

$$\sigma_{\hat{\rho}_{35}} = \frac{\sqrt{13(4)+7}}{12(4+1)} = 0.1280$$

Then, the test statistic assumes the value

$$Z_0 = -\frac{0.1945}{0.1280} = -1.516$$

Now, the critical value from Table A.3 is

$$z_{0.025} = 1.96$$

Therefore, the hypothesis of independence cannot be rejected on the basis of this test.

It can be observed that this test is not very sensitive for small values of M, particularly when the numbers being tested are on the low side. Imagine what would happen if each of the entries in the foregoing computation of $\hat{\rho}_{im}$ were equal to zero. Then $\hat{\rho}_{im}$ would be equal to -0.25 and the calculated Z would have the value of -1.95, not quite enough to reject the hypothesis of independence.

There are many sequences that can be formed in a set of data, given a large value of N. For example, beginning with the first number in the sequence, possibilities include (1) the sequence of all numbers, (2) the sequence formed from the first, third, fifth, ..., numbers, (3) the sequence formed from the first, fourth, ..., numbers, and so on. If $\alpha = 0.05$, there is a probability of 0.05 of rejecting a true hypothesis. If 10 independent sequences are examined, the probability of finding no significant autocorrelation, by chance alone, is $(0.95)^{10}$ or 0.60. Thus, 40% of the time significant autocorrelation would be detected when it does not exist. If α is 0.10 and 10 tests are conducted, there is a 65% chance of finding autocorrelation by chance alone. In conclusion, in "fishing" for autocorrelation by performing numerous tests, autocorrelation might eventually be detected, perhaps by chance alone, even when there is no autocorrelation present.

7.5 SUMMARY

This chapter described the generation of random numbers and the subsequent testing of the generated numbers for uniformity and independence. Random numbers are used to generate random variates, the subject of Chapter 8.

Of the many types of random-number generators available, ones based on the linear congruential method are the most widely used, but they are being replaced by combined linear congruential generators. Of the many types of statistical tests that are used in testing random-number generators, two different types are described: one testing for uniformity, and one testing for independence.

The simulation analyst might never work directly with a random-number generator or with the testing of random numbers from a generator. Most computers and simulation languages have routines that generate a random number, or streams of random numbers, for the asking. But even generators that have been used for years, some of which are still in use, have been found to be inadequate. So this chapter calls the simulation analyst's attention to such possibilities, with a warning to investigate and confirm that the generator has been tested thoroughly. Some researchers have attained sophisticated expertise in developing methods for generating

and testing random numbers and the subsequent application of these methods. This chapter provides only a basic introduction to the subject matter; more depth and breadth are required for the reader to become a specialist in the area. The bible is Knuth [1998]; see also the reviews in Bratley, Fox, and Schrage [1996], Law and Kelton [2000], L'Ecuyer [1998], and Ripley [1987].

One final caution is due. Even if generated numbers pass all the tests (those covered in this chapter and those mentioned in the chapter), some underlying pattern might have gone undetected without the generator's having been rejected as faulty. However, the generators available in widely used simulation languages have been extensively tested and validated.

REFERENCES

BRATLEY, P., B. L. FOX, AND L. E. SCHRAGE [1996], *A Guide to Simulation*, 2d ed., Springer-Verlag, New York.

COX, D. R., AND P. A. W. LEWIS [1966], *The Statistical Analysis of Series of Events*, Methuen, London.

DURBIN, J. [1967], "Tests of Serial Independence Based on the Cumulated Periodogram," *Bulletin of the International Institute of Statistics*.

FISHMAN, G. S. [1978], *Principles of Discrete Event Simulation*, Wiley, New York.

GOOD, I. J. [1953], "The Serial Test for Sampling Numbers and Other Tests of Randomness," *Proceedings of the Cambridge Philosophical Society*, Vol. 49, pp. 276–284.

GOOD, I. J. [1967], "The Generalized Serial Test and the Binary Expansion of 4," *Journal of the Royal Statistical Society*, Ser. A, Vol. 30, No. 1, pp. 102–107.

KNUTH, D. W. [1998], *The Art of Computer Programming: Vol. 2, Semi-numerical Algorithms*, 2d ed., Addison–Wesley, Reading, MA.

LAW, A. M., AND W. D. KELTON [2000], *Simulation Modeling & Analysis*, 3d ed., McGraw–Hill, New York.

LEARMONTH, G. P., AND P. A. W. LEWIS [1973], "Statistical Tests of Some Widely Used and Recently Proposed Uniform Random Number Generators," *Proceedings of the Conference on Computer Science and Statistics: Seventh Annual Symposium on the Interface*, Western Publishing, North Hollywood, CA, pp. 163–171.

L'ECUYER, P. [1988], "Efficient and Portable Combined Random Number Generators," *Communications of the ACM*, Vol. 31, pp. 742–749, 774.

L'ECUYER, P. [1996], "Combined Multiple Recursive Random Number Generators," *Operations Research*, Vol. 44, pp. 816–822.

L'ECUYER, P. [1998], "Random Number Generation," Chapter 4 in *Handbook of Simulation*, Wiley, New York.

L'ECUYER, P. [1999], "Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators," *Operations Research*, Vol. 47, pp. 159–164.

L'ECUYER, P., R. SIMARD, E. J. CHEN, AND W. D. KELTON [2002], "An Object-Oriented Random-Number Package with Many Long Streams and Substreams," *Operations Research*, Vol. 50, pp. 1073–1075.

LEHMER, D. H. [1951], *Proceedings of the Second Symposium on Large-Scale Digital Computing Machinery*, Harvard University Press, Cambridge, MA.

LEWIS, P. A. W., A. S. GOODMAN, AND J. M. MILLER [1969], "A Pseudo-Random Number Generator for the System/360," *IBM Systems Journal*, Vol. 8, pp. 136–145.

RIPLEY, B. D. [1987], *Stochastic Simulation*, Wiley, New York.

SCHMIDT, J. W., And R. E. TAYLOR [1970], *Simulation and Analysis of Industrial Systems*, Irwin, Homewood, IL.

EXERCISES

1. Describe a procedure to physically generate random numbers on the interval [0, 1] with 2-digit accuracy. (*Hint*: Consider drawing something out of a hat.)
2. List applications, other than systems simulation, for pseudo-random numbers—for example, video gambling games.

3. How could random numbers that are uniform on the interval [0, 1] be transformed into random numbers that are uniform on the interval [−11, 17]? Transformations to more general distributions are described in Chapter 8.
4. Generate random numbers using multiplicative congruential method with $X_0 = 5$, $a = 17$, and $m = 64$.
5. Repeat Exercise 4 with $X_0 = 6$, 7, and 8.
6. Generate four-digit random numbers by linear congruential method with $X_0 = 21$, $a = 34$, and $c = 7$.
7. The sequence of numbers 0.54, 0.73, 0.98, 0.11, and 0.68 has been generated. Use the Kolmogorov–Smirnov test with $\alpha = 0.05$ to learn whether the hypothesis that the numbers are uniformly distributed on the interval [0, 1] can be rejected.
8. Generate 1000 random numbers between 0 and 99 using Excel. Conduct chi-square test with $a = 0.05$ and verify whether the numbers are uniformly distributed.
9. Figure out whether these linear congruential generators can achieve a maximum period; also, state restrictions on X_0 to obtain this period:
   (a) the mixed congruential method with
   $$a = 2,814,749,767,109$$
   $$c = 59,482,661,568,307$$
   $$m = 2^{48}$$
   (b) the multiplicative congruential generator with
   $$a = 69,069$$
   $$c = 0$$
   $$m = 2^{32}$$
   (c) the mixed congruential generator with
   $$a = 4951$$
   $$c = 247$$
   $$m = 256$$
   (d) the multiplicative congruential generator with
   $$a = 6507$$
   $$c = 0$$
   $$m = 1024$$
10. Use the mixed congruential method to generate a sequence of three two-digit random numbers with $X_0 = 37$ $a = 7$, $c = 29$, and $m = 100$.
11. Additive congruential method employs the following expression to generate random numbers:
    $$X_{n+1} = (X_1 + X_n) \bmod m$$
    where X_1 to X_n are the seeds and X_{n+1} is the new random number. Assuming $n = 5$, $X_1 = 20$, $X_2 = 82$, $X_3 = 42$, $X_4 = 76$, $X_5 = 59$, and $m = 100$, generate 10 new random numbers.
12. Write a computer program to generate random numbers using additive congruential method given in Exercise 11.

13. If $X_0 = 3579$ in Exercise 9(c), generate the first random number in the sequence. Compute the random number to four-place accuracy.

14. Investigate the random-number generator in a spreadsheet program on a computer to which you have access. In many spreadsheets, random numbers are generated by a function called RAND or @RAND.

(a) Check the user's manual to see whether it describes how the random numbers are generated.
(b) Write macros to conduct each of the tests described in this chapter. Generate 100 sets of random numbers, each set containing 100 random numbers. Perform each test on each set of random numbers. Draw conclusions.

15. Consider the multiplicative congruential generator under the following circumstances:

(a) $a = 11$, $m = 16$, $X_0 = 7$
(b) $a = 11$, $m = 16$, $X_0 = 8$
(c) $a = 7$, $m = 16$, $X_0 = 7$
(d) $a = 7$, $m = 16$, $X_0 = 8$

Generate enough values in each case to complete a cycle. What inferences can be drawn? Is maximum period achieved?

16. For 16-bit computers, L'Ecuyer [1988] recommends combining three multiplicative generators, with $m_1 = 32{,}363$, $a_1 = 157$, $m_2 = 31{,}727$, $a_2 = 146$, $m_3 = 31{,}657$, and $a_3 = 142$. The period of this generator is approximately 8×10^{12}. Generate 5 random numbers with the combined generator, using the initial seeds $X_{i,0} = 100, 300, 500$, for the individual generators $i = 1, 2, 3$.

17. Search the web and find various other methods of generating random numbers.

18. Use the principles described in this chapter to develop your own linear congruential random-number generator.

19. Use the principles described in this chapter to develop your own combined linear congruential random-number generator.

20. The following is the set of single-digit numbers from a random number generator.

6	7	0	6	9	9	0	6	4	6
4	0	8	2	6	6	1	2	6	8
5	6	0	4	7	1	3	5	0	7
1	4	9	8	6	0	9	6	6	7
1	0	4	7	9	2	0	1	4	8
6	9	7	7	5	4	2	3	3	3
6	0	5	8	2	5	8	8	3	1
4	0	8	1	7	0	0	6	2	8
5	6	0	8	0	6	9	7	0	0
3	1	5	4	3	8	3	3	2	4

Using appropriate test, check whether the numbers are uniformly distributed.

