Chapter 6 Experimentation and Output Analysis

6.1 Overview of the Issue

In this chapter we explore the activities of experimentation and output analysis, which are both central to the success of any modelling and simulation project. In other words, we examine the process of correctly formulating and carrying out goal-directed experiments with the simulation program and then extracting meaningful information from the data acquired via its output variables. The underlying complexity here arises from the uncertainty that is superimposed on all variables in any DEDS model by the random nature of input variables and by the random behaviour of 'internal' processes (e.g., message service time at the nodes of a communications network or failure characteristics of machines in a manufacturing plant). As we have previously noted, these random phenomena represent one of the essential differences between models arising from the DEDS context and those arising from the realm of continuous-time dynamic systems.

A simulation program provides an observation window onto a variety of random phenomena that unfold as a result of the model's execution. Each can be linked to a random variable and some of these random variables are of special interest from the perspective of the project goals.

The notion of output variables was explored in the discussions of both Chapters 2 and 4 where it was stressed that any model necessarily has one or more such variables associated with it. This follows simply because they serve as the conduits for the data that are essential for the resolution of the project's goals. In these earlier discussions we introduced two categories of output variable called point-set output variables (PSOVs) and derived scalar output variables (DSOVs).

There are two types of variable in the PSOV category; namely, time variables and sample variables. These share a common means for delivering data from any particular execution of the simulation program, namely, through the accumulation of a finite set of (possibly) time-indexed values. However, the specific values in such a set are rarely of interest. Instead what is of interest is typically some property of these accumulated data, for example, minimum, maximum, average, or number (a count of the number of values in the set). Such a value is computed and assigned to a designated scalar variable. Such variables are necessarily random variables and they fall into the category of DSOVs. Our interest throughout this chapter is primarily with DSOVs and for convenience we refer to these simply as output variables.

Let's consider some examples of DSOVs that might arise at the level of the ABCmod conceptual modelling framework as discussed in Chapter 4. The list below demonstrates the most fundamental feature of any such variable; namely that it always has a 'definition', that is, a meaning in terms of the behaviours that are represented within the conceptual model. Although this may appear obvious, it is a feature that must be unambiguously documented in the statement of project goals.

- An output variable Y_A which represents the proportion of customers that waited for more than five minutes for service at Kojo's Kitchen in the food court
- An output variable Y_B, which represents the average time spent waiting for tugboat service by the tankers that pass through an ocean port model
- An output variable Y_C, which represents the maximum number of messages in the input buffer of a particular node P of a communications network, over a 24 hour period
- An output variable Y_D, which represents the portion of time that all the attendants in a full-service gas station are busy, over the course of a business day

Some details for these four variables are presented in Table 6.1 in terms of the notions in our ABCmod framework as discussed in Chapter 4. In particular the table shows how a value might be established for each of these variables by carrying out an operation on some underlying output set of data values.

TABLE 6.1. Elaboration of representative output variables.

Output Variable (DSOV)	SUI Context	Underlying Time Variable/Sample Variable (PSOV)	Output Set (Trajectory or Sample Set)	Operator on Output Set
Y_A	Kojo's Kitchen	Attribute Customer.WaitTime of the Customer consumer entity class that represents the customers flowing through Kojo's Kitchen	PHI[C.Customer.WaitTime]	PropGT
Y_B	Ocean port	Attribute Tanker.TotalWait of the Tanker consumer entity class is used to accumulate the time spent by each tanker instance waiting for tugboat service	PHI[C.Tanker.TotalWait]	AVG
Y_C	Communication network	Attribute Pnode.N of the queue representing the input buffer of the particular node of interest	TRJ[A.Pnode.N]	MAX
Y_D	Full-service gas station	Attribute Attend.AllBusy of the resource entity associated with the attendants whose value is set to 1 when all attendants are busy and is zero otherwise	TRJ[R.Attend.AllBusy]	AVG

Any particular output variable listed in Table 6.1 acquires a value as a consequence of the execution of its respective simulation program (i.e., as a consequence of a ‘simulation run’ or simply a ‘run’). However, this value is not a direct outcome of the experiment but rather is obtained by carrying out an operation on a set of data values as illustrated in Figure 5.1. In the case of Y_A the data set is the sample set PHI[C.Customer.WaitTime] which is populated by values of the sample variable Customer.WaitTime (an attribute of the consumer entity class called Customer). Each Customer instance that passes through Kojo’s Kitchen contributes a value to PHI[C.Customer.WaitTime], and for any particular simulation run, the value acquired by Y_A is obtained as PropGT(5,PHI(C.Customer.WaitTime)). Here *PropGT(Val,SampleSet)* is a user-defined module specified in the ABCmod conceptual model for the Kojo's Kitchen project in Chapter 5 (see Table 5.10).

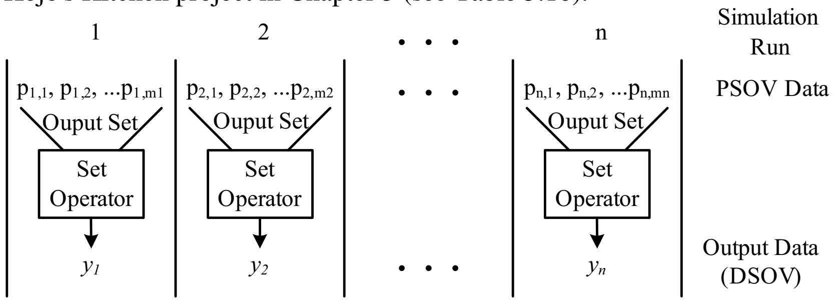

1
2
n
Simulation Run
$p_{1,1}, p_{1,2}, \ldots p_{1,m1}$
$p_{2,1}, p_{2,2}, \ldots p_{2,m2}$
$p_{n,1}, p_{n,2}, \ldots p_{n,mn}$
PSOV Data
Ouput Set
Set Operator
y_1
y_2
y_n
Output Data (DSOV)

FIGURE 6.1. Generation of data from multiple simulation runs.

As previously observed, any DSOV is a random variable. There are certainly circumstances where interest in a random variable can focus simply on a particular value (e.g., the sum of the dots showing on a pair of dice when the dice are thrown during a game of chance). However, the value of a DSOV acquired from a single simulation run generally falls far short of providing useful information from the perspective of the requirements of project goals. The information that is needed typically relates to the values of the parameters of the distribution of the DSOV (e.g. mean, variance) and meaningful estimates of such parameters can only be formulated from results obtained from multiple runs that have been organised to yield independent observations.[1]

A frequent misunderstanding occurs when the project goals require a mean value estimate that appears to coincide directly with some DSOV

[1] Strictly speaking, this is not entirely correct. In the context of a steady-state study, there does exist an approach called the method of batch means where all required data are generated from a single long simulation run. A brief discussion can be found in Section 6.3.2.

(for definiteness, let's call it Y) that is defined as an average. Consider, for example, our port project where the mean waiting time of tankers is required. A particular simulation run will yield a sample set whose members are the waiting times of the tankers that passed through the port during that run. The average of the values in this sample set (which we denote by $\hat{y}$) would represent a single observation of the DSOV, Y.

One might be tempted here to use $\hat{y}$ as an estimate of the mean value that we seek (namely, the mean waiting time of tankers that pass through the port). However there generally is a correlation among the values because, for example, a long wait by some tanker will likely result in long waits by succeeding tankers thereby introducing a bias in the collected data. This circumstance precludes the use of the standard methods of statistics which depend on the assumption of independence, for example. for the determination of the confidence interval that we discuss below. It is for this reason that suitably replicated simulation runs (or other equivalent approaches) are required which will generate a collection of independent observations of Y from which the desired mean value estimate and a confidence interval can be formulated. This is achieved by proper management of the seeds used in the random variate generation procedures that are embedded in the simulation model.

For the most part, our interest in experimentation focuses on the mean values of designated output variables within the simulation program. It needs to be recognised, however, that the determination of an exact value for these is rarely feasible. Experiments with the simulation program can, at best, deliver the data from which an estimate of the mean (called a *point estimate*) can be formulated together with an assessment of the quality of the estimate (i.e., a *confidence interval*). Guidance for determining what experiments need to be carried out and how the acquired data need to be handled in order to obtain credible estimates are provided by some of the fundamental results from probability theory. An overview of these can be found in the latter sections of Annex 1. The topic is explored in the discussions below.

6.2 Bounded Horizon Studies

We now consider the basic problem of analysing the values acquired by an output variable in the simulation program in the case of a bounded horizon study. From a collection of values acquired from n simulation runs, we determine a point estimate of the mean (i.e., a single number whose validity has some credible basis) and then formulate an interval in which the point estimate lies with a prescribed degree of confidence.

