SIMULATION

Jacques Resing

1 Introduction

Many systems that we encounter in operation research applications are too complicated for an exact mathematical analysis. In those situations we can decide to make some simplifications of the system leading to a problem which is mathematically tractable. However, alternatively we can decide to do a simulation study of the original, complicated system. This course is devoted to this latter approach. Several issues that arise during a simulation study will be discussed.

These lecture notes are organized as follows. In section 2 we discuss the advantages and disadvantages of a simulation study in comparison with an ordinary mathematical analysis. Next, in section 3 we will concentrate on so-called *discrete-event* simulations. These are simulations of systems for which the state of the system only changes at some random event times. These systems typically arise in the modelling of production and communication systems, where the event times are for example the completion times of jobs at machines or the arrival times of messages at communication links.

A simulation of a system containing random components requires of course a method of generating random variables. In section 4 we discuss the generation of uniformly distributed random variables. After that, in section 5 we show how these uniform random variables can be used to generate other discrete and continuous random variables.

The topic of section 6 will be the statistical aspects of a simulation study. Because running a simulation is like doing an experiment, the outcome of a simulation study will be an estimator of some performance measure of interest. Hence we are interested in the quality of the estimator. In section 7 we show how one can reduce the variance of the estimator by using variance reduction techniques. Finally, in section 8 we discuss what to do if we are interested in the simulation of rare events. This topic is particularly important in the simulation of communication systems in which the loss of packets is typically a rare event.

2 Analysis versus Simulation

In this section we will discuss the advantages and disadvantages of a simulation study in comparison with an ordinary mathematical analysis. We will do that using an example of a two-stage production line with one unreliable machine.

Consider a production line consisting of two machines with a finite storage buffer between the two machines (see Figure 1).

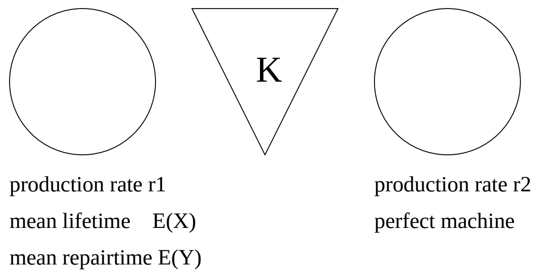

Figure 1: Two-machine production line

Machine 1 produces material and puts it into the buffer and machine 2 takes the material out of that buffer. The material is a fluid flowing in and out of the buffer, i.e. we are dealing with a fluid-flow model. The production rate of machine j is equal to r_j, $j = 1, 2$ and we assume $r_1 > r_2$. The first machine is subject to breakdowns. The sequence $X_1, Y_1, X_2, Y_2, \ldots$ of successive lifetimes and repairtimes forms an alternating renewal process, for which only the means $E(X)$ and $E(Y)$ are known. The second machine is a perfect machine. The size of the storage buffer is equal to K. Clearly, whenever the buffer is full, the production rate of machine 1 is also reduced to r_2. Performance measure of interest is the long-term average production rate of the production line as a function of K. A possible time path realization for the buffer content is shown in Figure 2.

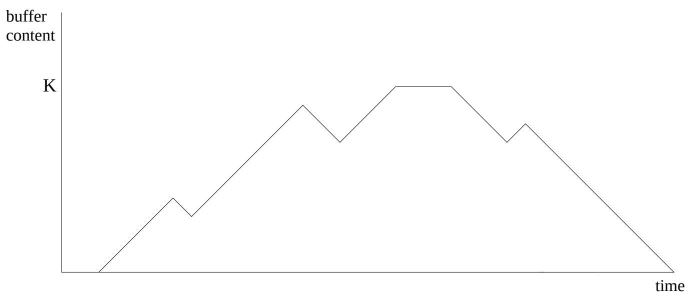

Figure 2: Time path realization of the buffer content

2.1 Analysis

First we try to solve the problem by analysis. In particular we will look at the cases with zero and infinite buffer and at the case with exponential lifetimes and repairtimes.

2.1.1 Zero and infinite buffer

In the case that the size of the buffer is equal to zero, the production rate of the line is equal to r_2 if the first machine is working and the production rate is equal to 0 if the first machine is in repair. Because the fraction of time that the first machine is working is equal to $E(X)/(E(X)+E(Y))$, it follows immediately that the long-term average production rate equals $r_2 E(X)/(E(X)+E(Y))$.

In the case that the size of the buffer is infinite the first machine can always work at full speed (the buffer never becomes full) and hence the average production rate of the first machine equals $r_1 E(X)/(E(X)+E(Y))$. Hence, the average production rate of the complete production line equals $\min(r_1 E(X)/(E(X)+E(Y)), r_2)$.

Clearly the average production rate in the case of a zero buffer is a lowerbound and in the case of an infinite buffer an upperbound for the average production rate in the case of an arbitrary buffer size K. Remark furthermore that both in the case of a zero buffer and in the case of an infinite buffer the average production rate does not depend on the actual *distribution* of the lifetimes and the repairtimes of the first machine.

2.1.2 Exponential lifetimes and repairtimes

Next, assume that both lifetimes and repairtimes are exponentially distributed with parameters $\lambda = 1/E(X)$ and $\mu = 1/E(Y)$ respectively. Then the process describing the state of the first machine and the content of the buffer is a continuous-time Markov process and we can again obtain the average production rate of the line by analysis.

More specifically, the state of the system is described by the pair (i, x) where i is the state of the first machine ($i = 1$ indicates that the machine is working, $i = 0$ indicates that the machine is in repair), and x is the buffer contents, $0 \le x \le K$. In order to analyze the system we introduce the time-dependent distribution function $F_t(i, x)$ and the stationary distribution function $F(i, x)$, indicating the probability that the first machine is in state i and the buffer content does not exceed x at time t respectively in steady state. For the time-dependent distribution function we can derive the set of equations

$$\begin{aligned}
F_{t+\Delta t}(0,x) &= F_t(0, x + r_2\Delta t)(1-\mu\Delta t) + F_t(1, x + O(\Delta t))\lambda\Delta t + O((\Delta t)^2),\\
F_{t+\Delta t}(1,x) &= F_t(1, x + (r_2 - r_1)\Delta t)(1-\lambda\Delta t) + F_t(0, x + O(\Delta t))\mu\Delta t + O((\Delta t)^2).
\end{aligned}$$

Dividing the equations by Δt and letting Δt tend to zero leads to a set of partial differential equations for the time-dependent distribution function

$$\begin{aligned}
\frac{\partial F_t(0,x)}{\partial t} - r_2 \frac{\partial F_t(0,x)}{\partial x} &= -\mu F_t(0,x) + \lambda F_t(1,x),\\
\frac{\partial F_t(1,x)}{\partial t} + (r_1 - r_2) \frac{\partial F_t(1,x)}{\partial x} &= \mu F_t(0,x) - \lambda F_t(1,x).
\end{aligned}$$

Now, letting t tend to infinity and using that

$$\lim_{t\to\infty} F_t(i,x) = F(i,x) \qquad \text{and} \qquad \lim_{t\to\infty} \frac{\partial F_t(i,x)}{\partial t} = 0,$$

we obtain for the stationary distribution

$$\begin{aligned} -r_2 F'(0,x) &= -\mu F(0,x) + \lambda F(1,x), \\ (r_1 - r_2) F'(1,x) &= \mu F(0,x) - \lambda F(1,x), \end{aligned}$$

which is equivalent to

$$\begin{pmatrix} F'(0,x) \\ F'(1,x) \end{pmatrix} = A \begin{pmatrix} F(0,x) \\ F(1,x) \end{pmatrix},$$

with

$$A = \begin{pmatrix} \mu/r_2 & -\lambda/r_2 \\ \mu/(r_1 - r_2) & -\lambda/(r_1 - r_2) \end{pmatrix}.$$

The solution of this equation is given by

$$\begin{pmatrix} F(0,x) \\ F(1,x) \end{pmatrix} = C_1 \vec{v_1} e^{\alpha_1 x} + C_2 \vec{v_2} e^{\alpha_2 x},$$

where α_1 and α_2 are eigenvalues of the matrix A and $\vec{v_1}$ and $\vec{v_2}$ the corresponding eigenvectors.

Straightforward calculations yield $\alpha_1 = 0$, $\alpha_2 = \mu/r_2 - \lambda/(r_1 - r_2)$, $\vec{v_1} = (\lambda, \mu)^T$ and $\vec{v_2} = (r_1 - r_2, r_2)^T$. So,

$$\begin{pmatrix} F(0,x) \\ F(1,x) \end{pmatrix} = C_1 \begin{pmatrix} \lambda \\ \mu \end{pmatrix} + C_2 \begin{pmatrix} r_1 - r_2 \\ r_2 \end{pmatrix} e^{\alpha_2 x}.$$

Finally, C_1 and C_2 follow from the boundary conditions $F(1,0) = 0$ and $F(0,K) = \lambda/(\lambda+\mu)$. It follows that

$$\begin{pmatrix} F(0,x) \\ F(1,x) \end{pmatrix} = \frac{\lambda}{\lambda+\mu} \cdot \left(\lambda r_2 - \mu(r_1 - r_2) e^{\alpha_2 K}\right)^{-1} \cdot \left[r_2 \begin{pmatrix} \lambda \\ \mu \end{pmatrix} - \mu_1 e^{\alpha_2 x} \begin{pmatrix} r_1 - r_2 \\ r_2 \end{pmatrix} \right].$$

The average production rate now equals

$$r_2 \cdot (1 - F(0,0)) = r_2 \cdot \frac{\mu}{\lambda+\mu} \cdot \frac{\lambda r_1 - (\lambda+\mu)(r_1 - r_2) e^{\alpha_2 K}}{\lambda r_2 - \mu(r_1 - r_2) e^{\alpha_2 K}}. \tag{1}$$

Equation (1) gives a closed form expression for the average production rate as a function of the input parameters λ, μ, r_1, r_2 and K, and so we can immediately see what happens with this average production rate if we change one of the parameters. This is something that we can never obtain from a simulation. For example, Table 1 shows the average production rate as a function of K in the case that $\lambda = 1/9, \mu = 1, r_1 = 5$ and $r_2 = 4$.