21. In some applications, it is useful to be able to quickly skip ahead in a pseudo-random number sequence without actually generating all of the intermediate values. (a) For a linear congruential generator with $c = 0$, show that $X_{i+n} = (a^n X_i) \bmod m$. (b) Next, show that $(a^n X_i) \bmod m = (a^n \bmod m) X_i \bmod m$ (this result is useful because $a^n \bmod m$ can be precomputed, making it easy to skip ahead n random numbers from any point in the sequence). (c) In Example 7.3, use this result to compute X_5, starting with $X_0 = 63$. Check your answer by computing X_5 in the usual way.

8

Random-Variate Generation

This chapter deals with procedures for sampling from a variety of widely-used continuous and discrete distributions. Previous discussions and examples indicated the usefulness of statistical distributions in modeling activities that are generally unpredictable or uncertain. For example, interarrival times and service times at queues and demands for a product are quite often unpredictable in nature, at least to a certain extent. Usually, such variables are modeled as random variables with some specified statistical distribution, and standard statistical procedures exist for estimating the parameters of the hypothesized distribution and for testing the validity of the assumed statistical model. Such procedures are discussed in Chapter 9.

In this chapter, it is assumed that a distribution has been completely specified, and ways are sought to generate samples from this distribution to be used as input to a simulation model. The purpose of the chapter is to explain and illustrate some widely-used techniques for generating random variates, not to give a state-of-the-art survey of the most efficient techniques. In practice, most simulation modelers will use either existing routines available in programming libraries or the routines built into the simulation language being used. However, some programming languages do not have built-in routines for all of the regularly used distributions, and some computer installations do not have random-variate-generation libraries; in such cases the modeler must construct an acceptable routine. Even though the chance of this happening is small, it is nevertheless worthwhile to understand how random-variate generation occurs.

This chapter discusses the inverse-transform technique and, more briefly, the acceptance–rejection technique and special properties. Another technique, the composition method, is discussed by Devroye [1986], Dagpunar [1988], Fishman [1978], and Law and Kelton [2000]. All the techniques in this chapter assume that a source of uniform [0,1] random numbers, $R_1, R_2, \ldots$ is readily available, where each R_i has pdf

$$f_R(x) = \begin{cases} 1, & 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$$

and cdf

$$F_R(x)=\begin{cases}0, & x<0\\ x, & 0\le x\le 1\\ 1, & x>1\end{cases}$$

Throughout this chapter R and $R_1, R_2, \ldots$ represent random numbers uniformly distributed on [0,1] and generated by one of the techniques in Chapter 7 or taken from a random number table, such as Table A.1.

8.1 INVERSE-TRANSFORM TECHNIQUE

The inverse-transform technique can be used to sample from the exponential, the uniform, the Weibull, the triangular distributions and from empirical distributions. Additionally, it is the underlying principle for sampling from a wide variety of discrete distributions. The technique will be explained in detail for the exponential distribution and then applied to other distributions. Computationally, it is the most straightforward, but not always the most efficient, technique.

8.1.1 Exponential Distribution

The exponential distribution, discussed in Section 5.4, has the probability density function (pdf)

$$f(x)=\begin{cases}\lambda e^{-\lambda x}, & x\ge 0\\ 0, & x<0\end{cases}$$

and the cumulative distribution function (cdf)

$$F(x)=\int_{-\infty}^{x} f(t)dt=\begin{cases}1-e^{-\lambda x}, & x\ge 0\\ 0, & x<0\end{cases}$$

The parameter λ can be interpreted as the mean number of occurrences per time unit. For example, if interarrival times $X_1, X_2, X_3, \ldots$ had an exponential distribution with rate λ, then λ could be interpreted as the mean number of arrivals per time unit, or the arrival rate. Notice that, for any i,

$$E(X_i)=\frac{1}{\lambda}$$

and so $1/\lambda$ is the mean interarrival time. The goal here is to develop a procedure for generating values $X_1, X_2, X_3, \ldots$ that have an exponential distribution.

The inverse-transform technique can be utilized, at least in principle, for any distribution, but it is most useful when the cdf, $F(x)$, is of a form so simple that its inverse, F^{-1}, can be computed easily.[1] One step-by-step procedure for the inverse-transform technique, illustrated by the exponential distribution, consists of the following steps:

Step 1. Compute the cdf of the desired random variable X.

For the exponential distribution, the cdf is $F(x) = 1-e^{-\lambda x}, x \ge 0$.

Step 2. Set $F(X) = R$ on the range of X.

For the exponential distribution, it becomes $1-e^{-\lambda X} = R$ on the range $x \ge 0$.

[1]The notation F^{-1} denotes the solution of the equation $r = F(x)$ in terms of r; it does not denote $1/F$.

X is a random variable (with the exponential distribution in this case), so $1-e^{-\lambda X}$ is also a random variable, here called R. As will be shown later, R has a uniform distribution over the interval [0, 1].

Step 3. Solve the equation $F(X) = R$ for X in terms of R.

For the exponential distribution, the solution proceeds as follows:

$$\begin{aligned}1-e^{-\lambda X}&=R\\ e^{-\lambda X}&=1-R\\ -\lambda X&=\ln(1-R)\\ X&=-\frac{1}{\lambda}\ln(1-R)\end{aligned}\qquad(8.1)$$

Equation (8.1) is called a random-variate generator for the exponential distribution. In general, Equation (8.1) is written as $X = F^{-1}(R)$. Generating a sequence of values is accomplished through Step 4.

Step 4. Generate (as needed) uniform random numbers $R_1, R_2, R_3, \ldots$ and compute the desired random variates by

$$X_i = F^{-1}(R_i)$$

For the exponential case, $F^{-1}(R) = (-1/\lambda)\ln(1-R)$ by Equation (8.1), so

$$X_i=-\frac{1}{\lambda}\ln(1-R_i)\qquad(8.2)$$

for $i = 1, 2, 3, \ldots$. One simplification that is usually employed in Equation (8.2) is to replace $1-R_i$ by R_i to yield

$$X_i=-\frac{1}{\lambda}\ln R_i\qquad(8.3)$$

This alternative is justified by the fact that both R_i and $1-R_i$ are uniformly distributed on [0,1].

Example 8.1

Table 8.1 gives a sequence of random numbers from Table A.1 and the computed exponential variates, X_i, given by Equation (8.2) with the value $\lambda = 1$. Figure 8.1(a) is a histogram of 200 values, $R_1, R_2, \ldots, R_{200}$ from the uniform distribution, and Figure 8.1(b) is a histogram of the 200 values, $X_1, X_2, \ldots, X_{200}$, computed by Equation (8.2). Compare these empirical histograms with the theoretical density functions in Figure 8.1(c) and (d). As illustrated here, a histogram is an estimate of the underlying density function. (This fact is used in Chapter 9 as a way to identify distributions.)

Table 8.1 Generation of Exponential Variates X_i with Mean 1, given Random Numbers R_i

i	1	2	3	4	5
R_i	0.1306	0.0422	0.6597	0.7965	0.7696
X_i	0.1400	0.0431	1.078	1.592	1.468

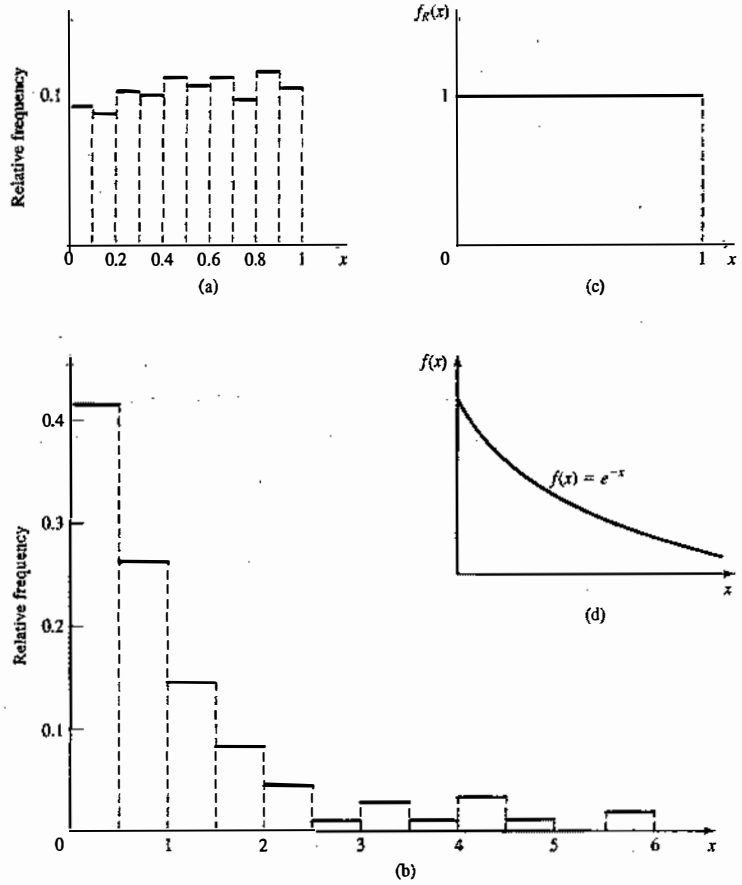


Figure 8.1 (a) Empirical histogram of 200 uniform random numbers; (b) empirical histogram of 200 exponential variates; (c) theoretical uniform density on [0, 1]; (d) theoretical exponential density with mean 1.