The considerations that follow rely heavily on the results presented in Annex A1, in particular the results in Sections A1.5 through A1.7.

6.2.1 Point Estimates

Suppose *Y* is an output variable (i.e. DSOV) of the simulation program **M** and we seek an estimate of the mean of the distribution of *Y,* namely an estimate of μ = E[*Y*]. The fundamental result from probability theory upon which we rely is the strong law of large numbers (see Section A1.5 of Annex A1). The interpretation in our context is that $\overline{Y}(n)$ approaches μ as *n* becomes large where:

$$\overline{Y}(n) = \frac{1}{n}\sum_{k=1}^{n} Y_k$$

and we regard the Y_ks as surrogate random variables for *Y* that are associated with a sequence of *n* correctly replicated experiments with **M** (the variable Y_k is associated with experiment *k*). All Y_k's have the same distribution as *Y* because they reflect the same process (namely the simulation program **M**). Furthermore, because they are linked to a sequence of correctly replicated experiments, we can assume that the Y_k's are independent. Hence the Y_k can be taken to be a set of independent identically distributed (IID) random variables.

Correctly replicated simulation runs are a key requirement in formulating the estimate that we seek. The implication here is that there is appropriate management of the seeds used to initialise the various random number generators from run to run to create a meaningful set of independent and identically distributed observations (initial conditions, however, must remain invariant except when their values are part of the random envelope).

On the basis of the above, a point estimate of μ can be obtained in the following way.

1. Choose a suitable value for *n*, the number of replications (in principle, *n* needs to be large, but a value in the order of 30 is generally satisfactory).
2. Collect the *n* observed values y_1, y_2, . . . , y_n for the random variables Y_k, *k* = 1, 2, . . . , *n*, that result from *n* replicated simulation runs of the simulation program **M.**
3. Compute:

$$\overline{y}(n) = \frac{1}{n}\sum_{k=1}^{n} y_k \ .$$

The numerical value that results for $\overline{y}(n)$ is then taken to be the point estimate for μ = E[*Y*] that we seek.

6.2.2 Interval Estimation

We now expand our task by undertaking to find a suitable value for the number of replications *n* which will ensure a particular 'quality' for the estimate $\bar{y}(n)$. We know from Section A1.7 that an interval (called the confidence interval) can be established within which μ falls with a prescribed level of confidence. This interval has the form $[y(\bar{n}) - \zeta(n), y(\bar{n}) + \zeta(n)]$ where $\zeta(n) = (t_{n\text{-}1,a}\ s(n)/\sqrt{n})$, and $s(n)$ is an estimate of the standard deviation. We call $\zeta(n)$ the confidence interval half length. Its value is clearly dependent on the Student t-distribution value $t_{n\text{-}1,a}$.

The quality criterion we introduce is the requirement that, with confidence 100*C*% ($0 < C < 1$), $|\bar{y}(n) - \mu| < \zeta^*$. In other words, we want to ensure that (with a prescribed level of confidence) the interval half length $\zeta(n)$ is less than a specific value denoted by ζ^*. A possible choice for ζ^* is $r\,\bar{y}(n)$ where *r* is a value chosen in the range (0, 1). With this choice, the maximum displacement of the estimate from μ is proportional to the value of the estimate itself. Note that our quality measure can be interpreted as

$$\frac{\zeta(n)}{\bar{y}(n)} < r$$

The procedure is outlined below and is based entirely on the discussion in Section A1.7 of Annex 1 (Equation (A1.36) has particular relevance).

1. Choose values for *r* and for the confidence level parameter *C*, and as well, an initial value for *n* that is not smaller than 20.
2. Collect the *n* observed values $y_1, y_2, \ldots, y_n$ for the random variables Y_k, $k = 1, 2, \ldots, n$, that result from *n* replicated simulation runs of the simulation program, **M**.
3. From tabulated data for the Student *t*-distribution, determine $t_{n\text{-}1,a}$ where $a = (1 - C)/2$.
4. Compute:

$$\bar{y}(n) = \frac{1}{n}\sum_{k=1}^{n} y_k$$

$$s^2(n) = \frac{\sum_{k=1}^{n}(y_k - \bar{y}(n))^2}{n-1} \qquad (6.1)$$

$$\zeta(n) = \frac{t_{n-1,\alpha}\, s(n)}{\sqrt{n}}$$

5. If $\zeta(n) < \zeta^* = r\bar{y}(n)$ (or $\zeta(n) / \bar{y}(n) < r$) then accept $\bar{y}(n)$ as the point estimate of μ and end the procedure, otherwise continue to Step 6.
6. Choose Δn no smaller than 3 and collect additional observations $y_{n+1}, y_{n+2}, \ldots, y_{n+\Delta_n}$ through a further Δn replications, replace n with $n + \Delta n$, and repeat from Step 3.

6.2.3 Output Analysis for Kojo's Kitchen Project

This section examines how the analysis techniques described in Section 6.2.2 can be applied to achieving the goal set out in the Kojo's Kitchen project. Recall that the goal set out in Chapter 5 was to investigate the impact on the output variable *PropLongWait* (i.e., the proportion of customers waiting longer than five minutes) of adding an additional employee. The Java event-scheduling simulation program presented in Section 5.4.2 is used to experiment with the simulation model and generate data for analysis. The collected data are analysed using a number of useful data analysis tools available in Microsoft Excel.

Figure 6.2 shows the Java method used to carry out multiple simulation runs with the Kojo's Kitchen simulation model (see Section 5.3). Note the following.

- The first part of the method generates the random seeds used in all the simulation runs. The CERN Java package offers a Class *Random-SeedGenerator* that provides the means to generate appropriate (uncorrelated) random seeds. This ensures that the different simulation runs provide independent values for the *PropLongWait* output variable. Note also that the seeds are stored in an array of *Seeds* objects. Also note that four seeds make up a *Seeds*object, one for each random number generator used in the simulation program. Thus they can be reused when executing the runs for the alternate case. This is important for comparing the two cases as discussed in Section 6.4.
- For each simulation run, a new *KojoKitchen* object is created using the Class constructor. The constructor provides the data necessary for the simulation run, that is, specifies the observation interval (the first two arguments specify the right- and left-hand boundaries of the interval), a value for the *empSchedCase* parameter (either Case1 or Case2), and finally a *Seeds* object to seed the random number generators. Recall that Case 1 is the base case and Case 2 is the situation where the third employee is hired during busy periods.
- After each run, the value generated for *PropLongWait* is displayed along with the run number. The output of the running program can be redirected into a file and subsequently loaded into an Excel worksheet for analysis.

```
class KojoExperiment1
{
  public static void main(String[] args)
  {
     final int NUMRUNS = 10000;
     int i;
     Seeds[] sds = new Seeds[NUMRUNS];
     KojoKitchen kojo;  // simulation program
     double propLongWait;

     // Get a set of uncorrelated seeds
     RandomSeedGenerator rsg = new RandomSeedGenerator();
     for(i=0 ; i<NUMRUNS ; i++)
         sds[i] = new Seeds(rsg.nextSeed(),rsg.nextSeed(),
                            rsg.nextSeed(),rsg.nextSeed());

     // Loop for NUMRUN simulation runs for each case
     // Case 1
     System.out.println("Case 1 - no additional employee");
     for(i=0 ; i < NUMRUNS ; i++)
     {
       kojo = new KojoKitchen(0.0,660.0,KojoKitchen.Case1,sds[i]);
       kojo.runSimulation();
       propLongWait = kojo.getPropGT(5);
       System.out.println((i+1)+", "+propLongWait);
     }
     // Case 2
     System.out.println("Case 2 - add employee during busy times");
     for(i=0 ; i < NUMRUNS ; i++)
     {
       kojo = new KojoKitchen(0.0,660.0, KojoKitchen.Case2,sds[i]);
       kojo.runSimulation();
       propLongWait = kojo.getPropGT(5);
       System.out.println((i+1)+", "+propLongWait);
     }
  }
}
```

FIGURE 6.2. Java method for experimentation with the Kojo's Kitchen simulation program.

Table 6.2 shows the values for *propLongWait* for the first 20 simulation runs for each of the two cases. The values for the point estimate ($\overline{y}(n)$), the standard deviation ($s(n)$), and the confidence interval half length ($\zeta(n)$) are shown in the table, with n = 20. These were computed by using Equation (6.1) with a 90% confidence level (i.e C=0.9). The left boundary $\overline{y}(n)-\zeta(n)$ and right boundary $\overline{y}(n)+\zeta(n)$ of the confidence interval are given by CI Min and CI Max, respectively.