K	0	2	4	5	6	8	10	∞
prod. rate	3.600	3.746	3.829	3.857	3.880	3.914	3.938	4.000

Table 1: Average production rate as a function of K

Hence, if an analysis is possible we always prefer analysis above simulation. However, to be able to do the analysis we had to assume that both repairtimes and lifetimes are exponentially distributed (otherwise we were not dealing with a Markov process). One big advantage of a simulation of this production line is that we don't have to assume exponentiality. Hence, we can use simulation for example to investigate whether or not the average production rate is sensitive to the distribution of the repairtimes and/or lifetimes.

2.2 Simulation

The following program illustrates a simulation of the production line.

```
PROGRAM: TWO-MACHINE PRODUCTION LINE

INITIALISATION

time := 0;
buffercontent := 0;
empty := 0;

MAIN PROGRAM

while time < runlength
do
    x := new lifetime;
    time := time + x;
    buffercontent := min ( buffercontent + x * (r1 − r2), K);
    y := new repairtime;
    time := time + y;
    if buffercontent - y * r2 < 0
    then empty := empty + y - buffercontent / r2;
    buffercontent := max ( buffercontent - y * r2, 0);
end

OUTPUT

throughput := r2 * (1 - empty / time);
```

Remark that the behaviour of the system only significantly changes at the moments that machine 1 breaks down and the moments that machine 1 is repaired. At these instants the buffer content switches from increase to decrease or from decrease to increase. In the simulation we jump from one of these so-called event times to the next and calculate the buffer content at these event times. Once, we know the buffer content at these moments we are able to estimate the average production rate of the production line. At the moment we do not answer the question how we can generate realizations of the lifetimes and repairtimes. If we do a simulation, we obtain one value for the average production rate for one choice of the input parameters. If we run a simulation for the parameter choice of Table 1 and choose $K = 4$, then a possible outcome of our simulation is that the average production rate equals 3.827. However, if we do another run the outcome may be 3.822. So, we must realize that doing a simulation is nothing else than doing an experiment. Different experiments can lead to different results. This immediately lead to questions like:

1. How good is the quality of the outcome of a simulation?
2. What is a good choice for the runlength of a simulation?
3. What is the effect of the choice of the initial conditions on the outcome of a simulation?

These questions will be answered during this course. Another disadvantage of a simulation is that if we change one of the input parameters we have to do a new simulation run. For example, if we change the maximum buffer size from $K = 4$ to $K = 5$, a new simulation run leads to the outcome 3.854.

However, big advantage of a simulation is that only by a slight modification of the program we can obtain results for the case in which e.g. the lifetimes or repairtimes are NOT exponentially distributed. Remark that this is not the case for the analysis because, as mentioned before, we then loose the fact that the process is a continuous-time Markov process. For example, if we change the assumption of exponential repairtimes to deterministic repairtimes (with the same mean) a slight modification of our program leads to the outcome 3.923 in the case of buffer size $K = 4$. Hence, simulation is very suitable to investigate whether or not the average production rate is sensitive to the distribution of the repairtimes and/or lifetimes.

3 Discrete-Event Simulation

Roughly speaking, there are three different kinds of systems: continuous systems, discrete systems and discrete-event systems. A *continuous system* is a system which state varies continuously in time. Such systems are usually described by a set of differential equations. For example, continuous systems often arise in chemical applications. A *discrete system* is a system which is observed only at some fixed regular time points. Such systems are usually described by a set of difference equations. An example of a discrete system is an inventory model in which we inspect the stock only once a week. The characteristic feature of a *discrete-event system* is that the system is completely determined by a sequence of random event times $t_1, t_2, \ldots$, and by the changes in the state of the system which take place at these moments. Between these event times the state of the system may also change, however, according to some deterministic pattern. A first example of a discrete-event simulation was given in the previous section when we considered the two-machine production line. A second example of a discrete-event system is a single server queueing system.

A way to simulate a continuous system or a discrete system is by looking at the model at regular time points $0, t, 2t, \ldots$. Events which occur somewhere in the interval between two of these points are taken to happen at one of the end points of the interval. This approach is called *synchronous* simulation. For continous systems it might be necessary to take t very small in order to obtain a sufficiently accurate simulation and hence the simulation may be very slow.

For a discrete-event system it is not efficient to use a synchronous simulation. Instead, we use an *asynchronous* simulation, i.e. we jump from one event time to the next and describe the changes in the state at these moments. In this course, we will concentrate on asynchronous simulations of discrete-event systems.

When you write a simulation program for a discrete-event system you can take an *event scheduling approach* or a *process-interaction* approach. The event-scheduling approach concentrates on the events and how they affect system state. The process-interaction approach concentrates on the different entities in the system (e.g. the customers, the server and the arrival generator) and describes the sequence of events and activities such entities execute or undergo during their stay in the system. When using a general-purpose language, such as Fortran, Pascal or C, one uses in general the event-scheduling approach. However, simulation languages as Simscript or GPSS use the process-interaction approach. In this course, we will concentrate on the event-scheduling approach.

As an example of a discrete-event simulation we will consider the $G/G/1$ queueing system. Suppose that we use a simulation study to obtain an estimator for the long-term average waiting time of customers in this queueing system.

First, we remark that we can simulate the waiting time in a $G/G/1$ queue by using a simple discrete simulation instead of a discrete-event simulation. If we denote by W_n the waiting time of the n-th customer, by B_n the service time of the n-th customer and by A_n the interarrival time between the n-th and the $(n+1)$-st customer, then we can obtain the following difference equation for W_n:

$$W_{n+1} = \max(W_n + B_n - A_n, 0).$$

Hence, an estimator for the long-term average waiting time of customers can be obtained by the following discrete simulation program:

```
PROGRAM: G/G/1 QUEUEING SYSTEM
(discrete simulation)

INITIALIZATION

n := 0;
w := 0;
sum := 0;

MAIN PROGRAM

while n < maximum
do
        a := new interarrival time;
        b := new service time;
        w := max(w+b-a,0);
        sum := sum + w;
        n := n + 1;
end

OUTPUT

mean waiting time := sum / n ;
```

However, if we want to obtain more information from the simulation (e.g. the queue length distribution or the fraction of time the server is busy) then a discrete simulation would not be suitable. Also, if we consider more complicated queueing systems a simple recursive formula is not available and hence a discrete simulation would not work anymore. Hence, we will also describe a discrete-event simulation for the long-term waiting time in the $G/G/1$ queue.

The event times are the arrival and departure moments of customers. A difference with the example of the production line is that in the $G/G/1$ case we do not know beforehand in what sequence the different events will take place. Whether the next event is an arrival event or a departure event depends on the realizations of the interarrival times and service times of the customers. Therefore, we will make a so-called *event list* in which we list the next arrival event and the next departure event and we jump to the event which will occur the first.

Let us look in some detail what happens at the different event times for the $G/G/1$ queueing system. At an arrival event, we certainly have to generate a new interarrival time.

Furthermore, if the server is free, we start a new service, i.e. we generate a new service time. On the other hand, if the server is busy, we add the customer to the queue. At a departure event, we remove the leaving customer from the system. Furthermore, if the queue is not empty, we start a new service and registrate how long the customer was in the queue. This leads to the following program:

```
PROGRAM: G/G/1 QUEUEING SYSTEM
(discrete-event simulation)

INITIALIZATION

time := 0;
queue := nil;
sum := 0; throughput := 0;
generate first interarrival time;

MAIN PROGRAM

while time < runlength
do
    case nextevent of
      arrival event:
        time := arrivaltime;
        add customer to queue;
        start new service if server is idle;
        generate new interarrival time;
      departure event:
        time := departuretime;
        throughput := throughput + 1;
        remove customer from queue;
        if queue not empty
            sum := sum + waiting time;
            start new service;
end

OUTPUT
mean waiting time := sum / throughput ;
```

Remark that, in order to be able to calculate the waiting time of a customer, we must remember the arrival time of a customer at the moment that the customer gets into service. Furthermore, we must be able to add and remove customers from a queue. In principal, this can be done by using an array as data structure. However, we prefer to use records and linked lists. Each customer is represented by a record consisting of his arrival time and a reference to the customer who is next in the queue.

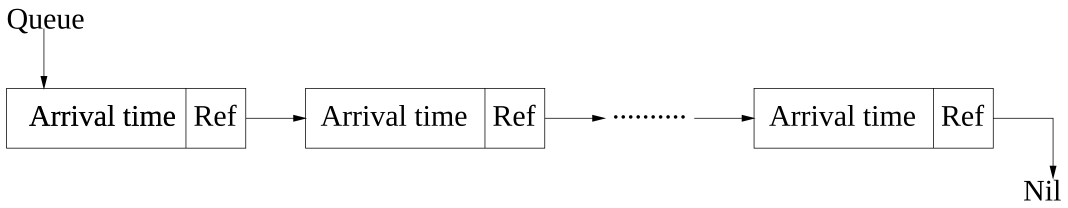

Figure 3: Linked list

In general, in an event-oriented approach of a discrete-event simulation there are a number of, say, N different events, each with its own corresponding activities $A_1, \ldots, A_N$. In the event list we keep track of the time points $T_1, \ldots, T_N$ at which the next events of the different types occur. The simulation then consists of finding the smallest T_i, setting the current time to this event time and executing the corresponding activities. A typical construction of a discrete-event simulation program is:

```
DISCRETE-EVENT SIMULATION

INITIALIZATION

set simulation time equal to 0;
initialize system state and statistical counters;
initialize event list;

MAIN PROGRAM

while simulation time < runlength
do
    determine next event type;
    advance simulation time;
    update system state + statistical counters;
    generate future events + add them to event list;
end

OUTPUT

compute + print estimates of interest;
```

3.1 Verification of a simulation model

One important issue during a simulation study is the following: How do we check whether a program of a discrete event simulation model is operating as intended? One of the techniques to do this is by using a *trace*. In a trace, the state of the simulated system, i.e. the values of the state variables, the statistical counters and the contents of the event list, is printed just after each event occurs. For a small runlength the results of the trace can then be compared with calculations by hand.