Figure 8.2 gives a graphical interpretation of the inverse-transform technique. The cdf shown is $F(x) = 1-e^{-x}$, an exponential distribution with rate $\lambda = 1$. To generate a value X_1 with cdf $F(x)$, a random number R_1 between 0 and 1 is generated, then a horizontal line is drawn from R_1 to the graph of the cdf, then a vertical line is dropped to the x axis to obtain X_1, the desired result. Notice the inverse relation between R_1 and X_1, namely

$$R_1 = 1 - e^{-X_1}$$

and

$$X_1 = -\ln(1 - R_1)$$

In general, the relation is written as

$$R_1 = F(X_1)$$

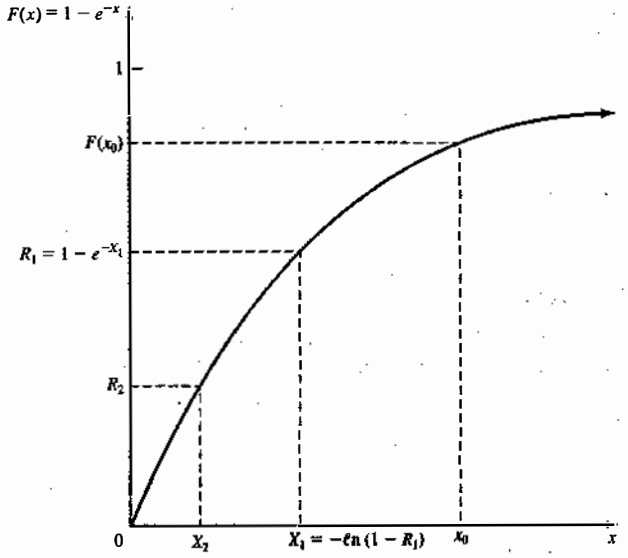


Figure 8.2 Graphical view of the inverse-transform technique.

and

$$X_1 = F^{-1}(R_1)$$

Why does the random variable X_1 generated by this procedure have the desired distribution? Pick a value x_0 and compute the cumulative probability

$$P(X_1 \le x_0) = P(R_1 \le F(x_0)) = F(x_0) \tag{8.4}$$

To see the first equality in Equation (8.4), refer to Figure 8.2, where the fixed numbers x_0 and $F(x_0)$ are drawn on their respective axes. It can be seen that $X_1 \le x_0$ when and only when $R_1 \le F(x_0)$. Since $0 \le F(x_0) \le 1$, the second equality in Equation (8.4) follows immediately from the fact that R_1 is uniformly distributed on [0,1]. Equation (8.4) shows that the cdf of X_1 is F; hence, X_1 has the desired distribution.

8.1.2 Uniform Distribution

Consider a random variable X that is uniformly distributed on the interval $[a, b]$. A reasonable guess for generating X is given by

$$X = a + (b - a)R \tag{8.5}$$

[Recall that R is always a random number on [0, 1].] The pdf of X is given by

$$f(x) = \begin{cases} \dfrac{1}{b-a}, & a \le x \le b \\ 0, & \text{otherwise} \end{cases}$$

The derivation of Equation (8.5) follows Steps 1 through 3 of Section 8.1.1:

Step 1. The cdf is given by

$$F(x)=\begin{cases}0, & x<a\\ \dfrac{x-a}{b-a}, & a\le x\le b\\ 1, & x>b\end{cases}$$

Step 2. Set $F(X)=(X-a)/(b-a)=R$.

Step 3. Solving for X in terms of R yields $X=a+(b-a)R$, which agrees with Equation (8.5).

8.1.3 Weibull Distribution

The Weibull distribution was introduced in Section 5.4 as a model for *time to failure* for machines or electronic components. When the location parameter ν is set to 0, its pdf is given by Equation (5.47):

$$f(x)=\begin{cases}\dfrac{\beta}{\alpha^{\beta}}x^{\beta-1}e^{-(x/\alpha)^{\beta}}, & x\ge 0\\ 0, & \text{otherwise}\end{cases}$$

where $\alpha>0$ and $\beta>0$ are the scale and shape parameters of the distribution. To generate a Weibull variate, follow Steps 1 through 3 of Section 8.1.1:

Step 1. The cdf is given by $F(X)=1-e^{-(x/\alpha)^{\beta}}$, $x\ge 0$.

Step 2. Set $F(X)=1-e^{-(X/\alpha)^{\beta}}=R$.

Step 3. Solving for X in terms of R yields

$$X=\alpha[-\ln(1-R)]^{1/\beta} \tag{8.6}$$

The derivation of Equation (8.6) is left as Exercise 10 for the reader. By comparing Equations (8.6) and (8.1), it can be seen that, if X is a Weibull variate, then X^{β} is an exponential variate with mean α^{β}. Conversely, if Y is an exponential variate with mean μ, then $Y^{1/\beta}$ is a Weibull variate with shape parameter β and scale parameter $\alpha=\mu^{1/\beta}$.

8.1.4 Triangular Distribution

Consider a random variable X that has pdf

$$f(x)=\begin{cases}x, & 0\le x\le 1\\ 2-x, & 1<x\le 2\\ 0, & \text{otherwise}\end{cases}$$

as shown in Figure 8.3. This distribution is called a triangular distribution with endpoints (0, 2) and mode at 1. Its cdf is given by

$$F(x)=\begin{cases}0, & x\le 0\\ \dfrac{x^2}{2}, & 0<x\le 1\\ 1-\dfrac{(2-x)^2}{2}, & 1<x\le 2\\ 1, & x>2\end{cases}$$

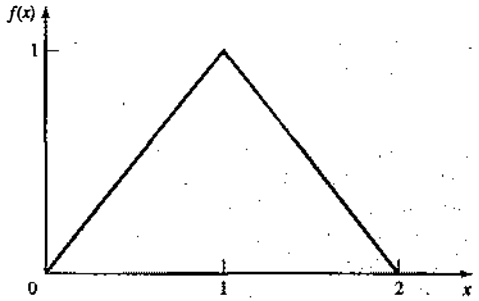


Figure 8.3 Density function for a particular triangular distribution.

For $0\le X\le 1$,

$$R=\frac{X^2}{2} \tag{8.7}$$

and for $1\le X\le 2$,

$$R=1-\frac{(2-X)^2}{2} \tag{8.8}$$

By Equation (8.7), $0\le X\le 1$ implies that $0\le R\le \frac{1}{2}$, in which case $X=\sqrt{2R}$. By Equation (8.8), $1\le X\le 2$ implies that $\frac{1}{2}\le R\le 1$, in which case $X=2-\sqrt{2(1-R)}$. Thus, X is generated by

$$X=\begin{cases}\sqrt{2R}, & 0\le R\le \frac{1}{2}\\ 2-\sqrt{2(1-R)}, & \frac{1}{2}<R\le 1\end{cases} \tag{8.9}$$

Exercises 2, 3, and 4 give the student practice in dealing with other triangular distributions. Notice that, if the pdf and cdf of the random variable X come in parts (i.e., require different formulas over different parts of the range of X), then the application of the inverse-transform technique for generating X will result in separate formulas over different parts of the range of R, as in Equation (8.9). A general form of the triangular distribution was discussed in Section 5.4.

8.1.5 Empirical Continuous Distributions

If the modeler has been unable to find a theoretical distribution that provides a good model for the input data, then it may be necessary to use the empirical distribution of the data. One possibility is to simply resample the observed data itself. This is known as using the *empirical distribution*, and it makes particularly good sense when the input process is known to take on a finite number of values. See Section 8.1.7 for an example of this type of situation and for a method for generating random inputs.

On the other hand, if the data are drawn from what is believed to be a continuous-valued input process, then it makes sense to interpolate between the observed data points to fill in the gaps. This section describes a method for defining and generating data from a continuous empirical distribution.

Example 8.2

Five observations of fire-crew response times (in minutes) to incoming alarms have been collected to be used in a simulation investigating possible alternative staffing and crew-scheduling policies. The data are

2.76 1.83 0.80 1.45 1.24

Before collecting more data, it is desired to develop a preliminary simulation model that uses a response-time distribution based on these five observations. Thus, a method for generating random variates from the response-time distribution is needed. Initially, it will be assumed that response times X have a range $0 \le X \le c$, where c is unknown, but will be estimated by $\hat{c} = \max\{X_i : i = 1, \ldots, n\} = 2.76$, where $\{X_i, i = 1, \ldots, n\}$ are the raw data and $n = 5$ is the number of observations.

Arrange the data from smallest to largest and let $x_{(1)} \le x_{(2)} \le \cdots \le x_{(n)}$ denote these sorted values. The smallest possible value is believed to be 0, so define $x_{(0)} = 0$. Assign the probability $1/n = 1/5$ to each interval

Table 8.2 Summary of Fire-Crew Response-Time Data

i	*Interval* $x_{(i-1)} < x \le x_{(i)}$	*Probability* $1/n$	*Cumulative Probability,* i/n	*Slope* a_i
1	$0.0 < x \le 0.80$	0.2	0.2	4.00
2	$0.80 < x \le 1.24$	0.2	0.4	2.20
3	$1.24 < x \le 1.45$	0.2	0.6	1.05
4	$1.45 < x \le 1.83$	0.2	0.8	1.90
5	$1.83 < x \le 2.76$	0.2	1.0	4.65

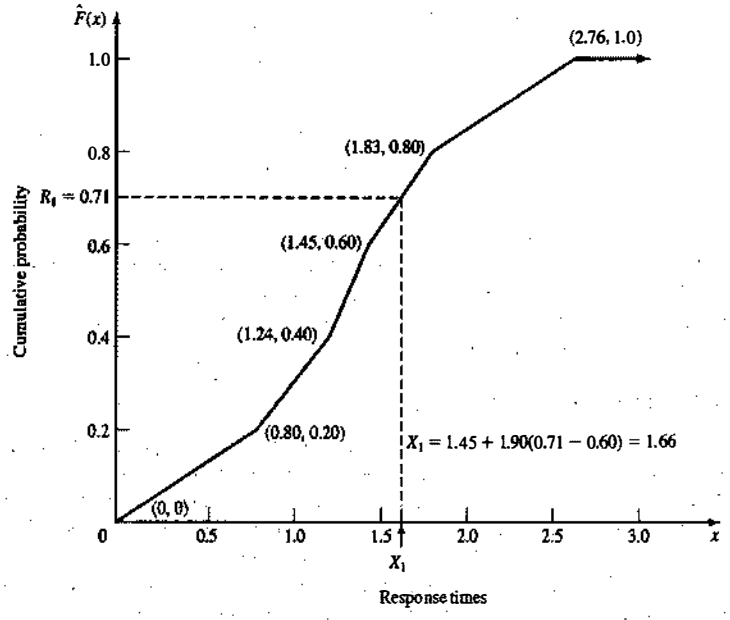


Figure 8.4 Empirical cdf of fire-crew response times.

$x_{(i-1)} < x \le x_{(i)}$, as shown in Table 8.2. The resulting empirical cdf, $\hat{F}(x)$, is illustrated in Figure 8.4. The slope of the ith line segment is given by

$$a_i = \frac{x_{(i)} - x_{(i-1)}}{i/n - (i-1)/n} = \frac{x_{(i)} - x_{(i-1)}}{1/n}$$

The inverse cdf is calculated by

$$X = \hat{F}^{-1}(R) = x_{(i-1)} + a_i\left(R - \frac{(i-1)}{n}\right) \tag{8.10}$$

when $(i-1)/n < R \le i/n$.

For example, if a random number $R_1 = 0.71$ is generated, then R_1 is seen to lie in the fourth interval (between $3/5 = 0.60$ and $4/5 = 0.80$); so, by Equation (8.10),

$$\begin{aligned} X_1 &= x_{(4-1)} + a_4(R_1 - (4-1)/n) \\ &= 1.45 + 1.90(0.71 - 0.60) \\ &= 1.66 \end{aligned}$$

The reader is referred to Figure 8.4 for a graphical view of the generation procedure.