TABLE 6.2. Analysis of generated data from the first 20 simulation runs.

Run	**Case 1**	**Case 2**
1	0.634	0.263
2	0.595	0.209
3	0.256	0.067
4	0.532	0.335
5	0.282	0.049
6	0.649	0.278
7	0.458	0.024
8	0.515	0.158
9	0.618	0.062
10	0.667	0.348
11	0.483	0.238
12	0.524	0.107
13	0.663	0.447
14	0.235	0.053
15	0.404	0.051
16	0.472	0.112
17	0.425	0.094
18	0.565	0.124
19	0.392	0.048
20	0.381	0.123
$\overline{y}(n)$	0.487	0.160
$s(n)$	0.134	0.121
$\zeta(n)$	0.052	0.047
CI Min	0.436	0.113
CI Max	0.539	0.206

Table 6.3 shows for each of the two cases the values of $\bar{y}(n)$, $s(n)$, and $\zeta(n)$ (computed using Equation (6.1)) as well as the boundaries of the confidence interval (CI Min and CI Max) and the ratio $\zeta(n)/\bar{y}(n)$ when n (the number of simulation runs) is increased. Note from the rightmost column how the ratio $\zeta(n)/\bar{y}(n)$ decreases as n increases. This is mainly a consequence of a decreasing value for the confidence interval half length $\zeta(n)$.

Observe that for Case 1, with 20 runs the half length of the confidence interval is essentially 10% of the point estimate (see rightmost column where the value is 0.106). However with 20 runs, the interval half length in Case 2 is almost 30% of the point estimate (value in rightmost column is 0.293). For Case 2, 100 runs are required to achieve a comparable confidence interval as Case 1.

TABLE 6.3. Impact of number of runs on the confidence interval.

Case 1						
n	$\bar{y}(n)$	***s(n)***	$\zeta(n)$	**CI Min**	**CI Max**	$\zeta(n)/\bar{y}(n)$
20	0.487	0.134	0.052	0.436	0.539	0.106
30	0.503	0.125	0.039	0.464	0.542	0.077
40	0.502	0.119	0.032	0.471	0.534	0.063
60	0.504	0.116	0.025	0.479	0.529	0.049
80	0.499	0.129	0.024	0.475	0.523	0.048
100	0.503	0.132	0.022	0.481	0.524	0.044
1000	0.510	0.120	0.006	0.504	0.517	0.012
10000	0.508	0.126	0.002	0.506	0.510	0.004

Case 2						
n	$\bar{y}(n)$	***s(n)***	$\zeta(n)$	**CI Min**	**CI Max**	$\zeta(n)/\bar{y}(n)$
20	0.160	0.121	0.047	0.113	0.206	0.293
30	0.192	0.124	0.039	0.153	0.230	0.201
40	0.193	0.119	0.032	0.161	0.225	0.165
60	0.187	0.115	0.025	0.162	0.211	0.133
80	0.185	0.121	0.023	0.162	0.207	0.122
100	0.187	0.123	0.020	0.167	0.207	0.109
1000	0.188	0.121	0.006	0.181	0.194	0.034
10000	0.184	0.120	0.002	0.182	0.186	0.011

6.3 Steady-State Studies

The fundamental requirement in a steady-state study is the postponement of data collection during a simulation run until it is apparent that the simulation model is operating under steady-state conditions; that is, the stochastic processes associated with the output variables of interest have become stationary. A necessary (but not sufficient) condition for steady-state behaviour of the simulation model is the requirement that the underlying random variables associated with autonomous stochastic processes, such as arrival rates and service rates, are themselves stationary. But even when this is the case, the model's initial conditions usually give rise to circumstances that cause dependent stochastic processes in the simulation model to pass through a transient phase at the start of a simulation run.

Recall that for steady-state studies, the right-hand boundary of the observation interval is not specified. This provides the flexibility to execute a simulation run for as long as necessary in order to first reach steady-state conditions and then acquire sufficient data to permit meaningful conclusions. Consequently the execution of experiments for steady-state studies must address two important issues:

- Determining a warm-up period: A transient period is always present at the beginning of any simulation run. Behaviour data from this interval are (by definition) incompatible with the steady-state requirements of the study. The implication here is that a warm-up period that precedes the collection of data needs to be recognised. The duration of this period cannot be predicted and hence a mechanism for determining the end of the warm-up period must be incorporated into the experimentation procedure. Data collection can begin only after this transient, or warm-up period, has come to an end.
- Establishing confidence in the conclusions. A single simulation run can be executed for an extended observation interval to yield data from which a point estimate of the mean of the output variable (or variables) of interest can be calculated. Provided the length of the run has generated a sample of sufficiently large size, the estimate can have reasonable credibility (e.g., on the basis of the law of large numbers) However, a confidence interval for any such estimate requires a collection of independent observations in order to apply the techniques described in Section 6.2.2.

6.3.1 Determining the Warm-up Period

Considerable research effort has addressed the problem of establishing a suitable warm-up period for a simulation run, that is, an interval which allows sufficient time for the dependent stochastic process of interest to reach a steady-state (see, e.g., [6.4], [6.6], [6.7], [6.8]). The Welch moving average method is one of the many available approaches. It is graphically oriented, relatively straightforward, and provides reasonable estimates. This section outlines the application of this method (a more extensive presentation can be found in Law and Kelton [6.5]).

The Welch moving average method relies on a relatively small number of simulation run replications (e.g., five to ten). The duration of each replication needs to be sufficiently long so that it extends beyond the transient period. A typical replication is shown in Figure 6.3 which illustrates a representative transient condition at the start of the simulation run (e.g., a case where the simulation model begins without any consumer entity instances being present). The horizontal axis in Figure 6.3 corresponds to (simulated) time which has been compartmentalised into *m* time cells. The vertical axis shows how the average value for some output variable might change if separate averages were computed within the time cells. The changing shape of a hypothesised distribution function for this output variable is superimposed. The Figure shows that starting at time cell *D,* changes in average value no longer occur and hence steady-state can be assumed.

Selecting the size of the time cells and the number of time cells (which is equivalent to establishing the length of the simulation run) depends on the underlying nature of the simulation model. The size of the time cell should be large enough to be provide reasonable results (i.e., enough data points to compute a credible average within the cell), and yet short enough to be able to detect the existence of the transient.

Replication *j* generates an output set of n_j values; for example, $\{y_{k,j}: k = 1, 2, \ldots, n_j\}$. The average of those values that fall into time cell *i* is computed to produce $\overline{y}_{i,j}$ which is the *i*th cell average for the *j*th replication. Thus *n* replications will produce the set of *n* averages $\{\overline{y}_{i,j} : j = 1, 2, \ldots, n\}$ where *i* is the time cell index. The following steps are carried out to obtain an estimate of the time cell index where the system transient terminates, in other words, the system reaches steady-state.

1. Obtain the value $\overline{a}_i$ as the average over the *n* replications of the *i*th cell averages ($\overline{y}_{i,j}$); that is,

$$\overline{a}_i = \frac{1}{n}\sum_{j=1}^{n}\overline{y}_{i,j} \ .$$

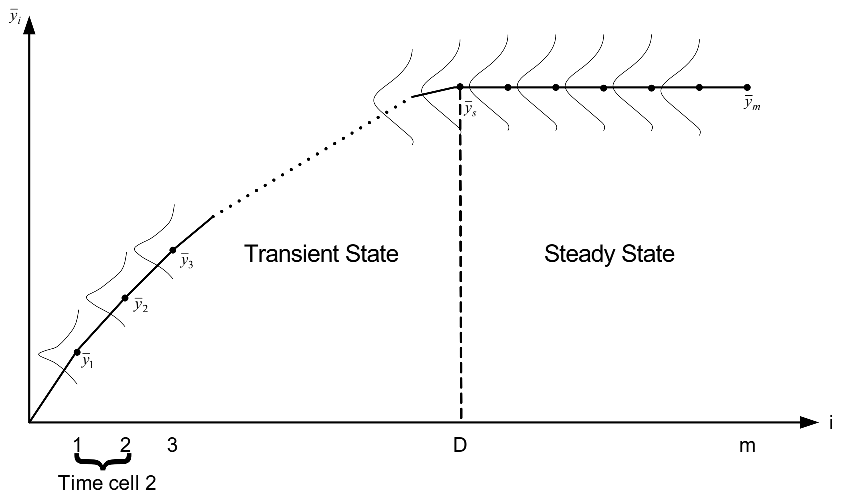


FIGURE 6.3. Reaching steady-state.