Furthermore, a model should, when possible, be run under special assumptions for which results can also be obtained by analysis. For example, we can first simulate the production line of section 2 with zero and infinite buffer. Also, we can simulate the system with exponential lifetimes and repairtimes to see whether or not the result of the simulation coincides with the result of the analysis. For the $G/G/1$ system we can first simulate the system with exponential interarrival and service times, for which it is possible to obtain exact results (see e.g. the courses on Stochastic Processes (2S500) and Queueing Theory (2S520).

Finally, it is of course sensible to write and check a simulation program in modules or subprograms.

4 Random-Number Generators

A simulation of a system which contains random components requires a method of generating random numbers. This section is devoted to the generation of random numbers from the uniform $(0, 1)$ distribution. Although this is only one of all possible distribution functions, it is very important to have an efficient way of generating uniformly distributed random numbers. This is due to the fact that random variables from other distributions can often be obtained by transforming uniformly distributed random variables. We come back to this issue in the next section.

A good random number generator should satisfy the following properties:

1. The generated numbers should satisfy the property of uniformity.
2. The generated numbers should satisfy the property of independence.
3. The random numbers should be replicable.
4. It should take a long time before numbers start to repeat.
5. The routine should be fast.
6. The routine should not require a lot of storage.

The generation of random numbers has a long history. In the beginning, random numbers were generated by hand using methods as throwing dice or drawing numbered balls. However, nowadays only numerical ways to generate random numbers are used. Usually, the random numbers are generated in a sequence, each number being completely determined by one or several of its predecessors. Hence, these generators are often called *pseudo-random number generators*.

4.1 Pseudo-Random Number Generators

Most pseudo-random number generators can be characterized by a five-tuple (S, s_0, f, U, g), where S is a finite set of states, $s_0 \in S$ is the initial state, $f : S \to S$ is the transition function, U is a finite set of output values, and $g : S \to U$ is the output function.

The pseudo random-number generator then operates as follows:

- (1) Let $u_0 = g(s_0)$.
- (2) For $i = 1, 2, \ldots$ let $s_i = f(s_{i-1})$ and $u_i = g(s_i)$.

The sequence $(u_0, u_1, \ldots)$ is the output of the generator. The choice of a fixed initial state s_0 rather than a probabilistic one makes replication of the random numbers possible.

Let us next give a number of examples of pseudo-random number generators.

4.1.1 Midsquare Method

One of the first pseudo-random number generators was the midsquare method. It starts with a fixed initial number, say of 4 numbers, called the seed. This number is squared and the middle digits of this square become the second number. The middle digits of this second number are then squared again to generate the third number and so on. Finally, we get realizations from the uniform (0,1) distribution after placement of the decimal point (i.e. after division by 10000). The choice of the seed is very important as the next examples will show.

Example 4.1

- *If we take as seed* $Z_0 = 1234$*, then we will get the sequence of numbers* 0.1234, 0.5227, 0.3215, 0.3362, 0.3030, 0.1809, 0.2724, 0.4201, 0.6484, 0.0422, 0.1780, 0.1684, 0.8358, 0.8561, 0.2907, ...

- *If we take as seed* $Z_0 = 2345$*, then we get the sequence of numbers* 0.2345, 0.4990, 0.9001, 0.0180, 0.0324, 0.1049, 0.1004, 0.0080, 0.0064, 0.0040, ... *Two successive zeros behind the decimal point will never disappear.*

- *If we choose* $Z_0 = 2100$*, then we get the sequence* 0.2100, 0.4100, 0.8100, 0.6100, 0.2100, 0.4100, ... *Only after four numbers the sequence starts to repeat itself.*

4.1.2 Linear Congruential Method

Nowadays, most of the random number generators in use are so-called linear congruential generators. They produce a sequence of integers between 0 and $m - 1$ according to the following recursion:

$$Z_i = (aZ_{i-1} + c) \bmod m, \qquad i = 1, 2, 3, \ldots.$$

The initial value Z_0 is called the seed, a is called the multiplier, c the increment and m the modulus. To obtain the desired uniformly (0,1) distributed random numbers we should choose $U_i = Z_i/m$. Remark that U_i can be equal to zero, which will cause some problems if we, for example, are going to generate exponentially distributed random variables (see section 4). A good choice of the values a, c and m is very important. One can prove that a linear congruential generator has a maximal cycle length m if the parameters a, c and m satisfy the following properties:

- c and m are relatively prime;

- if q is a prime divisor of m then $a = 1 \bmod q$;

- if 4 is a divisor of m then $a = 1 \bmod 4$.

Example 4.2

- *If we choose* $(a, c, m) = (1, 5, 13)$ *and take as seed* $Z_0 = 1$*, then we get the sequence of numbers* $1, 6, 11, 3, 8, 0, 5, 10, 2, 7, 12, 4, 9, 1, \ldots$ *which has maximal cycle length of 13.*

- *If we choose* $(a, c, m) = (2, 5, 13)$ *and take as seed* $Z_0 = 1$*, then we get the sequence of numbers* $1, 7, 6, 4, 0, 5, 2, 9, 10, 12, 3, 11, 1, \ldots$ *which has cycle length of 12. If we choose* $Z_0 = 8$*, we get the sequence* $8, 8, 8, \ldots$. *(cycle length 1!)*

4.1.3 Additive Congruential Method

The additive congruential method produces a sequence of integers between 0 and $m-1$ according to the recursion

$$Z_i = (Z_{i-1} + Z_{i-k}) \bmod m, \qquad , i = 1, 2, 3, \ldots,$$

where $k \geq 2$. To obtain the desired uniformly (0,1) distributed random numbers we again should choose $U_i = Z_i/m$. The method can have a maximal cycle length of m^k. However, the method has also some disadvantages. Consider for example the special case $k = 2$. Now, if we take three consecutive numbers U_{i-2}, U_{i-1} and U_i, it will never happen that $U_{i-2} < U_i < U_{i-1}$ or $U_{i-1} < U_i < U_{i-2}$. (For three independent, uniformly (0,1) distributed random variables both of these orderings have probability 1/6).

4.1.4 Tausworthe Generators

The recursions in the linear congruential method and in the additive congruential are special cases of the recursion

$$Z_i = (\sum_{j=1}^{k} a_j Z_{i-j} + c) \bmod m, \qquad i = 1, 2, 3, \ldots.$$

A special situation is obtained if we take $m = 2$ and all $a_j \in \{0, 1\}$. In that case the generator produces a sequence of bits. Such a generator is often called a *Tausworthe generator*, or a *shift register generator*. The sequence $Z_0, Z_1, Z_2, \ldots,$ is now transformed into a sequence $U_0, U_1, U_2, \ldots,$ of uniform random numbers in the following way: Choose an integer $l \leq k$ and put

$$U_n = \sum_{j=0}^{l-1} Z_{nl+j} 2^{-(j+1)}.$$

In other words, the numbers U_n are obtained by splitting up the sequence $Z_0, Z_1, Z_2, \ldots$ into consecutive blocks of length l and then interpreting each block as the digit expansion in base 2 of a number in [0,1).

In practice, often only two of the a_j's are chosen equal to one, so we get

$$Z_i = (Z_{i-h} + Z_{i-k}) \bmod 2.$$

Example 4.3 *If we choose $h = 3$ and $k = 4$ and start with the initial values 1,1,0,1, then we get the following sequence of bits* $1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, \ldots$. *The sequence is periodic with period $2^k - 1 = 15$. If we take $l = 4$, this leads to the random numbers* 13/16, 7/16, 8/16, 9/16, 10/16, 15/16, 1/16, 3/16, ...

4.2 Tests of Random Number Generators

The main properties that a random number generator should have are uniformity and independence. In this section we will describe two tests, the Kolmogorov-Smirnov test and the chi-square test, to compare the distribution of the set of generated numbers with a uniform distribution. Furthermore, we will describe a number of tests that are able to check whether or not the set of generated numbers satisfies independence.

4.2.1 The Kolmogorov-Smirnov test

This test compares the empirical distribution $F_N(x)$ of a set of N generated random numbers with the distribution function $F(x) = x, 0 \leq x \leq 1$, of a uniformly distributed random variable. Here, $F_N(x)$ is defined as the number of observations smaller than or equal to x divided by N, the total number of observations. Let α be the significance level of the test, i.e. α is the probability of rejecting the null hypothesis that the numbers are uniformly distributed on the interval $(0,1)$ given that the null hypothesis is true. Under the null hypothesis $F_N(x)$ will tend to $F(x)$ as N tends to infinity. The Kolmogorov-Smirnov test is based on

$$D = \max | F(x) - F_N(x) | ,$$

the maximal absolute difference between $F_N(x)$ and $F(x)$ over the range of the random variable. Now, if the value of D is greater than some critical value D_α the null hypothesis is rejected. If $D \leq D_\alpha$, we conclude that no difference has been detected between the empirical distribution of the generated numbers and the uniform distribution. The critical value D_α for the specified significance level α of the test and the given sample size N can be found in a table on the Kolmogorov-Smirnov test.

4.2.2 The chi-square test

In the chi-square test, we divide the interval $(0,1)$ in n subintervals of equal length. The test uses the statistic

$$Y = \sum_{i=1}^{n} \frac{(Y_i - E(Y_i))^2}{E(Y_i)},$$

where Y_i is the number of realizations in the i-th subinterval. Clearly, if N is the total number of observations, we have $E(Y_i) = N/n$. Under the null hypothesis of uniformity of the realizations, it can be shown that the distribution of Y is approximately the chi-square distribution with $n-1$ degrees of freedom. So, we compare the value of our statistic Y with the α-percentile of the chi-square distribution with $n-1$ degrees of freedom to conclude whether or not we reject the null hypothesis. Here, α is again the significance level of the test. It is recommended to choose n and N such that $N \geq 50$ and $E(Y_i) \geq 5$.

4.2.3 The serial test

The serial test is a 2-dimensional version of the uniformity test of the previous subsection to test the independence between successive observations. Therefore, we look at N successive tuples $(U_1, U_2), (U_3, U_4), \ldots, (U_{2N-1}, U_{2N})$ of our observations and we count how many observations fall into the N^2 different subsquares of the unit square. We then apply a chi-square test to these data. Of course, we can also formulate a higher-dimension version of this test.