In Example 8.2, each data point was represented in the empirical cdf. If a large sample of data is available (and sample sizes from several hundred to tens of thousands are possible with modern, automated data collection), then it might be more convenient (and computationally efficient) to first summarize the data into a frequency distribution with a much smaller number of intervals and then fit a continuous empirical cdf to the frequency distribution. Only a slight generalization of Equation (8.10) is required to accomplish this. Now the slope of the ith line segment is given by:

$$a_i = \frac{x_{(i)} - x_{(i-1)}}{c_i - c_{i-1}}$$

where c_i is the cumulative probability of the first i intervals of the frequency distribution and $x_{(i-1)} < x \le x_{(i)}$ is the ith interval. The inverse cdf is calculated as

$$X = \hat{F}^{-1}(R) = x_{(i-1)} + a_i\left(R - c_{i-1}\right) \tag{8.11}$$

when $c_{i-1} < R \le c_i$.

Example 8.3

Suppose that 100 broken-widget repair times have been collected. The data are summarized in Table 8.3 in terms of the number of observations in various intervals. For example, there were 31 observations between 0 and 0.5 hour, 10 between 0.5 and 1 hour, and so on. Suppose it is known that all repairs take at least 15 minutes, so that $X \ge 0.25$ hour always. Then we set $x_{(0)} = 0.25$, as shown in Table 8.3 and Figure 8.5.

For example, suppose the first random number generated is $R_1 = 0.83$. Then, since R_1 is between $c_3 = 0.66$ and $c_4 = 1.00$,

$$X_1 = x_{(4-1)} + a_4(R_1 - c_{4-1}) = 1.5 + 1.47(0.83 - 0.66) = 1.75 \tag{8.12}$$

Table 8.3 Summary of Repair-Time Data

i	*Interval (Hours)*	*Frequency*	*Relative Frequency*	*Cumulative Frequency, c_i*	*Slope a_i*
1	$0.25 \le x \le 0.5$	31	0.31	0.31	0.81
2	$0.5 < x \le 1.0$	10	0.10	0.41	5.0
3	$1.0 < x \le 1.5$	25	0.25	0.66	2.0
4	$1.5 < x \le 2.0$	34	0.34	1.00	1.47

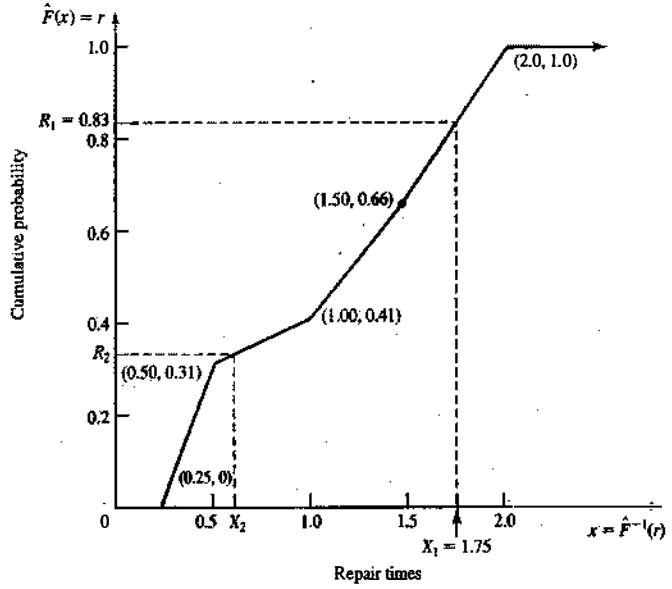


Figure 8.5 Generating variates from the empirical distribution function for repair-time data ($X \ge 0.25$).

As another illustration, suppose that $R_2 = 0.33$. Since $c_1 = 0.31 < R_2 \le 0.41 = c_2$,

$$\begin{aligned} X_2 &= x_{(1)} + a_2(R_2 - c_1) \\ &= 0.5 + 5.0(0.33 - 0.31) \\ &= 0.6 \end{aligned}$$

The point ($R_2 = 0.33$, $X_2 = 0.6$) is also shown in Figure 8.5.

Now reconsider the data of Table 8.3. The data are restricted in the range $0.25 \le X \le 2.0$, but the underlying distribution might have a wider range. This provides one important reason for attempting to find a theoretical statistical distribution (such as the gamma or Weibull) for the data: that these distributions do allow a wider range—namely, $0 \le X < \infty$. On the other hand, an empirical distribution adheres closely to what is present in the data itself, and the data are often the best source of information available.

When data are summarized in terms of frequency intervals, it is recommended that relatively short intervals be used, for doing so results in a more accurate portrayal of the underlying cdf. For example, for the repair-time data of Table 8.3, for which there were $n = 100$ observations, a much more accurate estimate could have been obtained by using 10 to 20 intervals, certainly not an excessive number, rather than the four fairly wide intervals actually used here for purposes of illustration.

Several comments are in order:

1. A computerized version of the procedure will become more inefficient as the number of intervals, n, increases. A systematic computerized version is often called a table-lookup generation scheme, because, given a value of R, the computer program must search an array of c_i values to find the interval i in which R lies, namely the interval i satisfying
$$c_{i-1} < R \le c_i$$
The more intervals there are, the longer on the average the search will take if it is implemented in the crude way described here. The analyst should consider this trade-off between accuracy of the estimating cdf and computational efficiency when programming the procedure. If a large number of observations are available, the analyst may well decide to group the observations into from 20 to 50 intervals (say) and then use the procedure of Example 8.3—or a more efficient table-lookup procedure could be used, such as the one described in Law and Kelton [2000].
2. In Example 8.2, it was assumed that response times X satisfied $0 \le X \le 2.76$. This assumption led to the inclusion of the points $x_{(0)} = 0$ and $x_{(5)} = 2.76$ in Figure 8.4 and Table 8.2. If it is known a priori that X falls in some other range, for example, if it is known that response times are always between 15 seconds and 3 minutes—that is,
$$0.25 \le X \le 3.0$$
—then the points $x_{(0)} = 0.25$ and $x_{(6)} = 3.0$ would be used to estimate the empirical cdf of response times. Notice that, because of inclusion of the new point $x_{(6)}$, there are now six intervals instead of five, and each interval is assigned probability $1/6 = 0.167$.

8.1.6 Continuous Distributions without a Closed-Form Inverse

A number of useful continuous distributions do not have a closed form expression for their cdf or its inverse; examples include the normal, gamma, and beta distributions. For this reason, it is often stated that the inverse-transform technique for random-variate generation is not available for these distributions. It can, in effect, become available if we are willing to *approximate* the inverse cdf, or numerically integrate and search the cdf. Although this approach sounds inaccurate, notice that even a closed-form inverse requires approximation in order to evaluate it on a computer. For example, generating exponentially distributed random variates via the inverse cdf $X = F^{-1}(R) = -\ln(1 - R)/\lambda$ requires a numerical approximation for the logarithm function. Thus, there is no essential difference between using an approximate inverse cdf and approximately evaluating a closed-form inverse. The problem with using approximate inverse cdfs is that some of them are computationally slow to evaluate.

To illustrate the idea, consider a simple approximation to the inverse cdf of the standard normal distribution, proposed by Schmeiser [1979]:

$$X = F^{-1}(R) \approx \frac{R^{0.135} - (1 - R)^{0.135}}{0.1975}$$

This approximation gives at least one-decimal-place accuracy for $0.0013499 \le R \le 0.9986501$. Table 8.4 compares the approximation with exact values (to four decimal places) obtained by numerical integration for several values of R. Much more accurate approximations exist that are only slightly more complicated. A good source of these approximations for a number of distributions is Bratley, Fox, and Schrage [1996].

Table 8.4 Comparison of Approximate Inverse with Exact Values (To Four Decimal Places) for the Standard Normal Distribution

R	*Approximate Inverse*	*Exact Inverse*
0.01	−2.3263	−2.3373
0.10	−1.2816	−1.2813
0.25	−0.6745	−0.6713
0.50	0.0000	0.0000
0.75	0.6745	0.6713
0.90	1.2816	1.2813
0.99	2.3263	2.3373

8.1.7 Discrete Distributions

All discrete distributions can be generated via the inverse-transform technique, either numerically through a table-lookup procedure or, in some cases, algebraically, the final generation scheme being in terms of a formula. Other techniques are sometimes used for certain distributions, such as the convolution technique for the binomial distribution. Some of these methods are discussed in later sections. This subsection gives examples covering both empirical distributions and two of the standard discrete distributions, the (discrete) uniform and the geometric. Highly efficient table-lookup procedures for these and other distributions are found in Bratley, Fox, and Schrage [1996] and in Ripley [1987].

Example 8.4: An Empirical Discrete Distribution

At the end of any day, the number of shipments on the loading dock of the IHW Company (whose main product is the famous "incredibly huge widget") is either 0, 1, or 2, with observed relative frequency of occurrence of 0.50, 0.30, and 0.20, respectively. Internal consultants have been asked to develop a model to improve the efficiency of the loading and hauling operations; as part of this model, they will need to be able to generate values, X, to represent the number of shipments on the loading dock at the end of each day. The consultants decide to model X as a discrete random variable with the distribution given in Table 8.5 and shown in Figure 8.6.

The probability mass function (pmf), $p(x)$, is given by

$$p(0) = P(X = 0) = 0.50$$
$$p(1) = P(X = 1) = 0.30$$
$$p(2) = P(X = 2) = 0.20$$

Table 8.5 Distribution of Number of Shipments, X

x	$p(x)$	$F(x)$
0	0.50	0.50
1	0.30	0.80
2	0.20	1.00

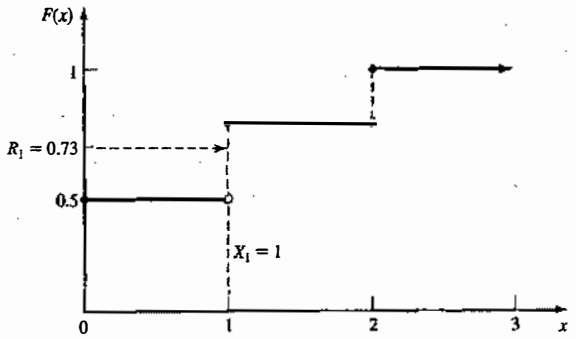


Figure 8.6 Cdf of number of shipments, X.

and the cdf, $F(x) = P(X \le x)$, is given by

$$F(x) = \begin{cases} 0 & x < 0 \\ 0.5 & 0 \le x < 1 \\ 0.8 & 1 \le x < 2 \\ 1.0 & 2 \le x \end{cases}$$

Recall that the cdf of a discrete random variable always consists of horizontal line segments with jumps of size $p(x)$ at those points, x, that the random variable can assume. For example, in Figure 8.6, there is a jump of size $p(0) = 0.5$ at $x = 0$, of size $p(1) = 0.3$ at $x = 1$, and of size $p(2) = 0.2$ at $x = 2$.