2. The values $\bar{a}_i$, i = 1, 2, . . . , m usually vary considerably. If plotted against index i the resulting graph is 'choppy' and difficult to interpret. A smoothing operation is required in order to smooth out rapid variations to obtain a smoother curve that captures the trend. For this purpose, the moving-average values $\bar{a}_i(w)$ are computed using Equation (6.2). The parameter w represents a window size that controls the smoothing operation. Its selection is by trial and error. Usually a number of values for w need to be tried. The objective is to find as small a value as possible that provides the desired smoothing effect.

$$\bar{a}_i(w) = \begin{cases} \dfrac{\sum_{l=-(i-1)}^{i-1} \bar{a}_{i+l}}{2i-1} & i = 1,\ldots,w \\ \\ \dfrac{\sum_{l=-w}^{w} \bar{a}_{i+l}}{2w+1} & i = w+1,\ldots,m-w \end{cases} . \tag{6.2}$$

3. Equation (6.2) is not as complex as it might appear. When $i > w$ there are w cell averages on either side of $\bar{a}_i$ that are averaged to produce the running average value $\bar{a}_i(w)$. When $i \le w$ there are not enough values preceding time cell $i$to fill the window. In this

case w is replaced with $(i - 1)$. Table 6.4 shows how the running averages are computed for the case where $w = 3$.

The values $\overline{a}_i(w)$ are plotted against the cell index i and it should be apparent from this graph when steady-state has been achieved. A good practice is to extend the apparent length of the warm-up period (say by 30%). The idea here is to err on the safe side by making the warm-up period somewhat longer than necessary rather than inappropriately short.

TABLE 6.4. Welch running average with $w = 3$.

i	$\overline{a}_i(3)$ **Equation**	$\overline{a}_i(3)$ **Expansion**
1	$\frac{\sum_{l=0}^{0} \overline{a}_{i+l}}{1}$	$\frac{\overline{a}_1}{1}$
2	$\frac{\sum_{l=-1}^{1} \overline{a}_{i+l}}{3}$	$\frac{\overline{a}_1 + \overline{a}_2 + \overline{a}_3}{3}$
3	$\frac{\sum_{l=-2}^{2} \overline{a}_{i+l}}{5}$	$\frac{\overline{a}_1 + \overline{a}_2 + \overline{a}_3 + \overline{a}_4 + \overline{a}_5}{5}$
4	$\frac{\sum_{l=-3}^{3} \overline{a}_{i+l}}{7}$	$\frac{\overline{a}_1 + \overline{a}_2 + \overline{a}_3 + \overline{a}_4 + \overline{a}_5 + \overline{a}_6 + \overline{a}_7}{7}$
5	$\frac{\sum_{l=-3}^{3} \overline{a}_{i+l}}{7}$	$\frac{\overline{a}_2 + \overline{a}_3 + \overline{a}_4 + \overline{a}_5 + \overline{a}_6 + \overline{a}_7 + \overline{a}_8}{7}$
•	•	•
•	•	•
•	•	•
$m - 3$	$\frac{\sum_{l=-3}^{3} \overline{a}_{i+l}}{7}$	$\frac{\overline{a}_{m-6} + \overline{a}_{m-5} + \overline{a}_{m-4} + \overline{a}_{m-3} + \overline{a}_{m-2} + \overline{a}_{m-1} + \overline{a}_m}{7}$

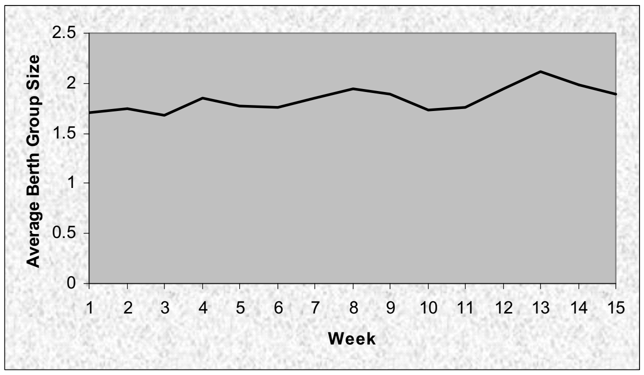
We illustrate the use of the Welch moving average method using version 1 of our port project (no intervention and no storms). The output variables of interest are berth group size and the tanker total wait time. Figures 6.4 and 6.5 show the results of 10 simulation runs ($n = 10$) each of duration 15 weeks. The time cells have a width of 1 week which means that $m = 15$. The following observations are noteworthy.

- In the case of the berth group size, there is no apparent transient. Even without the use of running averages (see Figure 6.4a), the graph is relatively smooth. This result can be attributed to the small size of the group (namely, three) which results in the available berths being quickly filled by the first few arrivals of tankers.
- A transient is certainly apparent for the tanker total wait time as shown in Figure 6.5 and moving averages are required to smooth out the graph. A window size of five provides a suitable result and shows that the transient lasts for approximately three weeks. Either four or five weeks can be selected as a suitable warm-up period.
- The warm-up period has relevance for the elimination of the transient in the tanker total wait time output variable. However, this does not preclude the collection of berth group size data during the warm-up period.

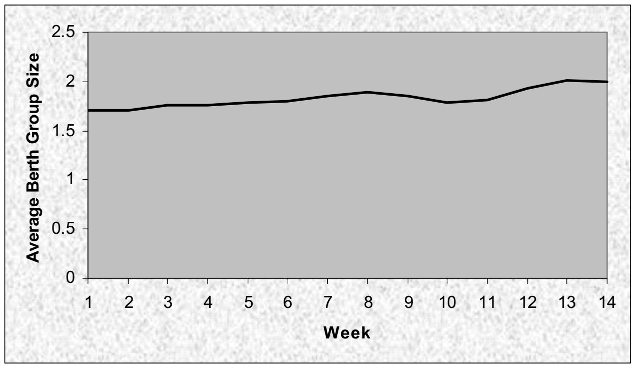
6.3.2 Collection and Analysis of Results

Extending the right-hand boundary of the observation interval allows more data to be collected during a simulation run and this provides the basis for a number of methods for generating the necessary data for analysis (i.e., a set of IID values for the output variable). We examine two approaches. The replication–deletion method is described and illustrated using experimentation with the port simulation program as presented in Section 6.3.3. An overview of the method of batch means is also given. A more comprehensive presentation of the available options can be found in Law and Kelton [6.5].

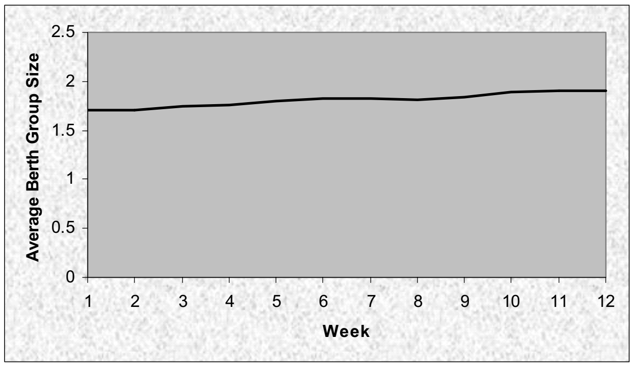
Our problem continues to be the determination of an estimate of the mean of an output variable Y, i.e. $\mu = E[Y]$. However, as previously noted, in steady-state studies we must reduce the effect of transient data, and ideally eliminate it.