4.2.4 The permutation test

In the permutation test we look at N successive k-tuples of realizations $(U_0, \ldots, U_{k-1})$, $(U_k, \ldots, U_{2k-1})$, …, $(U_{(N-1)k}, \ldots, U_{Nk-1})$. Among the elements in a k-tuple there are $k!$ possible orderings and these orderings are all equally likely. Hence, we can determine the frequencies of the different orderings among the N different k-tupels and apply a chi-squared test to these data.

4.2.5 The run test

In the run test we divide the realizations in blocks, where each block consists of a sequence of increasing numbers followed by a number which is smaller than its predecessor. For example, if the realizations are 1,3,8,6,2,0,7,9,5, then we divide them in the blocks (1,3,8,6), (2,0) and (7,9,5). A block consisting of $k+1$ numbers is called a *run-up of length* k. Under the null hypothesis that we are dealing with independent, uniformly distributed random variables the probability of having a run-up of length k equals $1/k! - 1/(k+1)!$. In a practical implementation of the run test we now observe a large number N of blocks. Furthermore, we choose an integer h and count the number of runs-up of length $0, 1, 2, \ldots, h-1$ and $\geq h$. Then we apply a chi-square test to these data. Of course, we can do a similar test for runs-down.

4.2.6 The gap test

Let J be some fixed subinterval of $(0,1)$. If we have that $U_{n+j} \notin J$ for $0 \leq j \leq k$ and both $U_{n-1} \in J$ and $U_{n+k+1} \in J$, we say that we have a *gap of length* k. Under the null hypothesis that the random numbers U_n are independent and uniformly distributed on (0,1), we have that the gap lengths are geometrically distributed with parameter p, where p is the length of interval J (i.e. $\mathrm{P}(\text{gap of length } k) = p(1-p)^k$). In a practical implementation of the gap test we observe again a large number N of gaps. Furthermore, we choose an integer h and count the number of gaps of length $0, 1, 2, \ldots, h-1$ and $\geq h$. Then we apply a chi-square test to these data.

4.2.7 The serial correlation test

In the serial correlation test we calculate the serial correlation coefficient

$$R = \sum_{j=1}^{N}(U_j - \bar{U})(U_{j+1} - \bar{U}) / \sum_{j=1}^{N}(U_j - \bar{U})^2,$$

where $\bar{U} = \sum_{j=1}^{N} U_j/N$ and U_{N+1} should be replaced by U_1. If the U_j's are really independent the serial correlation coefficient should be close to zero. Hence we reject the null hypothesis that the U_j's are independent if R is too large. The exact distribution of R is unknown, but for large values of N we have that, if the U_j's are independent, then $P(-2/\sqrt{N} \leq R \leq 2/\sqrt{N}) \approx 0.95$. Hence, we reject the hypothesis of independence at the 5% level if $R \notin (-2/\sqrt{N}, 2/\sqrt{N})$.

5 Generating Random Variables

In this section we discuss a number of approaches for generating random variables. First, we discuss the generation of discrete random variables. After that, we will introduce several methods to generate continuous random variables.

5.1 Generating Discrete Random Variables

Let X be a discrete random variable with arbitrary probabilities

$$P(X = x_j) = p_j, \qquad j = 0, 1, 2, \ldots, \qquad \sum_{j=0}^{\infty} p_j = 1.$$

To generate a realization of X, we generate a random number U and set

$$X = \begin{cases} x_0 & \text{if} \quad U \in [0, p_0), \\ x_1 & \text{if} \quad U \in [p_0, p_0 + p_1), \\ \vdots & \\ x_j & \text{if} \quad U \in [\sum_{i=0}^{j-1} p_i, \sum_{i=0}^{j} p_i), \\ \vdots & \end{cases}$$

A nice feature of this method is that we always need exactly one uniformly distributed variable to generate one realization of the desired random variable. Other methods that we will discuss require often more than one uniformly distributed variables to generate one realization of the desired random variable. This observation can be important when you want to use simulation for the comparison of different systems and you want to synchronize the input random streams.

Although this method can be used for the generation of arbitrary discrete random variables, it is not always the most efficient method. Hence we shall give some alternative methods for some special distributions.

5.1.1 The geometric distribution

A random variable X has a geometric distribution with parameter p if

$$P(X = k) = p(1-p)^{k-1}, \quad k = 1, 2, 3, \ldots$$

The geometric distribution can be interpreted as the distribution of the waiting time until the first head in a sequence of independent coin tossing experiments, where p equals the probability of throwing a head. Hence, if $U_1, U_2, \ldots$ are independent, identically distributed uniform (0,1) random variables and X is the index of the first U_i for which $U_i \leq p$, then X is geometrically distributed with parameter p.

Another way to generate a geometric random variable is by using the fact that if U is uniform (0,1), then

$$\left\lceil \frac{\ln (U)}{\ln (1-p)} \right\rceil$$

is geometric with parameter p.

5.1.2 The binomial distribution

A random variable X has a binomial distribution with parameters n and p if

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, 2, \ldots, n.$$

The binomial distribution can be interpreted as the distribution of the number of heads in a sequence of n independent coin tossing experiments, where p equals the probability of throwing a head. Hence, if $U_1, \ldots, U_n$ are independent, identically distributed uniform (0,1) random variables, then

$$X = \sum_{i=1}^{n} 1_{[U_i \le p]}$$

is binomial distributed with parameters n and p. An alternative generation of a binomial distributed random variable can be obtained using one of the following two lemma's.

Lemma 5.1 *Let $G_1, G_2, \ldots$ be independent, identically distributed geometric random variables with parameter p. If X is the smallest integer such that*

$$\sum_{i=1}^{X+1} G_i > n,$$

then X is binomial distributed with parameters n and p.

Lemma 5.2 *Let $E_1, E_2, \ldots$ be independent, identically distributed exponential random variables with parameter 1. If X is the smallest integer such that*

$$\sum_{i=1}^{X+1} \frac{E_i}{n-i+1} > -\ ln\ (1-p),$$

then X is binomial distributed with parameters n and p.

5.1.3 The Poisson distribution

A random variable X has a Poisson distribution with parameter λ if

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}, \quad k = 0, 1, 2, \ldots.$$

An alternative generation of a Poisson distributed random variable can be obtained using one of the following two lemma's.

Lemma 5.3 *Let $E_1, E_2, \ldots$ be independent, identically distributed exponential random variables with parameter 1. If X is the smallest integer such that*

$$\sum_{i=1}^{X+1} E_i > \lambda,$$

then X is Poisson distributed with parameter λ.

Lemma 5.4 *Let $U_1, U_2, \ldots$ be independent, identically distributed uniform (0,1) random variables. If X is the smallest integer such that*

$$\prod_{i=1}^{X+1} U_i < e^{-\lambda},$$

then X is Poisson distributed with parameter λ.

5.2 Generating Continuous Random Variables

We describe four general methods to generate continuous random variables: the inverse transform method, the acceptance-rejection method, the composition method and the convolution method. After that, we pay special attention to the generation of normally distributed random variables.

5.2.1 Inverse Transform Method

The inverse transform method is applicable when the distribution function F of a random variable is continuous and strictly increasing on the domain of the random variable. In that case, the inverse function F^{-1} exists and it is easily seen that, if U in uniformly distributed on $(0,1)$, then $X = F^{-1}(U)$ has distribution function F:

$$P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x).$$

Example 5.5

- $$X = a + (b-a)U$$ *is uniformly distributed on the interval (a, b).*
- $$X = -\frac{1}{\lambda}\, ln\, (1-U)$$ *is exponentially distributed with parameter λ.*
- $$X = \frac{1}{\lambda}(-\, ln\, (1-U)^{1/a}$$ *is Weibull distributed with parameters a and λ.*

If F is not continuous or not strictly increasing, then instead of working with the inverse function we have to introduce the generalized inverse function

$$F^{-1}(u) := \min\{x : F(x) \geq u\}.$$

If we do so, then the general method described in section 5.1 for generating discrete random variables is in fact a special case of the inverse transform method.

Unfortunately, for some often used distribution functions like the normal distribution and the Erlang distribution we do not have a closed form expression for F^{-1}.

5.2.2 Acceptance-Rejection Method

Suppose that we want to generate a random variable X with distribution function F and probability density function f. The acceptance-rejection method requires a function g which majorizes the density f, i.e. $g(x) \geq f(x)$ for all x. Clearly, $g(x)$ will not be a probability density function because $c := \int g(x)dx > 1$. However, if $c < \infty$, then $h(x) = g(x)/c$ is a density. The method applies if we are able to generate easily a random variable Y with density h. The method consists of three steps. First, we generate Y having density h. Next, we generate a uniformly distributed random variable U on the interval $(0, 1)$. Finally, we set $X = Y$ if $U \leq f(Y)/g(Y)$ or, if not, we go back to the first step.

Theorem 5.6 *The random variable X generated by the acceptance-rejection method has probability density function f. The number of iterations of the algorithm that are needed is a geometric random variable with mean c.*

Proof: The probability that a single iteration produces an accepted value which is smaller than x equals

$$P(Y \leq x,\, Y \text{ is accepted}) = \int_{-\infty}^{x} \frac{f(y)}{g(y)} h(y)dy = \frac{1}{c} \int_{-\infty}^{x} f(y)dy$$

and hence each iteration is accepted with probability $1/c$. As each iteration is independent, we see that the number of iterations needed is geometric with mean c. Furthermore,

$$P(X \leq x) = \sum_{n=1}^{\infty} (1 - 1/c)^{n-1} 1/c \int_{-\infty}^{x} f(y)dy = \int_{-\infty}^{x} f(y)dy.$$

5.2.3 Composition Method

The composition method applies when the distribution function F from which we want to sample can be expressed as a convex combination of other distribution functions $F_1, F_2, \ldots$, i.e.

$$F(x) = \sum_{j=1}^{\infty} p_j F_j(x)$$

with $p_j \geq 0$ for all j and $\sum_{j=1}^{\infty} p_j = 1$. The method is useful when it is more easy to generate random variables with distribution function F_j then to generate a random variable with distribution function F. The method consists of two steps. First, we generate a discrete random variable J, such that $P(J = j) = p_j$. Next, given $J = j$, we generate a random variable X with distribution function F_j. It is easily seen that the random variable X has the desired distribution function F. The method can be used, for example, to generate a hyperexponentially distributed random variable.