For generating discrete random variables, the inverse transform technique becomes a table-lookup procedure, but, unlike in the case of continuous variables, interpolation is not required. To illustrate the procedure, suppose that $R_1 = 0.73$ is generated. Graphically, as illustrated in Figure 8.6, first locate $R_1 = 0.73$ on the vertical axis, next draw a horizontal line segment until it hits a "jump" in the cdf, and then drop a perpendicular to the horizontal axis to get the generated variate. Here $R_1 = 0.73$ is transformed to $X_1 = 1$. This procedure is analogous to the procedure used for empirical continuous distributions in Section 8.1.5 and illustrated in Figure 8.4, except that the final step, linear interpolation, is eliminated.

The table-lookup procedure is facilitated by construction of a table such as Table 8.6. When $R_1 = 0.73$ is generated, first find the interval in which R_1 lies. In general, for $R = R_1$, if

$$F(x_{i-1}) = r_{i-1} < R \le r_i = F(x_i) \qquad (8.13)$$

then set $X_1 = x_i$. Here, $r_0 = 0$; $x_0 = -\infty$; $x_1, x_2, \ldots, x_n$ are the possible values of the random variable; and $r_k = p(x_1) + \cdots + p(x_k)$, $k = 1, 2, \ldots, n$. For this example, $n = 3$, $x_1 = 0$, $x_2 = 1$, and $x_3 = 2$; hence, $r_1 = 0.5$, $r_2 = 0.8$, and $r_3 = 1.0$. (Notice that $r_n = 1.0$ in all cases.)

Table 8.6 Table for Generating the Discrete Variate X

i	*Input,* r_i	*Output,* x_i
1	0.50	0
2	0.80	1
3	1.00	2

Since $r_1 = 0.5 < R_1 = 0.73 \le r_2 = 0.8$, set $X_1 = x_2 = 1$. The generation scheme is summarized as follows:

$$X = \begin{cases} 0, & R \le 0.5 \\ 1, & 0.5 < R \le 0.8 \\ 2, & 0.8 < R \le 1.0 \end{cases}$$

Example 8.4 illustrated the table-lookup procedure; the next example illustrates an algebraic approach that can be used for certain distributions.

Example 8.5: A Discrete Uniform Distribution

Consider the discrete uniform distribution on $\{1, 2, \ldots, k\}$ with pmf and cdf given by

$$p(x) = \frac{1}{k}, \quad x = 1, 2, \ldots, k$$

and

$$F(x) = \begin{cases} 0, & x < 1 \\ \frac{1}{k}, & 1 \le x < 2 \\ \frac{2}{k}, & 2 \le x < 3 \\ \vdots & \vdots \\ \frac{k-1}{k}, & k-1 \le x < k \\ 1, & k \le x \end{cases}$$

Let $x_i = i$ and $r_i = p(1) + \cdots + p(x_i) = F(x_i) = i/k$ for $i = 1, 2, \ldots, k$. Then, from Inequality (8.13), it can be seen that, if the generated random number R satisfies

$$r_{i-1} = \frac{i-1}{k} < R \le r_i = \frac{i}{k} \tag{8.14}$$

then X is generated by setting $X = i$. Now Inequality (8.14) can be solved for i:

$$\begin{aligned} i - 1 < Rk &\le i \\ Rk \le i &< Rk + 1 \end{aligned} \tag{8.15}$$

Let $\lceil y \rceil$ denote the smallest integer $\ge y$. For example, $\lceil 7.82 \rceil = 8$, $\lceil 5.13 \rceil = 6$, and $\lceil -1.32 \rceil = -1$. For $y \ge 0$, $\lceil y \rceil$ is a function that rounds up. This notation and Inequality (8.15) yield a formula for generating X, namely

$$X = \lceil Rk \rceil \tag{8.16}$$

For example, consider the generating of a random variate X that is uniformly distributed on $\{1, 2, \ldots, 10\}$. The variate, X, might represent the number of pallets to be loaded onto a truck. Using Table A.1 as a source of random numbers R and using Equation (8.16) with $k = 10$ yields

$$\begin{aligned} R_1 &= 0.78 \quad X_1 = \lceil 7.8 \rceil = 8 \\ R_2 &= 0.03 \quad X_2 = \lceil 0.3 \rceil = 1 \\ R_3 &= 0.23 \quad X_3 = \lceil 2.3 \rceil = 3 \\ R_4 &= 0.97 \quad X_4 = \lceil 9.7 \rceil = 10 \end{aligned}$$

The procedure discussed here can be modified to generate a discrete uniform random variate with any range consisting of consecutive integers. Exercise 13 asks the student to devise a procedure for one such case.

Example 8.6: The Geometric Distribution

Consider the geometric distribution with pmf

$$p(x) = p(1-p)^x, \quad x = 0, 1, 2, \ldots$$

where $0 < p < 1$. Its cdf is given by

$$\begin{aligned} F(x) &= \sum_{j=0}^{x} p(1-p)^j \\ &= \frac{p\{1-(1-p)^{x+1}\}}{1-(1-p)} \\ &= 1-(1-p)^{x+1} \end{aligned}$$

for $x = 0, 1, 2, \ldots$ Using the inverse-transform technique [i.e., Inequality (8.13)], recall that a geometric random variable X will assume the value x whenever

$$F(x-1) = 1 - (1-p)^x < R \le 1 - (1-p)^{x+1} = F(x) \tag{8.17}$$

where R is a generated random number assumed $0 < R < 1$. Solving Inequality (8.17) for x proceeds as follows:

$$\begin{aligned} (1-p)^{x+1} \le 1 - R &< (1-p)^x \\ (x+1)\ln(1-p) \le \ln(1-R) &< x\ln(1-p) \end{aligned}$$

But $1 - p < 1$ implies that $\ln(1-p) < 0$, so that

$$\frac{\ln(1-R)}{\ln(1-p)} - 1 \le x < \frac{\ln(1-R)}{\ln(1-p)} \tag{8.18}$$

Thus, $X = x$ for that integer value of x satisfying Inequality (8.18). In brief, and using the round-up function $\lceil \cdot \rceil$, we have

$$X = \left\lceil \frac{\ln(1-R)}{\ln(1-p)} - 1 \right\rceil \tag{8.19}$$

Since p is a fixed parameter, let $\beta = -1/\ln(1-p)$. Then $\beta > 0$ and, by Equation (8.19), $X = \lceil -\beta \ln(1-R) - 1 \rceil$. By Equation (8.1), $-\beta \ln(1-R)$ is an exponentially distributed random variable with mean β; so one way of generating a geometric variate with parameter p is to generate (by any method) an exponential variate with parameter $\beta^{-1} = -\ln(1-p)$, subtract one, and round up.

Occasionally, there is needed a geometric variate X that can assume values $\{q, q+1, q+2, \ldots\}$ with pmf $p(x) = p(1-p)^{x-q}(x = q, q+1, \ldots)$. Such a variate X can be generated, via Equation (8.19), by

$$X = q + \left\lceil \frac{\ln(1-R)}{\ln(1-p)} - 1 \right\rceil \tag{8.20}$$

One of the most common cases is $q = 1$.

Example 8.7

Generate three values from a geometric distribution on the range $\{X \geq 1\}$ with mean 2. Such a geometric distribution has pmf $p(x) = p(1-p)^{x-1}(x = 1, 2, \ldots)$ with mean $1/p = 2$, or $p = 1/2$. Thus, X can be generated by Equation (8.20) with $q = 1$, $p = 1/2$, and $1/\ln(1-p) = -1.443$. Using Table A.1, $R_1 = 0.932$, $R_2 = 0.105$, and $R_3 = 0.687$ yields

$$\begin{aligned} X_1 &= 1 + \lceil -1.443 \ln(1-0.932) - 1 \rceil \\ &= 1 + \lceil 3.878 - 1 \rceil = 4 \\ X_2 &= 1 + \lceil -1.443 \ln(1-0.105) - 1 \rceil = 1 \\ X_3 &= 1 + \lceil -1.443 \ln(1-0.687) - 1 \rceil = 2 \end{aligned}$$

Exercise 15 deals with an application of the geometric distribution.

8.2 ACCEPTANCE–REJECTION TECHNIQUE

Suppose that an analyst needed to devise a method for generating random variates, X, uniformly distributed between 1/4 and 1. One way to proceed would be to follow these steps:

Step 1. Generate a random number R.

Step 2a. If $R \geq 1/4$, accept $X = R$, then go to Step 3.

Step 2b. If $R < 1/4$, reject R, and return to Step 1.

Step 3. If another uniform random variate on [1/4, 1] is needed, repeat the procedure beginning at Step 1. If not, stop.

Each time Step 1 is executed, a new random number R must be generated. Step 2a is an "acceptance" and Step 2b is a "rejection" in this acceptance–rejection technique. To summarize the technique, random variates (R) with some distribution (here uniform on [0, 1]) are generated until some condition ($R > 1/4$) is satisfied. When the condition is finally satisfied, the desired random variate, X (here uniform on [1/4, 1]), can be computed ($X = R$). This procedure can be shown to be correct by recognizing that the accepted values of R are conditioned values; that is, R itself does not have the desired distribution, but R conditioned on the event $\{R \geq 1/4\}$ does have the desired distribution. To show this, take $1/4 \leq a < b \leq 1$; then

$$P(a < R \leq b \mid 1/4 \leq R \leq 1) = \frac{P(a < R \leq b)}{P(1/4 \leq R \leq 1)} = \frac{b-a}{3/4} \tag{8.21}$$

which is the correct probability for a uniform distribution on [1/4, 1]. Equation (8.21) says that the probability distribution of R, given that R is between 1/4 and 1 (all other values of R are thrown out), is the desired distribution. Therefore, if $1/4 \leq R \leq 1$, set $X = R$.

The efficiency of an acceptance–rejection technique depends heavily on being able to minimize the number of rejections. In this example, the probability of a rejection is $P(R < 1/4) = 1/4$, so that the number of rejections is a geometrically distributed random variable with probability of "success" being $p = 3/4$ and mean number of rejections $(1/p - 1) = 4/3 - 1 = 1/3$. (Example 8.6 discussed the geometric distribution.) The mean number of random numbers R required to generate one variate X is one more than the number of rejections; hence, it is $4/3 = 1.33$. In other words, to generate 1000 values of X would require approximately 1333 random numbers R.