(a) $\overline{a}_i$

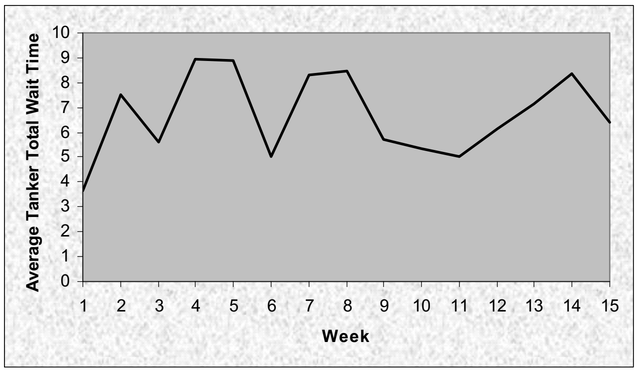


(b) $\overline{a}_i(1)$

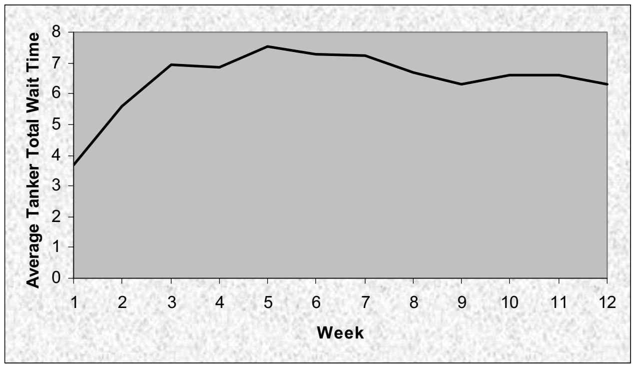


(c) $\overline{a}_i(3)$

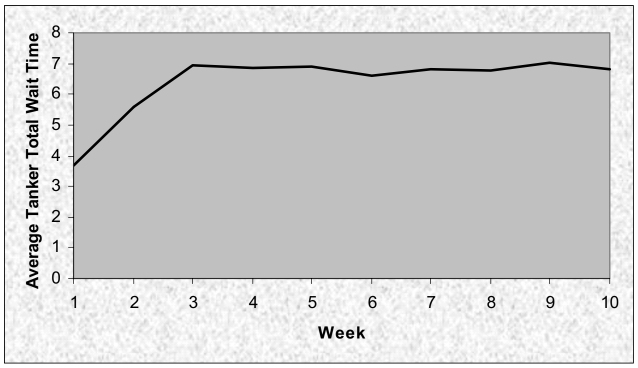
FIGURE 6.4. Welch method applied to berth group size.



(a) $\overline{a}_i$



(b) $\overline{a}_i(3)$



(c) $\overline{a}_i(5)$

FIGURE 6.5. Welch method applied to tanker total wait time.

A practical approach in the replication–deletion method is to determine the right boundary of the warm-period t_w (using methods such as the one described in Section 6.3.1) and to delete any output set data generated prior to t_w. The right boundary of the observation interval (i.e., t_f) is simply taken to be the value of (simulated) time when a sufficient amount of data has been collected to generate a valid and meaningful collection of output observations. In fact a sequence of n simulation runs is executed to produce a set of n output values. An important feature of this method is that it naturally generates a set of IID values. It does, however, have the computing time overhead of repeating the warm-up period for each of the replications.

This approach resembles the experimentation and output analysis previously outlined for a bounded horizon study (see Section 6.2). From the output data, a point estimate and confidence interval can be obtained using Equation (6.1). In the discussion of Section 6.2.2 it was noted that increasing the number of simulation runs (i.e., replications) reduced the confidence interval half length $\zeta(n)$ and increased the quality of the point estimate. This equally applies in the replication–deletion approach for a steady-state study. However, in a steady-state study, $\zeta(n)$ can also be reduced by increasing the length of the simulation run, that is, by adjusting the right-hand boundary t_f of the observation interval. Based on these observations the procedure for the replication–deletion method can be formulated as a straightforward extension of the earlier procedure presented in Section 6.2.2. It is as follows.

1. Choose values for r, and for the confidence level parameter C, and as well an initial reasonable value for t_f, and an initial value for n that is not smaller than 20.
2. Collect the n observed values $y_1, y_2, \ldots, y_n$ for the random variables Y_k, $k = 1, 2, \ldots, n$, that result from n replicated simulation runs of the simulation program **M** that terminate at time t_f. (Note that the output set, from which y_i's are obtained include only data collected after the end of the warm-up period)
3. From tabulated data for the Student t-distribution, determine $t_{n-1,a}$ where $a = (1 - C)/2$.
4. Compute $\bar{y}(n)$ and $\zeta(n)$ using Equation (6.1).
5. If $\zeta(n) < r\,\bar{y}(n)$ (or $\zeta(n)/\bar{y}(n) < r$) then accept $\bar{y}(n)$ as the estimate of μ and end the procedure, otherwise continue to Step 6.
6. EITHER choose Δn no smaller than 3 and collect the additional observations $y_{n+1}, y_{n+2}, \ldots, y_{n+\Delta n}$ through a further Δn replications, replace n with $n + \Delta n$ and repeat from Step 3

   OR increase the value of t_f by at least 50% and repeat from Step 2.[2]

[2] In some environments (e.g., Java), it may be possible to save the state of the simulation program for each replication so that simulations runs can be continued from the previously specified t_f.

The batch means method is an entirely different approach that requires only a single (but potentially 'long') simulation run. An advantage of this approach is economy of computing time because the warm-up period only needs to be accommodated once. The end of the observation interval t_f is selected to generate all the data necessary for analysis. However possible autocorrelation of the output data must be dealt with in order to generate the necessary IID data.

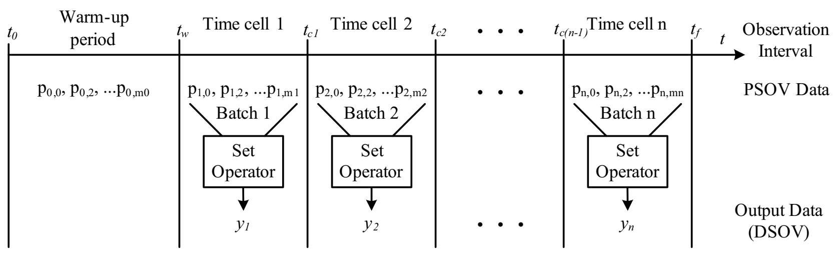


FIGURE 6.6. Output values using method of batch means.

To generate a set of IID values, the observation interval beyond the warm-up period is divided into n time cells as shown in Figure 6.6. The PSOV values that fall into a time cell is called a batch. The end result is a set of n batches. A DSOV output value is then computed for each batch, providing a set of output values y_i for $i = 1, 2, \ldots n$. Equation (6.1) can then be used to obtain a point estimate of the mean value of the distribution of the output variable of interest, together with the corresponding confidence interval.

One of the challenges of the batch means method is the proper selection of the length of the time cells. If the length is too short the y_i values may be correlated. Appropriate checks therefore need to be incorporated. Details about this method (and also other methods which use a single simulation run), can be found in Law and Kelton [6.5].

6.3.3 Experimentation and Data Analysis for the Port Project

The Java program given in Figure 6.7 illustrates how the required experiments for the steady-state study of the Port project can be implemented. The replication–deletion method is being used to generate the necessary output data with a warm-up period of five weeks (previously determined by Welch's method). The major steps include:

1. The *main* method obtains the value of t_f (the right-hand boundary of the observation interval expressed in weeks) from the command line arguments (*arg[0]*). A simulation run termination time expressed in hours is assigned to *endTime*.
2. A set of uncorrelated seeds is generated (to ensure independent replications) and saved into an array. The seeds are reused when carrying out the simulation runs for the alternative case of the port project (this implements the use of common random numbers described in Section 6.4).
3. Two sets of simulation runs are carried out, one set where *numBerths* = 3 (base case), and one set where *numBerths* = 4 (alternate case). This is implemented as a loop that increments *numBerths.*
4. A simulation run consists of instantiating the *PortVer1System* object that is initialised with the start time, number of berths, and random number generator seeds. The termination time of the *PortVer1System* object is first set to *warmUpTime* (using the *setTimef* method) and the simulation program executes for the warm-up period. The output set for the tanker total wait time is cleared. The termination time of the *PortVer1System* object is now set to *endTime* and continues execution until the end of the observation interval.
5. The output data values are then computed by the *ESOuputSet* methods,*computeTrjDSOVs*, and *computePhiDSOVs*. The values are then printed.