5.2.4 Convolution Method

For some important distributions, the desired random variable X can be expressed as the sum of other independent random variables, i.e. $X = Y_1 + \cdots + Y_n$, which can be generated more easily than the random variable X itself. The convolution method simply says that you first generate the random variables Y_j and then add them to obtain a realization of the random variable X. We can apply this method, for example, to obtain Erlang distributed random variable. However, when n, the number of phases of the Erlang distribution, is large this may be not the most efficient way to generate an Erlang distributed random variable.

5.2.5 Generating Normal Random Variables

In this section we will show three methods to generate standard normal random variables. The first method is an approximative method using the central limit theorem. The second and third method are exact methods. The second method uses the acceptance-rejection method. The third method is a method developed by Box and Muller which generates two independent standard normal random variables by a transformation of two independent uniform random variables.

Method 1: Using the Central Limit Theorem

An easy, but not very good, approximative method of generating standard normally distributed random variables makes use of the central limit theorem. Recall that the central limit theorem states that for a sequence $Y_1, Y_2, \ldots$ of independent, identically distributed random variables with mean μ and variance σ^2,

$$\frac{\sum_{i=1}^{n} Y_i - n\mu}{\sigma\sqrt{n}} \xrightarrow{d} N(0,1),$$

where $\xrightarrow{d}$ means convergence in distribution and $N(0,1)$ is a standard normal random variable. If we assume that $Y_i = U_i$, i.e. Y_i is uniformly distributed on (0,1), then we have $\mu = 1/2$ and $\sigma^2 = 1/12$. Hence we can take e.g. $\sum_{i=1}^{12} U_i - 6$ as an approximative standard normal random variable.

Method 2: Using the Acceptance-Rejection Method

The probability density function of the absolute value of a standard normal random value equals

$$f(x) = \frac{2}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, \qquad 0 < x < \infty.$$

Now, the function

$$g(x) = \sqrt{2e/\pi}\, e^{-x}$$

majorizes the probability density function f. (Check!) We conclude that we can generate a standard normal random variable in the following way: Generate an exponential random variable Y with parameter 1 and a uniform random variable U on the interval $(0,1)$. Accept Y if $U \leq e^{-(Y-1)^2/2}$, otherwise repeat this procedure. Finally, set $X = Y$ or $X = -Y$, both with probability 1/2.

Method 3: Using the Box-Muller Method

Theorem 5.7 *If U_1 and U_2 are two independent, uniformly $(0,1)$ distributed random variables, then*

$$\begin{aligned} X_1 &= \sqrt{-2\ ln\ U_1}\ cos\ (2\pi U_2) \\ X_2 &= \sqrt{-2\ ln\ U_1}\ sin\ (2\pi U_2) \end{aligned}$$

are two independent, standard normally distributed random variables.

Proof: The joint density function of two independent, standard normal random variables (X_1, X_2) equals

$$f(x_1, x_2) = \frac{1}{2\pi} e^{-\frac{1}{2}(x_1^2+x_2^2)}, \qquad 0 < x_1, x_2 < \infty.$$

Introducing polar coordinates, i.e. $X_1 = R \cos \Phi$ and $X_2 = R \sin \Phi$ we get for the joint density function of R and Φ

$$g(r, \phi) = \frac{1}{2\pi} r e^{-\frac{1}{2}r^2}, \qquad 0 < \phi < 2\pi, \quad 0 < r < \infty.$$

We conclude that R and Φ are independent. Furthermore, it is easy to check that R has the same density as $\sqrt{-2 \ln U_1}$ and Φ has the same density as $2\pi U_2$, where U_1 and U_2 are uniformly $(0,1)$ distributed. Hence the theorem follows.

5.3 Generating Poisson Processes

In practice, we often want to simulate a homogeneous or non-homogeneous Poisson process. The generation of a homogeneous Poisson process can of course be done by using the fact that this process has independent, exponentially distributed interarrival times. For the simulation of a non-homogeneous Poisson process we can use a thinning approach. Suppose that we want to simulate a non-homogeneous Poisson process with arrival rate $\lambda(t)$ at time t until time T. Assume that $\lambda(t) < \lambda$ for all $t \in [0, T]$. We can simulate this process in the following way: we generate a Poisson process with rate λ and accept a possible arrival at time t with probability $\lambda(t)/\lambda$.

Another popular arrival process is a so-called *Markov modulated Poisson process.* For this process, the rate at which, say, customers arrive is modulated by an underlying Markov process. If the underlying Markov process is in state i, then the arrival rate of the Poisson process equals λ_i. Find a way to generate a Markov modulated Poisson process.

6 Output Analysis

In this section we will consider the statistical aspects of a simulation study. We have to realize that doing a simulation is nothing else than doing an experiment (with some realizations for the random elements involved in the simulation). The outcome of a simulation will be, like the outcome of an experiment, an *estimator* of the performance measure of interest. Hence, we also like to say something about the quality of the estimator, e.g. by constructing a *confidence interval* of the performance measure of interest.

First, we will describe how to construct an estimator and a confidence interval for the unknown mean of a random variable, given that we have a number of independent realizations of the random variable. After that, we will show how we can use these results in the output analysis of a simulation study.

6.1 Confidence intervals

Suppose that $X_1, X_2, \ldots, X_n$ are independent realizations of a random variable with unknown mean μ and unknown variance σ^2. Let us define the sample mean

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

and the sample variance

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2.$$

Clearly, we can use the sample mean $\bar{X}$ as an estimator for the unknown mean μ. The question remains how we can construct a confidence interval for μ. To do this, we will use the central limit theorem which states that for sufficiently large n,

$$\frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}}$$

is approximately a standard normal random variable. In fact, this statement remains valid if the unknown standard deviation σ is replaced by the sample standard deviation S. Now, let z_α be the $100(1-\alpha)$-th percentile of the standard normal distribution, i.e.

$$P(Z > z_\alpha) = \alpha$$

where Z is a standard normal random variable. For example, $z_{0.025} = 1.96$. Then

$$P(-z_{\alpha/2} \leq \frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}} \leq z_{\alpha/2}) \approx 1 - \alpha,$$

which is equivalent to

$$P(\bar{X} - z_{\alpha/2}\frac{S}{\sqrt{n}} \leq \mu \leq \bar{X} + z_{\alpha/2}\frac{S}{\sqrt{n}}) \approx 1 - \alpha.$$

In other words, the interval $[\bar{X} - z_{\alpha/2}\frac{S}{\sqrt{n}}, \bar{X} + z_{\alpha/2}\frac{S}{\sqrt{n}}]$ is an approximate $100(1-\alpha)\%$ confidence interval for μ.

Remark 1: If the observations X_i are normally distributed, then the random variable

$$\frac{\sum_{i=1}^{n} X_i - n\mu}{S\sqrt{n}}$$

has *for all* n a Student's t distribution with $n-1$ degrees of freedom. Hence, in this case an exact confidence interval can be obtained by replacing $z_{\alpha/2}$ by $t_{n-1;\alpha/2}$, the $100(1-\alpha/2)$-th percentile of the t distribution with $n-1$ degrees of freedom.
Remark 2: The width of a confidence interval can be reduced either by increasing the number of observations or by decreasing the value of S. In fact, the reduction obtained by halving the value of S is equivalent to the reduction obtained by gathering four times as many observations. Hence, variance reduction techniques are an important issue in simulation. This will be the topic of section 7.

6.2 Short-term simulations

Suppose that we are interested in the performance of a system which starts in some specific initial state and runs for some fixed amount of time. For example, think of the expected total number of customers served in a shop during one working day of, say, eight hours. The way we simulate this system is by using the so-called *independent replications method.* We do N independent simulation runs of the system, each time with a different seed of the random number generator. This gives N independent realizations of a random variable, which expectation is equal to the performance measure of interest. Hence, we can directly use the results of section 6.1 to obtain an estimator and a confidence interval for the performance measure.

6.3 Long-term or steady-state simulations

Now, suppose that we want to have an estimator and a confidence interval for a *long-term or steady-state* performance measure of a system. In general, in this case there arise two problems:

- *Initialization effect*: We are interested in the long-term behaviour of the system and maybe the choice of the initial state of the simulation will influence the quality of our estimate.
- *Correlation*: The random variables involved in a simulation are often positively correlated. This complicates the construction of confidence intervals.

To overcome these problems, there are three different simulation methods developed which will be described in the next three subsections. We will illustrate the three methods using the example of the simulation of the long-term average waiting time EW of customers in a $G/G/1$ queueing system.

6.3.1 Independent Replications Method

Like in the situation of short-term simulations, we do N independent simulation runs of the system, each time with a different seed of the random number generator. In each run we observe the waiting time of, say, K customers. Let $W_{i,j}$ be the waiting time of the j-th customer in the i-th simulation run and

$$W_i = \frac{1}{K}\sum_{j=1}^{K} W_{i,j}$$

be the sample mean of the waiting times in the i-th run. Now, $W_1, \ldots, W_N$ are independent observations of a random variable and, for $K \to \infty$, $E(W_i) \to E(W)$, the performance measure of interest. Hence, we take K large and construct, using the results of section 6.1, an estimator and a confidence interval for $E(W)$. The disadvantage of this method is that in each run we have to deal with the initialization effect. One way of dealing with this problem is to choose K very large and to neglect the initialization effect. Another way of dealing with this problem is to estimate the time until steady-state has been reached (e.g. by using a small simulation) and then to throw away in each run the first part of the data (i.e. we set $W_i = \frac{1}{K-K_0}\sum_{j=K_0+1}^{K} W_{i,j}$, where K_0 is the time at which steady-state has been reached).