In the present situation, an alternative procedure exists for generating a uniform variate on [1/4, 1]—namely, Equation (8.5), which reduces to $X = 1/4 + (3/4)R$. Whether the acceptance–rejection technique or an alternative procedure, such as the inverse-transform technique [Equation (8.5)], is the more efficient depends on several considerations. The computer being used, the skills of the programmer and the relative inefficiency of generating the additional (rejected) random numbers needed by acceptance–rejection should be compared to the computations required by the alternative procedure. In practice, concern with generation efficiency is left to specialists who conduct extensive tests comparing alternative methods (i.e., until a simulation model begins to require excessive computer runtime due to the generator being used).

For the uniform distribution on [1/4, 1], the inverse-transform technique of Equation (8.5) is undoubtedly much easier to apply and more efficient than the acceptance–rejection technique. The main purpose of this example was to explain and motivate the basic concept of the acceptance–rejection technique. However, for some important distributions, such as the normal, gamma and beta, the inverse cdf does not exist in closed form and therefore the inverse-transform technique is difficult. These more advanced techniques are summarized by Bratley, Fox, and Schrage [1996], Fishman [1978], and Law and Kelton [2000].

In the following subsections, the acceptance–rejection technique is illustrated for the generation of random variates for the Poisson, nonstationary Poisson, and gamma distributions.

8.2.1 Poisson Distribution

A Poisson random variable, N, with mean $\alpha > 0$ has pmf

$$p(n) = P(N = n) = \frac{e^{-\alpha}\alpha^n}{n!}, \quad n = 0, 1, 2, \ldots$$

More important, however, is that N can be interpreted as the number of arrivals from a Poisson arrival process in one unit of time. Recall from Section 5.5 that the interarrival times, $A_1, A_2, \ldots$ of successive customers are exponentially distributed with rate α (i.e., α is the mean number of arrivals per unit time); in addition, an exponential variate can be generated by Equation (8.3). Thus, there is a relationship between the (discrete) Poisson distribution and the (continuous) exponential distribution:

$$N = n \tag{8.22}$$

if and only if

$$A_1 + A_2 + \cdots + A_n \leq 1 < A_1 + \cdots + A_n + A_{n+1} \tag{8.23}$$

Equation (8.22), $N = n$, says there were exactly n arrivals during one unit of time; but Relation (23) says that the nth arrival occurred before time 1 while the $(n+1)$st arrival occurred after time 1. Clearly, these two statements are equivalent. Proceed now by generating exponential interarrival times until some arrival, say $n+1$, occurs after time 1; then set $N = n$.

For efficient generation purposes, Relation (8.23) is usually simplified by first using Equation (8.3), $A_i = (-1/\alpha) \ln R_i$, to obtain

$$\sum_{i=1}^{n} -\frac{1}{\alpha} \ln R_i \leq 1 < \sum_{i=1}^{n+1} -\frac{1}{\alpha} \ln R_i$$

Next multiply through by $-\alpha$, which reverses the sign of the inequality, and use the fact that a sum of logarithms is the logarithm of a product, to get

$$\ln \prod_{i=1}^{n} R_i = \sum_{i=1}^{n} \ln R_i \geq -\alpha > \sum_{i=1}^{n+1} \ln R_i = \ln \prod_{i=1}^{n+1} R_i$$

Finally, use the relation $e^{\ln x} = x$ for any number x to obtain

$$\prod_{i=1}^{n} R_i \geq e^{-\alpha} > \prod_{i=1}^{n+1} R_i \tag{8.24}$$

which is equivalent to Relation (8.23). The procedure for generating a Poisson random variate, N, is given by the following steps:

Step 1. Set $n = 0$, $P = 1$.

Step 2. Generate a random number R_{n+1}, and replace P by $P \cdot R_{n+1}$.

Step 3. If $P < e^{-\alpha}$, then accept $N = n$. Otherwise, reject the current n, increase n by one, and return to step 2.

Notice that, upon completion of Step 2, P is equal to the rightmost expression in Relation (8.24). The basic idea of a rejection technique is again exhibited; if $P \geq e^{-\alpha}$ in Step 3, then n is rejected and the generation process must proceed through at least one more trial.

How many random numbers will be required, on the average, to generate one Poisson variate, N? If $N = n$, then $n + 1$ random numbers are required, so the average number is given by

$$E(N + 1) = \alpha + 1$$

which is quite large if the mean, α, of the Poisson distribution is large.

Example 8.8

Generate three Poisson variates with mean $\alpha = 0.2$. First, compute $e^{-\alpha} = e^{-0.2} = 0.8187$. Next, get a sequence of random numbers R from Table A.1 and follow the previously described Steps 1 to 3:

Step 1. Set $n = 0$, $P = 1$.

Step 2. $R_1 = 0.4357$, $P = 1 \cdot R_1 = 0.4357$.

Step 3. Since $P = 0.4357 < e^{-\alpha} = 0.8187$, accept $N = 0$.

Step 1–3. ($R_1 = 0.4146$ leads to $N = 0$.)

Step 1. Set $n = 0$, $P = 1$.

Step 2. $R_1 = 0.8353$, $P = 1 \cdot R_1 = 0.8353$.

Step 3. Since $P \geq e^{-\alpha}$, reject $n = 0$ and return to Step 2 with $n = 1$.

Step 2. $R_2 = 0.9952$, $P = R_1 R_2 = 0.8313$.

Step 3. Since $P \geq e^{-\alpha}$, reject $n = 1$ and return to Step 2 with $n = 2$.

Step 2. $R_3 = 0.8004$, $P = R_1 R_2 R_3 = 0.6654$.

Step 3. Since $P < e^{-\alpha}$, accept $N = 2$.

The calculations required for the generation of these three Poisson random variates are summarized as follows:

n	R_{n+1}	P	*Accept/Reject*	*Result*
0	0.4357	0.4357	$P < e^{-\alpha}$ (accept)	$N = 0$
0	0.4146	0.4146	$P < e^{-\alpha}$ (accept)	$N = 0$
0	0.8353	0.8353	$P \geq e^{-\alpha}$ (reject)	
1	0.9952	0.8313	$P \geq e^{-\alpha}$ (reject)	
2	0.8004	0.6654	$P < e^{-\alpha}$ (accept)	$N = 2$

It took five random numbers, R, to generate three Poisson variates here ($N = 0$, $N = 0$, and $N = 2$), but in the long run, to generate, say, 1000 Poisson variates with mean $\alpha = 0.2$, it would require approximately $1000(\alpha + 1)$ or 1200 random numbers.

Example 8.9

Buses arrive at the bus stop at Peachtree and North Avenue according to a Poisson process with a mean of one bus per 15 minutes. Generate a random variate, N, which represents the number of arriving buses during a 1-hour time slot. Now, N is Poisson distributed with a mean of four buses per hour. First compute $e^{-\alpha} = e^{-4} = 0.0183$. Using a sequence of 12 random numbers from Table A.1 yields the following summarized results:

n	R_{n+1}	P	*Accept/Reject*	*Result*
0	0.4357	0.4357	$P \geq e^{-\alpha}$ (reject)	
1	0.4146	0.1806	$P \geq e^{-\alpha}$ (reject)	
2	0.8353	0.1508	$P \geq e^{-\alpha}$ (reject)	
3	0.9952	0.1502	$P \geq e^{-\alpha}$ (reject)	
4	0.8004	0.1202	$P \geq e^{-\alpha}$ (reject)	
5	0.7945	0.0955	$P \geq e^{-\alpha}$ (reject)	
6	0.1530	0.0146	$P < e^{-\alpha}$ (accept)	$N = 6$

It is immediately seen that a larger value of α (here $\alpha = 4$) usually requires more random numbers; if 1000 Poisson variates were desired, approximately $1000(\alpha + 1) = 5000$ random numbers would be required.

When α is large, say $\alpha \geq 15$, the rejection technique outlined here becomes quite expensive, but fortunately an approximate technique based on the normal distribution works quite well. When the mean, α, is large, then

$$Z = \frac{N - \alpha}{\sqrt{\alpha}}$$

is approximately normally distributed with mean zero and variance 1; this observation suggests an approximate technique. First, generate a standard normal variate Z, by Equation (8.28) in Section 8.3.1, then generate the desired Poisson variate, N, by

$$N = \lceil \alpha + \sqrt{\alpha} Z - 0.5 \rceil \tag{8.25}$$

where $\lceil \cdot \rceil$ is the round-up function described in Section 8.1.7. (If $\alpha + \sqrt{\alpha} Z - 0.5 < 0$, then set $N = 0$.) The "0.5" used in the formula makes the round-up function become a "round to the nearest integer" function. Equation (8.25) is not an acceptance–rejection technique, but, when used as an alternative to the acceptance–rejection method, it provides a fairly efficient and accurate method for generating Poisson variates with a large mean.

Table 8.7 Arrival Rate for NSPP Example

t (min)	Mean Time between Arrivals (min)	Arrival Rate $\lambda(t)$ (arrivals/min)
0	15	1/15
60	12	1/12
120	7	1/7
180	5	1/5
240	8	1/8
300	10	1/10
360	15	1/15
420	20	1/20
480	20	1/20

8.2.2 Nonstationary Poisson Process

Another type of acceptance–rejection method (which is also called "thinning") can be used to generate interarrival times from a nonstationary Poisson process (NSPP) with arrival rate $\lambda(t)$, $0 \le t \le T$. A NSPP is an arrival process with an arrival rate that varies with time; see Section 5.5.2.

Consider, for instance, the arrival-rate function given in Table 8.7 that changes every hour. The idea behind thinning is to generate a stationary Poisson arrival process at the fastest rate (1/5 customer per minute in the example), but "accept" or admit only a portion of the arrivals, thinning out just enough to get the desired time-varying rate. Next we give the generic algorithm, which generates $\mathcal{T}_i$ as the time of the ith arrival. Remember that, in a stationary Poisson arrival process, the times between arrivals are exponentially distributed.

Step 1. Let $\lambda^* = \max_{0 \le t \le T} \lambda(t)$ be the maximum of the arrival rate function and set $t = 0$ and $i = 1$.

Step 2. Generate E from the exponential distribution with rate λ^* and let $t = t + E$ (this is the arrival time of the stationary Poisson process).

Step 3. Generate random number R from the $U(0, 1)$ distribution. If $R \le \lambda(t)/\lambda^*$ then $\mathcal{T}_i = t$ and $i = i + 1$.

Step 4. Go to Step 2.