For each of the two cases (number of berths equals three and four), output data for each of the output variables (the average group size and tanker total wait time) using t_f equal to 10, 20, and 30 weeks are generated and analysed. In each case results are obtained for a sequence of increasing values of n (number of replications). For each of the output variables, Table 6.5 shows point estimate ($\bar{y}\, n)$), standard deviation ($s(n)$), and confidence interval half length ($\zeta(n)$) values computed from the recorded data using Equation (6.1). The last column shows the ratio of $\zeta(n)/\bar{y}(n)$ which gives a measure of the quality of the point estimate. As expected, increasing the number of runs (n) reduces the confidence interval half length $\zeta(n)$. Increasing the simulation run length also improves the quality of the results. A comparison of the three and four berth options is undertaken in Section 6.4.1 using an appropriate statistical framework.

```
public static void main(String[] args)
{
    double week=(7*24);
    double startTime=0.0;
    double warmUpTime=5*week;
    double endTime
    int NUMRUNS=10000;
    Seeds [] sds = new Seeds[NUMRUNS];
    // Find end time, tf, from command argument
    if(args.length != 1)
    {
       System.out.println("Usage: PortV1Exp2 <endTime>");
       System.exit(1);
    }
    endTime=Double.valueOf(args[0])*week;
    // Lets get a set of uncorrelated seeds
    RandomSeedGenerator rsg = new RandomSeedGenerator();
    for(int i=0 ; i<NUMRUNS ; i++)
       sds[i] = new Seeds(rsg.nextSeed(),rsg.nextSeed(),
                          rsg.nextSeed(),rsg.nextSeed(),
                          rsg.nextSeed());
    // Simulation Runs
    System.out.println("End Time = "+ args[0] +
                       "("+endTime+")");
    // Run for 3 berths and then 4 berths
    for(int numBerths=3 ; numBerths<=4 ; numBerths++)
    {
       System.out.println("Number of berths = "+numBerths);
       for(int i=0 ; i<NUMRUNS ; i++)
       {
        PortVer1System portSys = new PortVer1System(
                                 startTime,numBerths,sds[i]);
        portSys.setTimef(warmUpTime); // end of warmup
        portSys.runSimulation();
        portSys.tankerTW.clearSet();  // clear output set
        portSys.setTimef(endTime); // now run to tf
        portSys.runSimulation();
        // compute DSOV
        portSys.berthGrpN.computeTrjDSOVs(
                                portSys.time0,portSys.timef);
        portSys.tankerTW.computePhiDSOVs();
        System.out.println(portSys.berthGrpN.mean+", "+
                           portSys.tankerTW.mean);
       }
    }
}
```

FIGURE 6.7. Experiments with the Java port simulation program (corresponds to Example 1 of Section 4.3.1).

TABLE 6.5. Results from experiments with the Java port simulation program of Figure 6.7.

(a) Number of berths = 3.

t_f:	10 weeks				20 weeks				30 weeks			
n	$\bar{y}(n)$	s(n)	$\zeta(n)$	$\zeta(n)/\bar{y}(n)$	$\bar{y}(n)$	s(n)	$\zeta(n)$	$\zeta(n)/\bar{y}(n)$	$\bar{y}(n)$	s(n)	$\zeta(n)$	$\zeta(n)/\bar{y}(n)$
	Berth Group Size											
20	1.845	0.138	0.053	0.0290	1.858	0.072	0.028	0.0149	1.866	0.066	0.025	0.0137
30	1.820	0.139	0.043	0.0237	1.833	0.088	0.027	0.0149	1.842	0.078	0.024	0.0132
40	1.824	0.138	0.037	0.0201	1.833	0.095	0.025	0.0138	1.838	0.071	0.019	0.0103
60	1.827	0.147	0.032	0.0173	1.833	0.099	0.021	0.0117	1.828	0.077	0.017	0.0090
80	1.828	0.138	0.026	0.0141	1.834	0.102	0.019	0.0103	1.833	0.076	0.014	0.0077
100	1.823	0.138	0.023	0.0126	1.834	0.101	0.017	0.0091	1.830	0.078	0.013	0.0071
1000	1.834	0.130	0.007	0.0037	1.833	0.094	0.005	0.0027	1.833	0.077	0.004	0.0022
10000	1.826	0.132	0.002	0.0012	1.832	0.095	0.002	0.0009	1.833	0.078	0.001	0.0007
	Tanker Total Waiting Time											
20	7.144	2.387	0.923	0.1292	7.196	1.937	0.749	0.1041	7.256	1.543	0.597	0.0822
30	6.815	2.432	0.754	0.1107	7.042	1.695	0.526	0.0747	7.268	1.394	0.432	0.0595
40	6.703	2.482	0.661	0.0987	7.099	2.020	0.538	0.0758	7.320	1.487	0.396	0.0541
60	7.958	5.733	1.237	0.1554	7.451	2.626	0.567	0.0760	7.324	1.727	0.373	0.0509
80	7.857	5.243	0.976	0.1242	7.454	2.438	0.454	0.0609	7.410	1.762	0.328	0.0442
100	7.844	5.148	0.855	0.1090	7.665	2.498	0.415	0.0541	7.456	1.781	0.296	0.0397
1000	7.613	4.518	0.235	0.0309	7.682	2.817	0.147	0.0191	7.700	2.299	0.120	0.0155
10000	7.449	4.510	0.074	0.0100	7.686	2.926	0.048	0.0063	7.729	2.293	0.038	0.0049

(b) Number of berths = 4.

t_f:	10 weeks				20 weeks				30 weeks			
n	$\bar{y}(n)$	s(n)	$\zeta(n)$	$\zeta(n)/\bar{y}(n)$	$\bar{y}(n)$	s(n)	$\zeta(n)$	$\zeta(n)/\bar{y}(n)$	$\bar{y}(n)$	s(n)	$\zeta(n)$	$\zeta(n)/\bar{y}(n)$
	Berth Group Size											
20	1.859	0.149	0.058	0.0310	1.872	0.075	0.029	0.0154	1.877	0.068	0.026	0.0140
30	1.835	0.146	0.045	0.0247	1.847	0.090	0.028	0.0152	1.854	0.079	0.025	0.0132
40	1.839	0.145	0.039	0.0210	1.847	0.096	0.025	0.0138	1.850	0.071	0.019	0.0103
60	1.844	0.155	0.033	0.0181	1.846	0.101	0.022	0.0118	1.840	0.078	0.017	0.0092
80	1.844	0.146	0.027	0.0147	1.847	0.105	0.019	0.0105	1.845	0.077	0.014	0.0078
100	1.839	0.144	0.024	0.0130	1.847	0.104	0.017	0.0094	1.841	0.080	0.013	0.0072
1000	1.850	0.135	0.007	0.0038	1.846	0.097	0.005	0.0027	1.845	0.079	0.004	0.0022
10000	1.842	0.136	0.002	0.0012	1.845	0.098	0.002	0.0009	1.846	0.080	0.001	0.0007
	Tanker Total Waiting Time											
20	2.533	1.045	0.404	0.1595	2.534	0.641	0.248	0.098	2.533	0.422	0.163	0.0644
30	2.488	0.938	0.291	0.1170	2.556	0.569	0.177	0.069	2.598	0.450	0.140	0.0537
40	2.452	0.885	0.236	0.0962	2.554	0.564	0.150	0.059	2.582	0.422	0.112	0.0435
60	2.769	1.562	0.337	0.1217	2.601	0.705	0.152	0.059	2.547	0.487	0.105	0.0412
80	2.709	1.432	0.267	0.0984	2.582	0.691	0.129	0.050	2.549	0.499	0.093	0.0364
100	2.711	1.379	0.229	0.0845	2.624	0.683	0.113	0.043	2.552	0.496	0.082	0.0323
1000	2.539	1.114	0.058	0.0228	2.557	0.639	0.033	0.013	2.559	0.508	0.026	0.0103
10000	2.501	1.094	0.018	0.0072	2.557	0.662	0.011	0.004	2.564	0.512	0.008	0.0033

6.4 Comparing Alternatives

A frequently occurring requirement among the goals of a modelling and simulation project is the evaluation of several alternate system designs. For example, what reduction in maximum patient waiting time could be expected in the emergency admitting area of a large hospital if an additional orthopaedic specialist were hired or what might be the impact on traffic flow in the downtown core of a large city if a network of one-way streets were implemented? There can be a large number of such design alternatives that need to be evaluated but we first examine the case where there are only two.

In principle the problem solution is straightforward. Develop a simulation program for each of the scenarios (alternatives), obtain a value for some common performance measure applied to each scenario (e.g., a mean value estimate for the distribution of some DSOV), and compare the values obtained. There is, however, a serious complication that emerges, namely, what assurance is there that any observed difference between the performance measure values is a consequence of the design difference being studied and not simply a consequence of the inherent random behaviour within the model?

A number of different approaches have emerged for dealing with this problem and comprehensive discussions can be found in the literature (e.g., Banks et al. [6.1] and Goldsman and Nelson [6.2]). One of the most straightforward is called the *paired-t confidence interval method* The objective here is to first establish a confidence interval for an estimate of the mean of a random variable that is the difference between the output variables associated with each of the scenarios. A decision about relative superiority is then based on the position of the confidence interval relative to zero. Some details are provided below.