6.3.2 Batch Means Method

Instead of doing N independent runs, we do only one simulation run and divide this run into N subruns. The sizes of the subruns are chosen such that in each subrun the waiting times of K customers are observed (i.e. equal-size subruns). Let $W_{i,j}$ be the waiting time of the j-th customer in the i-th subrun and

$$W_i = \frac{1}{K}\sum_{j=1}^{K} W_{i,j}$$

be the sample mean of the waiting times in the i-th subrun. Again, $W_1, \ldots, W_N$ are observations of a random variable and, for $K \to \infty$, $E(W_i) \to E(W)$, the performance measure of interest. However, unfortunately the random variables $W_1, \ldots, W_N$ are not independent! Again, there are two ways to deal with this problem. First, we can take K very large, neglect the dependence between successive observations and construct, using the results of section 6.1, an estimator and a confidence interval for $E(W)$. Alternatively, we can take (part of) the dependence of successive observations into account when we construct a confidence interval for $E(W)$. Let us briefly describe this second method. The classical central limit theorem states that, if $X_1, \ldots, X_n$ are independent, identically distributed random variables with mean μ and variance σ^2, then

$$\frac{\sum_{i=1}^{n} X_i - E(\sum_{i=1}^{n} X_i)}{\sqrt{\text{var}\,(\sum_{i=1}^{n} X_i)}}$$

is approximately standard normal distributed. It turns out that this result remains true if the X_i's are not independent, but *weakly dependent* (i.e. cov $(X_i, X_{i+n}) \to 0$ if $n \to \infty$). For a dependent, stationary process, we have that

$$\begin{aligned}\text{var }(\sum_{i=1}^{n} X_i) &= \sum_{i=1}^{n} \text{var }(X_i) + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \text{cov }(X_i, X_j) \\ &= n \text{ var }(X) + 2\sum_{k=1}^{n-1}(n-k) \text{ cov }(X_1, X_{1+k}).\end{aligned}$$

Now, if we neglect the dependence between the X_i's, then we replace var $(\sum_{i=1}^{n} X_i)$ by n var (X). This is what we did in the first method. If we like to take into account at least part of the dependence between successive observations, then we can replace var $(\sum_{i=1}^{n} X_i)$ by n var $(X) + 2(n-1)$ cov (X_1, X_2) and then construct a confidence interval in a similar way as in section 6.1. The unknown quantities var (X) and cov (X_1, X_2) should of course be estimated from the simulation by

$$\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2, \qquad \text{and} \qquad \frac{1}{n-1}\sum_{i=1}^{n-1}(X_i - \bar{X})(X_{i+1} - \bar{X})$$

respectively.

To summarize, the batch means method has as advantage that we only have to deal once (in the first subrun) with the initialization effect. However, a disadvantage of this method is that outcomes of successive subruns are not independent (they are only approximately independent).

6.3.3 Regenerative Method

In the regenerative method, we again do only one simulation run but instead of dividing this simulation run into N equal-size subruns, we divide it into N *unequal-size* subruns. More specifically, the method makes use of the fact that many stochastic processes are *regenerative*. This means that there are random points in time, so-called regeneration points, such that the process starts afresh probabilistically at these time points. The behaviour of the process after the regeneration point is independent of and probabilistic identical to the behaviour of the process before the regeneration point. Now, in the regenerative method we start a new subrun at time points that the system regenerates itself. In this way, the method provides a solution to the problems of how to start a simulation and how to deal with correlated output.

Examples of regenererative processes

1. $G/G/1$ queue

   In the $G/G/1$ queue the moments that a customer arrives at an empty queue are regeneration points.

2. Two-machine production line

   In the two-machine production line of section 2, there are two types of regeneration points. The first type of regeneration points are the moments that machine 1 is repaired and the buffer is empty. The second type of regeneration points are the moments that machine 1 breaks down and the buffer is completely filled.

3. Periodic review inventory system

   Consider a periodic review inventory system with independent, identically distributed demands per period and instantaneous delivery of replenishment orders. Demand that cannot be satisfied directly from stock is lost. The stock level is controlled by an (s, S) policy. For this system, the regeneration points are the moments just after an arrival of a replenishment order.

We shall illustrate the method using again the simulation of the long term average waiting time in the $G/G/1$ queue.

Let K_i be the number of customers served in the i-th regeneration period (= subrun) and let Y_i be the sum of the waiting times of the customers that are served in the i-th regeneration period, i.e.

$$Y_i = \sum_{j=1}^{K_i} W_{i,j},$$

where $W_{i,j}$ is the waiting time of the j-th customer served in the i-th regeneration period. The outcome of the i-th subrun of our simulation is now the pair of random variables (Y_i, K_i) instead of only Y_i/K_i as was the case in the previous methods. This is due to the fact that the number of customers served in the different subruns is not equal!

Now, the theory of regenerative processes tells us that the long-term average waiting time of customers can be calculated using information over only one regeneration period. More specifically,

$$E(W) = \frac{E(Y_i)}{E(K_i)}.$$

Hence, an estimator for $E(W)$ is given by

$$\bar{W} = \frac{\bar{Y}}{\bar{K}}.$$

For the construction of a confidence interval, we define the random variables

$$V_i = Y_i - E(W)K_i. \qquad , i = 1, 2, \ldots, N.$$

The V_i's are independent, identically distributed random variables with mean 0 and variance

$$\sigma_V^2 = \sigma_Y^2 + E(W)^2\sigma_K^2 - 2E(W)\ \mathrm{Cov}(Y, K).$$

Using the central limit theorem for the V_i's it is straightforward to show (like in section 6.1) that

$$\left[\bar{W} - \frac{z_{\alpha/2}\sigma_V}{\bar{K}\sqrt{N}}, \bar{W} + \frac{z_{\alpha/2}\sigma_V}{\bar{K}\sqrt{N}}\right]$$

is an approximate $(1 - \alpha) \cdot 100\%$ confidence interval for $E(W)$. The unknown quantity σ_V^2 can be estimated by the sample variance

$$s_V^2 = s_Y^2 + \bar{W}^2 s_K^2 - 2\bar{W} s_{Y,K},$$

where

$$s_Y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(Y_i - \bar{Y})^2,$$
$$s_K^2 = \frac{1}{N-1}\sum_{i=1}^{N}(K_i - \bar{K})^2,$$
$$s_{Y,K} = \frac{1}{N-1}\sum_{i=1}^{N}(Y_i - \bar{Y})(K_i - \bar{K}).$$

To summarize, we have that the regenerative method has the advantages that successive observations are really independent and that you do not have to deal with the initialization effect. Disadvantages of the method are that not all systems have regeneration points and that hence the method is not generally applicable. Furthermore, visits of regeneration points can be very rare, especially for large computer- or production systems.
Remark: A similar analysis is possible if we are dealing with a performance measure in continuous time (like queue length) instead of a performance measure in discrete time (like waiting time). For example, if we are interested in the long-term average number of customers in the system of the $G/G/1$ queue, then we could define

$$\begin{aligned} Y_i &= \int_{\alpha_i}^{\alpha_{i+1}} Q_t \, dt, \\ K_i &= \alpha_{i+1} - \alpha_i, \end{aligned}$$

where $\alpha_1, \alpha_2, \alpha_3, \ldots$ are the successive regeneration points and Q_t is the number of customers in the system at time t. Hence, K_i represents the length of the i-th regeneration period and Y_i represents the integrated queue length over the i-th regeneration period. Now, the construction of an estimator and a confidence interval is the same as before.

6.4 Comparison of the three different methods

We will conclude this section with presenting some numerical results which compare the three different methods. For the comparison we use a simulation for the long-term average sojourn time (i.e. waiting time + service time) in an $M_\lambda/M_\mu/1$ queue. For this queue, the long-term sojourn time distribution is exponential with parameter $\mu - \lambda$ (see the course on Queueing Theory (2S520)) and hence we can compare the results of the simulation study also with the exact results obtained by analysis.

Ind. Rep.	$N = 10$, $K = 10000$ 2.025(±0.042)	$N = 10$, $K = 100000$ 1.991(±0.017)	$N = 10$, $K = 1000000$ 1.996(±0.006)
Bat. Mea.	$N = 10$, $K = 10000$ 2.032(±0.037)	$N = 10$, $K = 100000$ 1.991(±0.015)	$N = 10$, $K = 1000000$ 1.997(±0.004)
Regen.	$N = 50000$ 1.993(±0.036)	$N = 500000$ 1.997(±0.012)	$N = 5000000$ 1.997(±0.004)

Table 2: The case $\mu = 1$ and $\lambda = 0.5$.

Ind. Rep.	$N = 10$, $K = 10000$ $9.469(\pm 0.804)$	$N = 10$, $K = 100000$ $10.202(\pm 0.504)$	$N = 10$, $K = 1000000$ $10.076(\pm 0.143)$
Bat. Mea.	$N = 10$, $K = 10000$ $9.588(\pm 1.004)$	$N = 10$, $K = 100000$ $10.232(\pm 0.461)$	$N = 10$, $K = 1000000$ $10.062(\pm 0.116)$
Regen.	$N = 10000$ $9.688(\pm 0.869)$	$N = 100000$ $10.187(\pm 0.394)$	$N = 1000000$ $10.058(\pm 0.121)$

Table 3: The case $\mu = 1$ and $\lambda = 0.9$.

Ind. Rep.	$N = 10$, $K = 10000$ $20.379(\pm 5.223)$	$N = 10$, $K = 100000$ $20.366(\pm 1.575)$	$N = 10$, $K = 1000000$ $20.081(\pm 0.732)$
Bat. Mea.	$N = 10$, $K = 10000$ $20.865(\pm 5.089)$	$N = 10$, $K = 100000$ $20.140(\pm 1.470)$	$N = 10$, $K = 1000000$ $20.138(\pm 0.632)$
Regen.	$N = 5000$ $22.467(\pm 5.137)$	$N = 50000$ $20.162(\pm 1.449)$	$N = 500000$ $20.226(\pm 0.492)$

Table 4: The case $\mu = 1$ and $\lambda = 0.95$.