The thinning algorithm can be inefficient if there are large differences between the typical and the maximum arrival rate. However, thinning has the advantage that it works for any integrable arrival rate function, not just a piecewise-constant function as in this example.

Example 8.10

For the arrival-rate function in Table 8.7, generate the first two arrival times.

Step 1. $\lambda^* = \max_{0 \le t \le T} \lambda(t) = 1/5$, $t = 0$ and $i = 1$.

Step 2. For random number $R = 0.2130$, $E = -5 \ln(0.213) = 13.13$ and $t = 0 + 13.13 = 13.13$.

Step 3. Generate $R = 0.8830$. Since $R = 0.8830 \not\le \lambda(13.13)/\lambda^* = (1/15)/(1/5) = 1/3$, do not generate the arrival.

Step 4. Go to Step 2.

Step 2. For random number $R = 0.5530$, $E = -5 \ln(0.553) = 2.96$, and $t = 13.13 + 2.96 = 16.09$.

Step 3. Generate $R = 0.0240$. Since $R = 0.0240 \le \lambda(16.09)/\lambda^* = (1/15)/(1/5) = 1/3$, set $\mathcal{T}_1 = t = 16.09$ and $i = i + 1 = 2$.

Step 4. Go to Step 2.

Step 2. For random number $R = 0.0001$, $E = -5 \ln(0.0001) = 46.05$ and $t = 16.09 + 46.05 = 62.14$.

Step 3. Generate $R = 0.1443$. Since $R = 0.1443 \le \lambda(62.14)/\lambda^* = (1/12)/(1/5) = 5/12$, set $\mathcal{T}_1 = t = 62.14$ and $i = i + 1 = 3$.

Step 4. Go to Step 2.

8.2.3 Gamma Distribution

Several acceptance–rejection techniques for generating gamma random variates have been developed. (See Bratley, Fox, and Schrage [1996]; Fishman [1978]; and Law and Kelton [2000].) One of the more efficient is by Cheng [1977]; the mean number of trials is between 1.13 and 1.47 for any value of the shape parameter $\beta \ge 1$.

If the shape parameter β is an integer, say $\beta = k$, one possibility is to use the convolution technique in Example 8.12, because the Erlang distribution is a special case of the more general gamma distribution. On the other hand, the acceptance–rejection technique described here would be a highly efficient method for the Erlang distribution especially if $\beta = k$ were large. The routine generates gamma random variates with scale parameter θ and shape parameter β—that is, with mean $1/\theta$ and variance $1/\beta\theta^2$. The steps are as follows:

Step 1. Compute $a = 1/(2\beta - 1)^{1/2}$, $b = \beta - \ln 4$.

Step 2. Generate R_1 and R_2. Set $V = R_1/(1-R_1)$.

Step 3. Compute $X = \beta V^a$.

Step 4a. If $X > b + (\beta a + 1) \ln(V) - \ln(R_1^2 R_2)$, reject X and return to Step 2.

Step 4b. If $X \le b + (\beta a + 1) \ln(V) - \ln(R_1^2 R_2)$, use X as the desired variate.

The generated variates from Step 4b will have mean and variance both equal to β. If it is desired to have mean $1/\theta$ and variance $1/\beta\theta^2$ as in Section 5.4, then include Step 5.

(**Step 5.** Replace X by $X/(\beta\theta)$.)

The basic idea of all acceptance–rejection methods is again illustrated here, but the proof of this example is beyond the scope of this book. In Step 3, $X = \beta V^a = \beta[R_1/(1-R_1)]^a$ is not gamma distributed, but rejection of certain values of X in Step 4a guarantees that the accepted values in Step 4b do have the gamma distribution.

Example 8.11

Downtimes for a high-production candy-making machine have been found to be gamma distributed with mean 2.2 minutes and variance 2.10 minutes^2. Thus, $1/\theta = 2.2$ and $1/\beta\theta^2 = 2.10$, which together imply that $\beta = 2.30$ and $\theta = 0.4545$.

Step 1. $a = 0.53$, $b = 0.91$.

Step 2. Generate $R_1 = 0.832$, $R_2 = 0.021$. Set $V = 0.832/(1 - 0.832) = 4.952$.

Step 3. Compute $X = 2.3(4.952)^{0.53} = 5.37$.

Step 4. $X = 5.37 > 0.91 + [2.3(0.53) + 1] \ln(4.952) - \ln[(0.832)^2 0.021] = 8.68$, so reject X and return to Step 2.

Step 2. Generate $R_1 = 0.434$, $R_2 = 0.716$. Set $V = 0.434/(1-0.434) = 0.767$.

Step 3. Compute $X = 2.3(0.767)^{0.53} = 2.00$.

Step 4. Since $X = 2.00 \le 0.91 + [2.3(0.53) + 1] \ln(0.767) - \ln[(0.434)^2 0.716] = 2.32$, accept X.

Step 5. Divide X by $\beta\theta = 1.045$ to get $X = 1.91$.

This example took two trials (i.e., one rejection) to generate an acceptable gamma-distributed random variate, but, on the average, to generate, say, 1000 gamma variates, the method will require between 1130 and 1470 trials, or equivalently, between 2260 and 2940 random numbers. The method is somewhat cumbersome for hand calculations, but is easy to program on the computer and is one of the most efficient gamma generators known.

8.3 SPECIAL PROPERTIES

"Special properties" are just as the name implies: They are variate-generation techniques that are based on features of a particular family of probability distributions, rather than being general-purpose techniques like the inverse-transform or acceptance–rejection techniques.

8.3.1 Direct Transformation for the Normal and Lognormal Distributions

Many methods have been developed for generating normally distributed random variates. The inverse-transform technique cannot easily be applied, however, because the inverse cdf cannot be written in closed form. The standard normal cdf is given by

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt, \qquad -\infty < x < \infty$$

This section describes an intuitively appealing direct transformation that produces an independent pair of standard normal variates with mean zero and variance 1. The method is due to Box and Muller [1958]. Although not as efficient as many more modern techniques, it is easy to program in a scientific language, such as FORTRAN, C, C++, Visual Basic, or Java. We then show how to transform a standard normal variate into a normal variate with mean μ and variance σ^2. Once we have a method (this or any other) for generating X from a $N(\mu, \sigma^2)$ distribution, then we can generate a lognormal random variate Y with parameters μ and σ^2 by using the direct transformation $Y = e^X$. (Recall that μ and σ^2 are *not* the mean and variance of the lognormal; see Equations (5.58) and (5.59).)

Consider two standard normal random variables, Z_1 and Z_2, plotted as a point in the plane as shown in Figure 8.7 and represented in polar coordinates as

$$\begin{aligned} Z_1 &= B\cos\theta \\ Z_2 &= B\sin\theta \end{aligned} \tag{8.26}$$

It is known that $B^2 = Z_1^2 + Z_2^2$ has the chi-square distribution with 2 degrees of freedom, which is equivalent to an exponential distribution with mean 2. Thus, the radius, B, can be generated by use of Equation (8.3):

$$B = (-2\ln R)^{1/2} \tag{8.27}$$

By the symmetry of the normal distribution, it seems reasonable to suppose, and indeed it is the case, that the angle is uniformly distributed between 0 and 2π radians. In addition, the radius, B, and the angle, θ, are mutually independent. Combining Equations (8.26) and (8.27) gives a direct method for generating two independent standard normal variates, Z_1 and Z_2, from two independent random numbers, R_1 and R_2:

$$\begin{aligned} Z_1 &= (-2\ln R_1)^{1/2}\cos(2\pi R_2) \\ Z_2 &= (-2\ln R_1)^{1/2}\sin(2\pi R_2) \end{aligned} \tag{8.28}$$

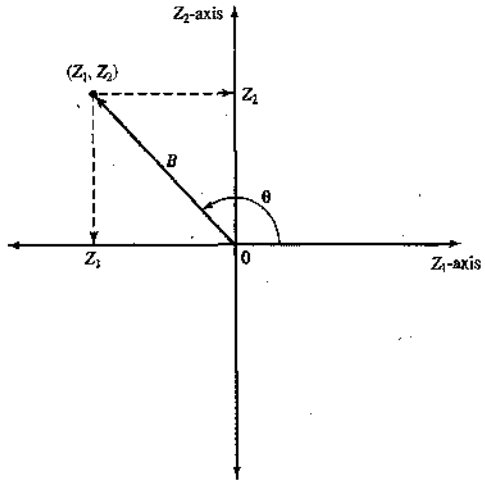


Figure 8.7 Polar representation of a pair of standard normal variables.

To illustrate the generation scheme, consider Equation (8.28) with $R_1 = 0.1758$ and $R_2 = 0.1489$. Two standard normal random variates are generated as follows:

$$\begin{aligned} Z_1 &= [-2\ln(0.1758)]^{1/2}\cos(2\pi 0.1489) = 1.11 \\ Z_2 &= [-2\ln(0.1758)]^{1/2}\sin(2\pi 0.1489) = 1.50 \end{aligned}$$

To obtain normal variates X_i with mean μ and variance σ^2, we then apply the transformation

$$X_i = \mu + \sigma Z_i \tag{8.29}$$

to the standard normal variates. For example, to transform the two standard normal variates into normal variates with mean $\mu = 10$ and variance $\sigma^2 = 4$, we compute

$$\begin{aligned} X_1 &= 10 + 2(1.11) = 12.22 \\ X_2 &= 10 + 2(1.50) = 13.00 \end{aligned}$$

8.3.2 Convolution Method

The probability distribution of a sum of two or more independent random variables is called a convolution of the distributions of the original variables. The convolution method thus refers to adding together two or more random variables to obtain a new random variable with the desired distribution. This technique can be applied to obtain Erlang variates and binomial variates. What is important is not the cdf of the desired random variable, but rather its relation to other variates more easily generated.

Example 8.12: Erlang Distribution

As was discussed in Section 5.4, an Erlang random variable X with parameters (k, θ) can be shown to be the sum of k independent exponential random variables, X_i, $i = 1, \ldots, k$, each having mean $1/k\theta$—that is,

$$X = \sum_{i=1}^{k} X_i$$

The convolution approach is to generate $X_1, X_2, \ldots, X^k$, then sum them to get X. In the case of the Erlang, each X_i can be generated by Equation (8.3) with $1/\lambda = 1/k\theta$. Therefore, an Erlang variate can be generated by

$$X = \sum_{i=1}^{k} -\frac{1}{k\theta} \ln R_i$$
$$= -\frac{1}{k\theta} \ln\left(\prod_{i=1}^{k} R_i\right) \tag{8.30}$$

It is more efficient computationally to multiply all the random numbers first and then to compute only one logarithm.