6.4.1 Comparing Two Alternatives

Suppose that Y is the output variable to be used for the evaluation and let's assume that we seek as large a value as possible for this variable. The simulation program for each of the design alternatives is replicated n times with appropriate care taken to ensure that in each case the n observations of Y can be assumed to be independent (i.e., by proper management of the random number streams that 'drive' the simulation models). Suppose that y_{1k} is the value of Y obtained for case 1 on the kth replication and suppose that y_{2k} is the value for case 2 on the kth replication. Let:

$$\begin{aligned}
d_k &= y_{2k} - y_{1k} \qquad k = 1,2,\ldots n \\
\bar{d}(n) &= \frac{1}{n}\sum_{k=1}^{n} d_k \\
s^2(n) &= \frac{\sum_{k=1}^{n}(d_k - \bar{d}(n))^2}{n-1} \\
\zeta &= \frac{t_{n-1,\alpha}\, s(n)}{\sqrt{n}}
\end{aligned} \tag{6.3}$$

where $t_{n\text{-}1,a}$ is a value from the Student *t*-distribution (see Table A1.4) that corresponds to $(n - 1)$ degrees of freedom and $a = (1 - C)/2$ with C the confidence level parameter. Here $\overline{d}(n)$ is a point estimate of the mean of the differences and $s^2(n)$ is the sample variance. (The similarity of these results with those given in Equation (6.1) is worth noting.) The associated confidence interval is $CI(n) = [\overline{d}(n) \pm \zeta]$.

There are three possible outcomes based on *C*I(*n*); namely,

a) If *C*I(*n*) lies entirely to the right of zero then the result of case 2 exceeds the result of case 1 with a level of confidence given by 100*C*%.
b) If *C*I(*n*) lies entirely to the left of zero then the result of case 1 exceeds the result of case 2 with a level of confidence given by 100*C*%.
c) If *C*I(*n*) includes zero then at the level of confidence, 100*C*%., there is no meaningful difference between the two cases.

The procedure outlined above is best carried out in conjunction with a technique called *common random numbers* (CRN). When undertaking the comparison of the data that flow from the two simulation programs that embody the two design alternatives, there is reason to be concerned about the extent to which any observed difference is a genuine reflection of the design alternatives or is simply the result of a lack of symmetry in the random phenomena that take place within the respective simulation models.

The common random number technique seeks to establish this symmetry and thus enhance the reliability of the conclusions. The application of the technique corresponds to endeavouring to ensure that, insofar as possible, the random phenomena within the two simulation programs are co-ordinated; for example, comparable entities flowing in the two models are subjected to the same sequence of delays. In principle, this can be achieved by the strict management of the random variate generation procedures within the two programs. This coordination is straightforward for input data models. The coordination task can also be easily achieved with all data models when the simulation model is relatively simple. However, except for input data models, the coordination task can become increasingly more difficult as the simulation model complexity increases. Often the design differences themselves may inhibit a rigorous application of the approach.

The common random number procedure outlined above has the effect of establishing correlation between the output data generated in corresponding simulation runs with the two alternative designs. This, in turn, has a quantitative manifestation; more specifically, the procedure, when operating as intended, should yield the inequality:

$$s^2(n) < s_1^2(n) + s_2^2(n) \ , \tag{6.4}$$

where $s_1^2(n)$ and $s_2^2(n)$ are the variances for the data obtained for Case 1 and Case 2, respectively and $s^2(n)$ is the value obtained from Equation (6.3).

We return now to our experiments with the port project as outlined in Section 6.3.3. For the two cases where the number of berths is 3 and 4, Table 6.6 shows the output data for each of the output variables (the average group size and tanker total wait time) from a sequence of experiments with t_f = 20 weeks and n = 30. The difference column is obtained as (numBerths=4) – (numBerths=3). The comparison of the two alternatives is carried out using Equation (6.3) and the results are provided at the bottom of Table 6.6 (CI min and CI max are the boundaries of the confidence interval). Some interpretation of the data is as follows.

1. It is clear that increasing the number of berths from three to four does decrease the mean tanker total wait time (by almost 4.5 hours).
2. Although the confidence interval for the berth group size is to the right of zero, the point estimate of the difference is so small relative to the individual point estimates we are obliged to conclude that increasing the berth group size has no meaningful effect on this output variable. This is somewhat counterintuitive but is a consequence of the relative values of tanker arrival rate, the tug's cycle time (time to deberth and berth a tanker), and the tanker loading times. For example, experimentation with the model has shown that when the loading times are increased, the average berth group size does increase. An alternate measure that would be interesting is the percentage of time that all available berths are occupied. The interested reader is encouraged to experiment with the simulation program by exploring the effects of changing these various times in the simulation model (The *PortVer1* simulation model is available from the textbook Web site).

Table 6.7 shows the data obtained from equivalent experiments which do not use common random numbers (CRN) for the two cases of interest, that is, numBerths=3 and numBerths=4. This was achieved by not using the same seeds for the random number generators that implement the data modules in the experiments. Note that the confidence interval half length $\zeta(n)$ increases for both output variables when compared to the results in Table 6.6. Note also that it can be shown that the Tanker Total Wait Time data in Table 6.7 is not consistent with the inequality of Equation (6.4).

TABLE 6.6. Comparing alternative cases in the port project of Example 1 (with CRN and n = 30).

	Berth Group Size			**Tanker Total Wait Time**		
Run	**numBerths=3**	**numBerths=4**	**Difference**	**numBerths=3**	**numBerths=4**	**Difference**
1	1.721	1.729	0.008	5.981	2.182	-3.799
2	1.847	1.862	0.014	7.433	3.182	-4.251
3	1.911	1.920	0.008	7.594	2.758	-4.836
4	1.876	1.888	0.011	6.299	2.039	-4.260
5	1.897	1.917	0.020	7.177	2.483	-4.694
6	1.833	1.843	0.011	4.589	1.679	-2.910
7	1.866	1.888	0.023	8.234	3.411	-4.823
8	1.864	1.877	0.013	9.310	2.995	-6.315
9	1.790	1.796	0.006	6.398	2.580	-3.819
10	1.949	1.969	0.020	7.131	2.403	-4.728
11	1.952	1.956	0.003	9.387	2.950	-6.438
12	1.830	1.846	0.016	7.776	2.548	-5.229
13	1.720	1.743	0.023	4.764	1.719	-3.046
14	2.004	2.044	0.039	12.640	4.030	-8.610
15	1.811	1.819	0.008	4.678	1.623	-3.055
16	1.834	1.852	0.018	6.126	2.446	-3.680
17	1.932	1.941	0.009	9.013	3.127	-5.887
18	1.836	1.841	0.005	5.782	1.731	-4.050
19	1.828	1.833	0.004	5.795	2.017	-3.778
20	1.856	1.870	0.014	7.816	2.782	-5.035
21	1.921	1.924	0.003	7.344	2.853	-4.490
22	1.744	1.754	0.011	6.092	2.491	-3.601
23	1.636	1.658	0.023	6.183	2.659	-3.524
24	1.823	1.856	0.032	6.797	3.126	-3.672
25	1.747	1.746	0.000	6.619	2.214	-4.405
26	1.843	1.848	0.005	7.074	2.422	-4.652
27	1.668	1.683	0.015	4.597	1.801	-2.795
28	1.699	1.715	0.015	7.429	2.920	-4.509
29	1.928	1.964	0.036	8.780	3.093	-5.686
30	1.828	1.833	0.005	6.410	2.417	-3.993
$\overline{y}(n)$			0.014			-4.486
$s(n)$			0.010			1.221
$\zeta(n)$			0.003			0.379
CI Min			0.011			-4.864
CI Max			0.017			-4.107

TABLE 6.7. Comparing alternative cases in the port project of Example 1 (without CRN and $n = 30$).