Results for the case $\mu = 1$ and $\lambda = 0.5$, $\lambda = 0.9$ and $\lambda = 0.95$ respectively are shown. When we used the independent-replications method we have neglected the initialization effect. Similar, when we used the batch means method we neglected the dependence between successive observations.

What we see is that when we take the total (expected) number of customers that are served the same in the three methods (resp. 100000, 1 million and 10 million in the three different columns) then both the quality of the estimator and the width of the confidence interval are approximately the same for the three methods. Furthermore, if we compare the different tables with each other we see that it is much easier to obtain small confidence intervals for lightly loaded systems (table 2) than for heavily loaded systems (table 4).

7 Variance Reduction Techniques

In a simulation study, we are interested in one or more performance measures for some stochastic model. For example, we want to determine the long-run average waiting time, $E(W)$, of customers in a $G/G/1$ queue. To estimate $E(W)$ we can do a number of, say K, independent runs, the i-th one yielding the output random variable W_i with $E(W_i) = E(W)$. After K runs have been performed, an estimator of $E(W)$ is given by $\bar{W} = \sum_{i=1}^{K} W_i/K$. However, if we were able to obtain a different unbiased estimator of $E(W)$ having a smaller variance than $\bar{W}$, we would obtain an estimator with a smaller confidence interval. In this section we will present a number of different methods that one can use to reduce the variance of the estimator $\bar{W}$. We will successively describe the following techniques:

1. Common Random Numbers
2. Antithetic Variables
3. Control Variates
4. Conditioning

The first method is typically used in a simulation study in which we want to compare performance measures of two different systems. All other methods are also useful in the case that we want to simulate a performance measure of only a single system.

7.1 Common Random Numbers

The idea of the method of common random numbers is that if we compare two different systems with some random components it is in general better to evaluate both systems with the same realizations of the random components. Key idea in the method is that if X and Y are two random variables, then

$$\mathrm{Var(X - Y) = Var(X) + Var(Y) - 2Cov(X, Y)}.$$

Hence, in the case that X and Y are *positively* correlated, i.e. $\mathrm{Cov(X, Y)} > 0$, the variance of $X - Y$ will be smaller than in the case that X and Y are independent. In general, the use of common random numbers leads to positive correlation of the outcomes of a simulation of two systems. As a consequence, it is better to use common random numbers instead of independent random numbers when we compare two different systems.

Let us illustrate the method using the following scheduling example. Suppose that a finite number of N jobs has to be processed on two identical machines. The processing times of the jobs are random variables with some common distribution function F. We want to compare the completion time of the last job, $C_{\max}$, under two different policies. In the LPTF policy, we always choose the remaining job with the longest processing time first, in the SPTF policy we choose the remaining job with the shortest processing time first.

In Table 5 and Table 6 we compare for $N = 10$ and $F(x) = 1 - e^{-x}$ the estimators and confidence intervals for $E(C_{\max}^{\mathrm{SPTF}} - C_{\max}^{\mathrm{LPTF}})$ when we do not, resp. do, use common random numbers.

# of runs	mean	st. dev.	95% conf. int.
1000	0.8138	2.5745	[0.645, 0.973]
10000	0.8293	2.4976	[0.780, 0.878]
100000	0.8487	2.4990	[0.833, 0.864]
1000000	0.8398	2.4951	[0.835, 0.845]

Table 5: Estimation of $E(C_{\max}^{\mathrm{SPTF}} - C_{\max}^{\mathrm{LPTF}})$ without using common random numbers

# of runs	mean	st. dev.	95% conf. int.
1000	0.8559	0.5416	[0.822, 0.889]
10000	0.8415	0.5230	[0.831, 0.852]
100000	0.8394	0.5164	[0.836, 0.843]
1000000	0.8391	0.5168	[0.838, 0.840]

Table 6: Estimation of $E(C_{\max}^{\mathrm{SPTF}} - C_{\max}^{\mathrm{LPTF}})$ using common random numbers

We conclude that in this example the use of common random numbers reduces the standard deviation of the estimator and hence also the length of the confidence interval with a factor 5.

When we want to use common random numbers, the problem of synchronization can arise: How can we achieve that the same random numbers are used for the generation of the same random variables in the two systems?

In the previous example, this synchronization problem did not arise. However, to illustrate this problem, consider the following situation. In a $G/G/1$ queueing system the server can work at two different speeds, v_1 and v_2. Aim of the simulation is to obtain an estimator for the difference of the waiting times in the two situations. We want to use the same realizations of the interarrival times and the sizes of the service requests in both systems (the service time is then given by the sizes of the service request divided by the speed of the server). If we use the program of the discrete event simulation of Section 3 of the $G/G/1$ queue, then we get the synchronization problem because the order in which departure and arrival events take place depends on the speed of the server. Hence, also the order in which interarrival times and sizes of service requests are generated depend on the speed of the server.

The synchronization problem can be solved by one of the following two approaches:

1. Use separate random number streams for the different sequences of random variables needed in the simulation.

2. Assure that the random variables are generated in exactly the same order in the two systems.

For the example of the $G/G/1$ queue, the first approach can be realized by using a separate random number stream for the interarrival times and for the service requests. The second approach can be realized by generating the service request of a customer already at the arrival instant of the customer.

7.2 Antithetic Variables

The method of antithetic variables makes use of the fact that if U is uniformly distributed on $(0,1)$ then so is $1-U$ and furthermore U and $1-U$ are negatively correlated. The key idea is that, if W_1 and W_2 are the outcomes of two successive simulation runs, then

$$\mathrm{Var}\left(\frac{W_1+W_2}{2}\right) = \frac{1}{4}\mathrm{Var}(\mathrm{W}_1) + \frac{1}{4}\mathrm{Var}(W_2) + \frac{1}{2}\mathrm{Cov}(W_1, W_2)].$$

Hence, in the case that W_1 and W_2 are *negatively* correlated the variance of $(W_1+W_2)/2$ will be smaller than in the case that W_1 and W_2 are independent.

The question remains how we can achieve that the outcome of two successive simulation runs will be negatively correlated. From the fact that U and $1-U$ are negatively correlated, we may expect that, if we use the random variables $U_1, \dots, U_m$ to compute W_1, the outcome of the first simulation run, and after that $1-U_1, \dots, 1-U_m$ to compute W_2, the outcome of the second simulation run, then also W_1 and W_2 are negatively correlated. Intuition here is that, e.g., in the simulation of the $G/G/1$ queue large realizations of the U_i's corresponding to large service times lead to large waiting times in the first run. Using the antithetic variables, this gives small realizations of the $1-U_i$'s corresponding to small service times and hence leading to small waiting times in the second run.

We illustrate the method of antithetic variables using the scheduling example of the previous subsection.

# of runs	mean	st. dev.	95% conf. int.
1000	5.0457	1.6201	[4.945, 5.146]
10000	5.0400	1.6020	[5.009, 5.071]
100000	5.0487	1.5997	[5.039, 5.059]
1000000	5.0559	1.5980	[5.053, 5.059]

Table 7: Estimation of $E(C_{\max}^{\mathrm{LPTF}})$ without using antithetic variables

# of pairs	mean	st. dev.	95% conf. int.
500	5.0711	0.7216	[5.008, 5.134]
5000	5.0497	0.6916	[5.030, 5.069]
50000	5.0546	0.6858	[5.049, 5.061]
500000	5.0546	0.6844	[5.053, 5.056]

Table 8: Estimation of $E(C_{\max}^{\mathrm{LPTF}})$ using antithetic variables

In Table 7 and Table 8 we compare, again for $N=10$ and $F(x)=1-e^{-x}$, the estimators and confidence intervals for $E(C_{\max}^{\mathrm{LPTF}})$ when we do not, resp. do, use antithetic variables. So, for example, we compare the results for 1000 independent runs with the results for 1000 runs consisting of 500 pairs of 2 runs where the second run of each pair uses antithetic variables. We conclude that in this example the use of antithetic variables reduces the length of the confidence interval with a factor 1.5.

Finally, remark that, like in the method of common random numbers, the synchronization problem can arise. Furthermore, it should be noted that the method is easier to implement if all random variables are generated using the inversion transform technique (only one uniform random number is needed for the realization of one random variable) than if we use, e.g., the rejection method to generate random variables (a random number of uniform random numbers are needed for the realization of one random number).

7.3 Control Variates

The method of control variates is based on the following idea. Suppose that we want to estimate some unknown performance measure $E(X)$ by doing K independent simulation runs, the i-th one yielding the output random variable X_i with $E(X_i) = E(X)$. An unbiased estimator for $E(X)$ is given by $\bar{X} = (\sum_{i=1}^{K} X_i)/K$. However, assume that at the same time we are able to simulate a related output variable Y_i, with $E(Y_i) = E(Y)$, where $E(Y)$ is known. If we denote by $\bar{Y} = (\sum_{i=1}^{K} Y_i)/K$, then, for any constant c, the quantity $\bar{X} + c(\bar{Y} - E(Y))$ is also an unbiased estimator of $E(X)$. Furthermore, from the formula

$$\mathrm{Var}(\bar{X} + c(\bar{Y} - E(Y))) = \mathrm{Var}(\bar{X}) + c^2\mathrm{Var}(\bar{Y}) + 2c\mathrm{Cov}(\bar{X}, \bar{Y}).$$

it is easy to deduce that $\mathrm{Var}(\bar{X} + c(\bar{Y} - E(Y)))$ is minimized if we take $c = c^*$, where

$$c^* = -\frac{\mathrm{Cov}(\bar{X}, \bar{Y})}{\mathrm{Var}(\bar{Y})}.$$

Unfortunately, the quantities $\mathrm{Cov}(\bar{X}, \bar{Y})$ and $\mathrm{Var}(\bar{Y})$ are usually not known beforehand and must be estimated from the simulated data. The quantity $\bar{Y}$ is called a *control variate* for the estimator $\bar{X}$. To see why the method works, suppose that $\bar{X}$ and $\bar{Y}$ are positively correlated. If a simulation results in a large value of $\bar{Y}$ (i.e. $\bar{Y}$ larger than its mean $E(Y)$) then probably also $\bar{X}$ is larger than its mean $E(X)$ and so we correct for this by lowering the value of the estimator $\bar{X}$. A similar argument holds when $\bar{X}$ and $\bar{Y}$ are negatively correlated.