Example 8.13

Trucks arrive at a large warehouse in a completely random fashion that is modeled as a Poisson process with arrival rate $\lambda = 10$ trucks per hour. The guard at the entrance sends trucks alternately to the north and south docks. An analyst has developed a model to study the loading/unloading process at the south docks and needs a model of the arrival process at the south docks alone. An interarrival time X between successive truck arrivals at the south docks is equal to the sum of two interarrival times at the entrance and thus it is the sum of two exponential random variables, each having mean 0.1 hour, or 6 minutes. Thus, X has the Erlang distribution with $K = 2$ and mean $1/\theta = 2/\lambda = 0.2$ hour. To generate the variate X, first obtain $K = 2$ random numbers from Table A.1, say $R_1 = 0.937$ and $R_2 = 0.217$. Then, by Equation (8.30),

$$X = 0.1 \ln[0.937(0.217)]$$
$$= 0.159 \text{ hour} = 9.56 \text{ minutes}$$

In general, Equation (8.30) implies that K uniform random number are needed for each Erlang variate generated. If K is large, it is more efficient to generate Erlang variates by other techniques, such as one of the many acceptance–rejection techniques for the gamma distribution given in Section 8.2.3, or by Bratley, Fox and Schrage [1996], Fishman [1978], and Law and Kelton [2000].

8.3.3 More Special Properties

There are many relationships among probability distributions that can be exploited for random-variate generation. The convolution method in the Section 8.3.2 is one example. Another particularly useful example is the relationship between the beta distribution and the gamma distribution.

Suppose that X_1 has a gamma distribution with shape parameter β_1 and scale parameter $\theta_1 = 1/\beta_1$, while X_2 has a gamma distribution with shape parameter β_2 and scale parameter $\theta_2 = 1/\beta_2$, and that these two random variables are independent. Then

$$Y = \frac{X_1}{X_1 + X_2}$$

has a beta distribution with parameters β_1 and β_2 on the interval (0, 1). If, instead, we want Y to be defined on the interval (a, b), then set

$$Y = a + (b-a)\left(\frac{X_1}{X_1 + X_2}\right)$$

Thus, using the acceptance–rejection technique for gamma variates defined in the previous section, we can generate beta variates, with two gamma variates required for each beta.

Although this method of beta generation is convenient, there are faster methods based on acceptance–rejection ideas. See, for instance, Devroye [1986] or Dagpunar [1988].

8.4 SUMMARY

The basic principles of random-variate generation via the inverse-transform technique, the acceptance–rejection technique, and special properties have been introduced and illustrated by examples. Methods for generating many of the important continuous and discrete distributions, plus all empirical distributions, have been given. See Schmeiser [1980] for an excellent survey; for a state-of-the-art treatment, the reader is referred to Devroye [1986] or Dagpunar [1988].

REFERENCES

BRATLEY, P., B. L. FOX, AND L. E. SCHRAGE [1996], *A Guide to Simulation*, 2d ed., Springer-Verlag, New York.

BOX, G. E. P., AND M. F. MULLER [1958], "A Note on the Generation of Random Normal Deviates," *Annals of Mathematical Statistics*, Vol. 29, pp. 610–611.

CHENG, R. C. H. [1977], "The Generation of Gamma Variables," *Applied Statistician*, Vol. 26, No. 1, pp. 71–75.

DAGPUNAR, J. [1988], *Principles of Random Variate Generation*, Clarendon Press, Oxford, New York.

DEVROYE, L. [1986], *Non-Uniform Random Variate Generation*, Springer-Verlag, New York.

FISHMAN, G. S. [1978], *Principles of Discrete Event Simulation*, Wiley, New York.

LAW, A. M., AND W. D. KELTON [2000], *Simulation Modeling & Analysis*, 3d ed., McGraw–Hill, New York.

RIPLEY, B. D. [1987], *Stochastic Simulation*, Wiley, New York.

SCHMEISER, BRUCE W. [1979], "Approximations to the Inverse Cumulative Normal Function for Use on Hand Calculators," *Applied Statistics*, Vol. 28, pp. 175–176.

SCHMEISER, B. W. [1980], "Random Variate Generation: A Survey," in *Simulation with Discrete Models: A State of the Art View*, T. I. Ören, C. M. Shub, and P. F. Roth, eds., IEEE, New York.

EXERCISES

1. Develop a random-variate generator for X with pdf

$$f(x) = \begin{cases} \dfrac{3x^2}{2}, & -1 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$$

2. Develop a generation scheme for the triangular distribution with pdf

$$f(x) = \begin{cases} \dfrac{1}{2}(x-2), & 2 \le x \le 3 \\ \dfrac{1}{2}\left(2 - \dfrac{x}{3}\right), & 3 < x \le 6 \\ 0, & \text{otherwise} \end{cases}$$

Generate 10 values of the random variate, compute the sample mean, and compare it to the true mean of the distribution.

3. Develop the triangular random-variate generator with range (0, 12) and mode 5.

4. Develop a generator for a triangular distribution with range (1, 10) and a mean of 4.

5. Given the following cdf for a continuous variable with range from −3 to 4, develop a generator for the variable.

$$F(x)=\begin{cases}0, & x\le -3\\ \frac{1}{2}+\frac{x}{6}, & -3<x\le 0\\ \frac{1}{2}+\frac{x^2}{32}, & 0<x\le 4\\ 1, & x>4\end{cases}$$

6. Given the cdf $F(x)=x^4/16$ on $0\le x\le 2$, develop a generator for this distribution.

7. Given the pdf $f(x)=x^2/9$ on $0\le x\le 3$, develop a generator for this distribution.

8. The pdf of a random variable is

$$f(x)=\begin{cases}\frac{1}{5}, & 0<x\le 3\\ \frac{1}{8}, & 3<x\le 9\\ 0, & \text{otherwise}\end{cases}$$

Develop the random-variate generator.

9. The cdf of a discrete random variable X is given by

$$F(x)=\frac{x(x+1)(2x+1)}{n(n+1)(2n+1)},\quad x=1,2,\ldots,n$$

When $n=4$, generate three values of X, using $R_1=0.83$, $R_2=0.24$, and $R_3=0.57$.

10. Times to failure for an automated production process have been found to be randomly distributed with a Weibull distribution with parameters $\beta=2$ and $\alpha=10$. Derive Equation (8.6), and then use it to generate five values from this Weibull distribution, using five random numbers taken from Table A.1.

11. The details of time taken by a mechanic to repair a breakdown are

Repair Time Range (Hours)	*Frequency*
1–2	15
2–3	12
3–4	14
4–5	25
5–6	32
6–7	14

Develop a lookup table and generate five repair times using random numbers.

12. In an inventory system, the lead time is found to follow uniform distribution with mean 10 days and half width 3 days. Generate five lead times.

13. For a preliminary version of a simulation model, the number of pallets, X, to be loaded onto a truck at a loading dock was assumed to be uniformly distributed between 8 and 24. Devise a method for generating X, assuming that the loads on successive trucks are independent. Use the technique of Example 8.5 for discrete uniform distributions. Finally, generate loads for 10 successive trucks by using four-digit random numbers.

14. Develop a method for generating values from a negative binomial distribution with parameters p and k, as described in Section 5.3. Generate 3 values when $p=0.8$ and $k=2$. [*Hint*: Think about the definition of the negative binomial as the number of Bernoulli trials until the kth success.]

15. The weekly demand, X, for a slow-moving item has been found to be approximated well by a geometric distribution on the range {0, 1, 2, ...} with mean weekly demand of 2.5 items. Generate 10 values of X, demand per week, using random numbers from Table A.1. (*Hint*: For a geometric distribution on the range {q, $q+1$, ...} with parameter p, the mean is $1/p+q-1$.)

16. In Exercise 15, suppose that the demand has been found to have a Poisson distribution with mean 2.5 items per week. Generate 10 values of X, demand per week, using random numbers from Table A.1. Discuss the differences between the geometric and the Poisson distributions.

17. Service time of a bank teller is found to follow normal with $\mu=5$ minutes and $\sigma=1$ minute. Generate five service times.

18. The time to attend a breakdown call is found to follow exponential with a mean of 2 hours. Generate exponential random variates representing the time to attend.

19. A machine is taken out of production either if it fails or after 5 hours, whichever comes first. By running similar machines until failure, it has been found that time to failure, X, has the Weibull distribution with $\alpha=8$, $\beta=0.75$, and $\nu=0$ (refer to Sections 5.4 and 8.1.3). Thus, the time until the machine is taken out of production can be represented as $Y=\min(X, 5)$. Develop a step-by-step procedure for generating Y.

20. In an art gallery, the arrival of visitors follow Poisson with a mean of 4 per hour. Generate the arrivals for the next 1 hour.

21. Develop a technique for generating a binomial random variable, X, via the convolution technique. [*Hint*: X can be represented as the number of successes in n independent Bernoulli trials, each success having probability p. Thus, $X=\sum_{i=1}^{n}X_i$, where $P(X_i=1)=p$ and $P(X_i=0)=1-p$.]

22. Develop an acceptance–rejection technique for generating a geometric random variable, X, with parameter p on the range {0, 1, 2, ...}. (*Hint*: X can be thought of as the number of trials before the first success occurs in a sequence of independent Bernoulli trials.)

23. Write a computer program to generate exponential random variates for a given mean value. Generate 1000 values and verify the variates generated using chi-square test.

24. Develop a computer program to generate binomial random variates with p = probability of success, n = number of trials, and x = random number between 0 and 1.

25. Write a computer program to generate 500 normal random variates of given μ and σ values and prepare a histogram.

26. Many spreadsheet, symbolic-calculation, and statistical-analysis programs have built-in routines for generating random variates from standard distributions. Try to find out what variate-generation methods

are used in one of these packages by looking at the documentation. Should you trust a variate generator if the method is not documented?

27. Suppose that, somehow, we have available a source of exponentially distributed random variates with mean 1. Write an algorithm to generate random variates with a triangular distribution by transforming the exponentially distributed random variates. [*Hint*: First transform to obtain uniformly distributed random variates.]

28. A study is conducted on the arrival of customers in a bus stop during the post lunch period. The system starts at 12.30 P.M. and the arrival rate per hour during different intervals of time are

Time	*Arrival Rate/Hour*
12.30–1.30 P.M.	20
1.30–2.30 P.M.	35
2.30–3.30 P.M.	60
3.30–4.30 P.M.	80

Generate arrivals from this NSPP.

29. Generate 10 values from a beta distribution on the interval [0, 1] with parameters $\beta_1 = 1.47$ and $\beta_2 = 2.16$. Next transform them to be on the interval [−10, 20].

Part IV

Analysis of Simulation Data