	Berth Group Size			Tanker Total Wait Time		
Run	**numBerths=3**	**numBerths=4**	**Difference**	**numBerths=3**	**numBerths=4**	**Difference**
1	1.721	1.834	0.113	5.981	2.943	-3.038
2	1.847	1.881	0.033	7.433	2.591	-4.843
3	1.911	1.887	-0.024	7.594	2.507	-5.087
4	1.876	1.881	0.005	6.299	3.278	-3.021
5	1.897	1.862	-0.035	7.177	2.107	-5.070
6	1.833	1.835	0.002	4.589	3.588	-1.001
7	1.866	1.868	0.002	8.234	2.224	-6.009
8	1.864	1.923	0.059	9.310	2.527	-6.783
9	1.790	1.838	0.048	6.398	1.828	-4.570
10	1.949	1.811	-0.138	7.131	1.895	-5.236
11	1.952	1.905	-0.048	9.387	1.418	-7.969
12	1.830	1.876	0.047	7.776	3.003	-4.774
13	1.720	1.829	0.109	4.764	2.183	-2.581
14	2.004	1.879	-0.126	12.640	2.633	-10.007
15	1.811	1.822	0.011	4.678	2.741	-1.937
16	1.834	1.905	0.071	6.126	4.414	-1.712
17	1.932	1.925	-0.008	9.013	4.340	-4.673
18	1.836	1.854	0.018	5.782	1.658	-4.124
19	1.828	1.865	0.037	5.795	2.158	-3.637
20	1.856	1.878	0.022	7.816	2.346	-5.470
21	1.921	1.844	-0.077	7.344	2.360	-4.984
22	1.744	1.937	0.194	6.092	3.644	-2.448
23	1.636	1.709	0.073	6.183	2.934	-3.249
24	1.823	1.870	0.047	6.797	2.117	-4.680
25	1.747	1.712	-0.034	6.619	2.200	-4.419
26	1.843	1.868	0.026	7.074	2.472	-4.602
27	1.668	1.686	0.017	4.597	2.799	-1.798
28	1.699	1.769	0.070	7.429	1.895	-5.533
29	1.928	1.878	-0.050	8.780	1.786	-6.994
30	1.828	1.909	0.081	6.410	4.787	-1.623
$\overline{y}(n)$			0.018			-4.396
$s(n)$			0.069			2.003
$\zeta(n)$			0.021			0.621
CI Min			-0.003			-5.017
CI Max			0.039			-3.775

6.4.2 Comparing More than Two Alternatives

The paired-*t* confidence interval method described above can be extended to the case where multiple comparisons need to be carried out. The basis for carrying out this extension is provided by the Bonferroni inequality (sometimes called the Boole inequality). It states that:

$$P[\bigcap_{k=1}^{K} A_k] \geq (1-K) + \sum_{k=1}^{K} P[A_k] \ .$$

In our context, the A_k can be interpreted as the event (in a probability context) that the *k*th confidence interval contains the *k*th mean in a collection of *K* (pairwise) comparisons, The Bonferroni inequality, in effect, places constraints on the individual comparisons in order to achieve an overall result that has a prescribed level of confidence, 100*C*%. In other words with 100*C*% confidence, the mean differences all fall into their respective confidence intervals. The (simplified) result that flows from the Bonferonni inequality is that each of the *K* comparisons should be carried out with a confidence level parameter value of:

$$C_K = 1 - \left(\frac{1-C}{K} \right) \ . \qquad (6.5)$$

Note that the result given in Equation (6.5) is overly restrictive because it has imposed the unnecessary (but simplifying) requirement that the confidence level parameter of all constituent comparisons be equal.

The following is a typical scenario. There exists a 'base case' which normally corresponds to the current status of the SUI. The project goals introduce *M* alternate designs together with the requirement to identify the best of the alternate designs by comparing each alternative to the base case. Thus *K* = *M* comparisons need to be made. If an overall confidence level of 100*C*% is stipulated then the *K* individual comparisons have to be carried out with a confidence level parameter of C_K as given in Equation (6.5).

It may, on the other hand, be stipulated in the project goals that the *M* alternative designs not only be compared to the base case but also be pairwise compared to each other. In this case, there is a requirement for *K* = *M*(*M* + 1)/2 comparisons. The number of comparisons can easily rise quickly and the reliability of the procedure deteriorate. In addition, of course, the computational overhead can become overwhelming.

Some illustrative results obtained using the multiple alternatives procedure outlined above are given in Table 6.9. The results relate to the Kojo's Kitchen project. We consider a base case (Case 1) which corresponds to the two employees working over the entire business day (10:00 AM – 9:00 PM) and three alternative employee scheduling options (Cases 2, 3, 4). These options allocate different numbers of employees to various segments of the day. The employee scheduling schemes are summarised in Table 6.8. The rightmost column of this Table provides the total number of employee-hours associated with each option. This is relevant in the ultimate selection decision because it represents the 'cost' of the option. The output variable of interest continues to be the percentage of customers who wait more than five minutes before receiving service.

TABLE 6.8. Multiple scheduling alternatives for Kojo's Kitchen.

	Slow (10:00am-11:30am)	**Busy** (11:30am-1:30pm)	**Slow** (1:30pm-5:00pm)	**Busy** (5:00pm-7:00pm)	**Slow** (7:00pm-9:00pm)	**Emp-Hours**
Case 1 (Base Case)	2	2	2	2	2	22
Case 2	2	3	2	3	2	26
Case 3	1	3	1	3	1	19
Case 4	1	3	2	3	1	22.5

Table 6.9 provides a summary of the each of the three comparisons. The results shown for Diff21 are obtained by subtracting the results of the base case (Case 1) from Case 2 and applying Equation (6.3), and similarly for Diff31 and Diff41. These results were obtained using the Java simulation program previously discussed in Section 6.2.3. In each case the results are based on data from 100 replications (n = 100) and use of a confidence level parameter value of $C_k = 0.968$ in the determination of the confidence interval for the individual comparisons. This gives a value of $C = 0.904$ using Equation (6.5), that is, a confidence of 90.4% in the conclusions from the comparison. Table 6.9 suggests that the scheduling alternative of Case 2 provides the best improvement over the base case. (Unfortunately it is also the most expensive! Note however that scheduling in Case 4 provides a significant improvement at very little additional cost).

TABLE 6.9. Results for multiple scheduling alternatives (Kojo's Kitchen).

Comparison	**Point Estimate** $\tilde{y}(n)$	**s(n)**	ζ	**CI Min**	**CI Max**	$\zeta/\tilde{y}(n)$
Diff21	-0.315	0.011	0.024	-0.340	-0.291	-0.076
Diff31	-0.127	0.013	0.028	-0.155	-0.099	-0.220
Diff41	-0.243	0.012	0.025	-0.268	-0.218	-0.105

6.5 Exercises and Projects

6.1 Use the program developed in Problem 5.1 to carry out experiments that provide the values required for the graphs that are stipulated in the goals of the project outlined in Problem 4.1. Write a short report that outlines the problem, the goals of the modeling and simulation project, and the conclusions obtained from the study.

6.2 Use the program developed in Problem 5.2 to carry out experiments that provide the values required for the graphs that are stipulated in the goals of the project outlined in Problem 4.2. Write a short report that outlines the problem, the goals of the modeling and simulation project, and the conclusions obtained from the study.

6.3 Use the program developed in Problem 5.3 to carry out experiments that provide values for the proposed performance measures referred to in part (a) of Problem 4.3. Write a short report that outlines the problem, the goals of the modeling and simulation project, and the conclusions obtained from the study.

6.4 Use the program developed in Problem 5.5 to carry out experiments that provide values for the proposed performance measures referred to in part (a) of Problem 4.3. Write a short report that outlines the problem, the goals of the modeling and simulation project, and the conclusions obtained from the study.

6.5 Use the program developed in Problem 5.7 to carry out experiments that provide values for the proposed performance measures referred to in part (b) of Problem 4.4. Write a short report that outlines the problem, the goals of the modeling and simulation project, and the conclusions obtained from the study.

6.6 Use the program developed in Problem 5.8 to carry out experiments to evaluate the effects of balking introduced in Problem 4.5. Write a short report that outlines the problem, the goals of the modeling and simulation project, and the conclusions obtained from the study.

6.6 References

6.1. Banks J., Carson II, J.S., Nelson, B.L., and Nicol, D.M., (2005), *Discrete-Event System Simulation*, 4th edn., Pearson Prentice Hall, Upper Saddle River, NJ.

6.2. Goldsman, D. and Nelson, B.L., (1998), Comparing Systems Via Simulation, in J. Banks (Ed.), *Handbook of Simulation*, John Wiley, New York, pp. 273–306.

6.3. Goldsman, D. and Nelson, B.L., (2001), Statistical selection of the best system, in B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, (Eds.), *Proceedings of the 2001 Winter Simulation Conference* IEEE Press, Piscataway, NJ, pp. 139–146.

6.4. Goldsman, D., Schruben, L.W., and Swain J.J., (1994), Test for transient means in simulation time series, *Naval Research Logistics Quarterly*, **41**: 171–187.

6.5. Law, A.M. and Kelton, D.W., (2000), *Simulation Modeling and Analysis*, 3rd edn., McGraw-Hill, New York.
6.6. Robinson S., (2002), A statistical process control approach for estimating the warm-up period, in *Proceedings of the 2002 Winter Simulation Conference*, IEEE Piscataway, NJ, pp. 439–446
6.7. Roth, E., (1994), The relaxation time heuristic for the initial transient problem in M/M/K queuing systems, *European Journal of Operational Research*, **72**: 376–386.
6.8. Welch, P., (1983), The statistical analysis of simulation results, in S. Lavenberg (Ed.), *The Computer Performance Modeling Handbook*, Academic Press, New York, pp. 268–328.