In the simulation of the production line of section 2, a natural control variate would be the long-term average production rate for the line with zero buffers.

7.4 Conditioning

The method of conditioning is based on the following two formulas. If X and Y are two arbitrary random variables, then $E(X) = E(E(X|Y))$ and $\mathrm{Var}(X) = E(\mathrm{Var}(X|Y)) + \mathrm{Var}(E(X|Y)) \geq \mathrm{Var}(E(X|Y))$. Hence, we conclude that the random variable $E(X|Y)$ has the same mean as and a smaller variance than the random variable X.

How can we use these results to reduce the variance in a simulation? Let $E(X)$ be the performance measure that we want to estimate. If Y is a random variable such that $E(X|Y = y)$ is known, then the above formulas tell us that we can better simulate Y and use $E(X|Y)$ than that we directly simulate X.

The method is illustrated using the example of an $M_\lambda/M_\mu/1/N$ queueing model in which customers who find upon arrival N other customers in the system are lost. The performance measure that we want to simulate is $E(X)$, the expected number of lost customers at some fixed time t. A direct simulation would consist of K simulation runs until time t. Denoting by X_i the number of lost customers in run i, then $\bar{X} = (\sum_{i=1}^{K} X_i)/K$ is an unbiased estimator of $E(X)$. However, we can reduce the variance of the estimator in the following way. Let

Y_i be the total amount of time in the interval $(0, t)$ that there are N customers in the system in the i-th simulation run. Since customers arrive according to a Poisson process with rate λ, it follows that $E(X|Y_i) = \lambda Y_i$. Hence an improved estimator would be $\lambda \bar{Y}$, where $\bar{Y} = (\sum_{i=1}^{K} Y_i)/K$.

# of runs	mean	st. dev.	95% conf. int.
10	33.10	10.06	[26.86, 39.33]
100	34.09	9.21	[32.28, 35.90]
1000	33.60	8.88	[33.05, 34.15]

Table 9: Estimation of $E(X)$ without using conditioning

# of runs	mean	st. dev.	95% conf. int.
10	33.60	7.16	[29.17, 38.05]
100	33.82	6.91	[32.46, 35.18]
1000	33.36	6.57	[32.95, 33.76]

Table 10: Estimation of $E(X)$ using conditioning

In Table 9 and Table 10 we compare, for $\lambda = 0.5$, $\mu = 1$, $N = 3$ and $t = 1000$, the estimators and confidence intervals for $E(X)$ when we do not, resp. do, use conditioning. We conclude that in this example the use of conditioning reduces the standard deviation of the estimator and hence also the length of the confidence interval with a factor 1.3.

8 Rare Event Simulation

For the simulation of rare events, direct simulations are usually very slow and take a lot of computer time. In such situations one often uses a technique called *importance sampling*. The idea of importance sampling is to make rare events more frequently by changing the probability distributions underlying the system and by performing simulations of the changed system. Results for the original system are obtained by transforming the results back to the original system using so-called *likelihood ratios*. In this section, we will illustrate the use of importance sampling in the case of a rare event simulation.

Consider the following problem. We want to have an estimator for the probability, p, that during an arbitrary busy period in an $M/M/1$ queue there have been, at some point in time, at least M customers in the system. Typically, when M is large and the arrival intensity λ is much smaller than the service intensity μ, this is an example of a rare event. Below, we shall describe both a direct simulation and an indirect simulation using importance sampling for this problem.

8.1 Direct simulation

In a direct simulation for this problem it seems natural to use the regenerative method. So, we simulate K busy periods of the $M/M/1$ queue and define

$$V_k = \begin{cases} 1, & \text{if } \sharp \text{ of customers reaches level } M \text{ in period } k, \\ 0, & \text{if } \sharp \text{ of customers does not reach level } M \text{ in period } k. \end{cases}$$

Remark that we can stop the simulation of a busy period if the number of customers in the system reaches either level 0 or level M. As an estimator for the probability p we now can use

$$\hat{p} = \frac{1}{K} \sum_{k=1}^{K} V_k.$$

Suppose that we want to construct a 95% confidence interval for p and that the width of the confidence interval must be smaller than 10% of the actual value of p, i.e. we want that

$$P\left(p \in [\hat{p} - \frac{1}{20}p, \hat{p} + \frac{1}{20}p]\right) \approx 0.95.$$

How many regeneration periods do we have to simulate? To answer this question, observe that the random variables V_k are independent, identically distributed with mean $E(V) = p$ and variance $\sigma_V^2 = p(1-p) \approx p$ for p small. Hence, a 95% confidence interval for p is given by $[\hat{p} - \frac{1.96\sigma_V}{\sqrt{K}}, \hat{p} + \frac{1.96\sigma_V}{\sqrt{K}}]$ and to obtain the required width of the confidence interval we have to choose

$$K \approx \frac{1600}{p}.$$

8.2 Indirect simulation

In an indirect simulation of the problem we consider the $M/M/1$ queue with arrival rate μ and service intensity λ, i.e., the system obtained by interchanging the arrival rate and service rate of the original system.

Again, we simulate K busy periods for this interchanged $M/M/1$ queue and define

$$\tilde{V}_k = \begin{cases} 1, & \text{if } \sharp \text{ of customers reaches level } M \text{ in period } k \text{ in the new system,} \\ 0, & \text{if } \sharp \text{ of customers does not reach level } M \text{ in period } k \text{ in the new system.} \end{cases}$$

As an estimator for the probability p we now use

$$\hat{p} = \frac{1}{K} \sum_{k=1}^{K} L_k \tilde{V}_k,$$

where L_k is the *likelihood ratio* between the original and the interchanged system. This likelikood ratio relates the probability of reaching level M in the interchanged system with the same quantity in the original system. The probability of reaching level M in the interchanged system equals

$$\sum_{j=0}^{\infty} C_j \cdot \left(\frac{\lambda}{\lambda+\mu}\right)^j \left(\frac{\mu}{\lambda+\mu}\right)^{j+M-1}$$

where C_j is equal to the number of different paths of a simple random walk starting at time 0 in state 1 and ending at time $2j+M-1$ in state M without reaching state 0 or M before time $2j+M-1$. Similarly, the probability of reaching level M in the original system equals

$$\sum_{j=0}^{\infty} C_j \cdot \left(\frac{\mu}{\lambda+\mu}\right)^j \left(\frac{\lambda}{\lambda+\mu}\right)^{j+M-1}.$$

Hence, we conclude that the likelihood ratio for this problem satisfies

$$L_k = \left(\frac{\lambda}{\mu}\right)^{M-1}.$$

Again, the random variables $L_k \tilde{V}_k$ are independent, identically distributed with mean p, but now with variance

$$\sigma^2_{L_k \tilde{V}_k} = \left(\frac{\lambda}{\mu}\right)^{M-1} p - p^2.$$

The number of regeneration periods that we have to simulate using an indirect simulation in order to get a 95% confidence interval for p with width only 10% of the value of p is given by

$$K = \frac{1600\sigma^2_{L_k \tilde{V}_k}}{p^2} \approx 1600 \left(\left(\frac{\lambda}{\mu}\right)^{M-1} \frac{1}{p} - 1\right).$$

8.3 Comparison direct and indirect simulation

In order to be able to show the difference in simulation time between the direct and indirect simulation we make use of the fact that we also know the exact value of p for this problem. Let P_i denote the probability that we reach level M before level 0 if we start with i customers in the system. Clearly, we are interested in $p = P_1$. The P_i's satisfy the difference equation

$$P_i = \frac{\lambda}{\lambda + \mu} P_{i+1} + \frac{\mu}{\lambda + \mu} P_{i-1},$$

which is equivalent to

$$\begin{aligned} P_{i+1} - P_i &= \left(\frac{\mu}{\lambda}\right)(P_i - P_{i-1}) \\ &= \dots \\ &= \left(\frac{\mu}{\lambda}\right)^i (P_1 - P_0). \end{aligned}$$

From the boundary conditions $P_0 = 0$ and $P_M = 1$ we now conclude

$$1 = \sum_{i=0}^{M-1} (P_{i+1} - P_i) = \sum_{i=0}^{M-1} \left(\frac{\mu}{\lambda}\right)^i (P_1 - P_0) = \frac{1 - \left(\frac{\mu}{\lambda}\right)^M}{1 - \frac{\mu}{\lambda}} P_1,$$

and hence

$$p = \frac{\frac{\mu}{\lambda} - 1}{\left(\frac{\mu}{\lambda}\right)^M - 1}.$$

We conclude that for the indirect simulation the number of regeneration periods that we have to simulate is much smaller than for the direct simulation. For example, in the case $\lambda = 1/3$, $\mu = 2/3$ and $M = 21$ we have $p = 4.77 \cdot 10^{-7}$, $K_{direct} = 3.36 \cdot 10^9$ and $K_{indirect} = 1.6 \cdot 10^3$.

References

1. Banks, J. and Carson, J.S., *Discrete-Event System Simulation*, Prentice-Hall, Englewood Cliffs, NJ (1984).

2. Bratley, P., Fox, B.L. and Schrage, L.E., *A Guide to Simulation*, Springer-Verlag, New York, 2nd edition (1987).

3. Crane, M. and Lemoine, A.J., *An Introduction to the Regenerative Method for Simulation Analysis*, Springer-Verlag, New York (1977).

4. Devroye, L., *Non-uniform Random Variate Generation*, Springer-Verlag, New York (1986).

5. Knuth, D.E. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, Addison-Wesley, Reading, MA (1981).

6. Law, A.M. and Kelton, W.D. *Simulation Modelling and Analysis*, McGraw-Hill, New York, 2nd edition (1991).

7. L'Ecuyer, P. Efficient and portable combined random number generators, *Communications of the ACM*, 31 (6), 742-774 (1988).

8. Parekh, S. and Walrand, J., A quick simulation method for excessive backlogs in networks of queues. *IEEE Transactions on Automatic Control*, AC-34:54-56 (1989).

9. Ross, S.M., *Simulation*, Academic Press, 2nd edition (1996).