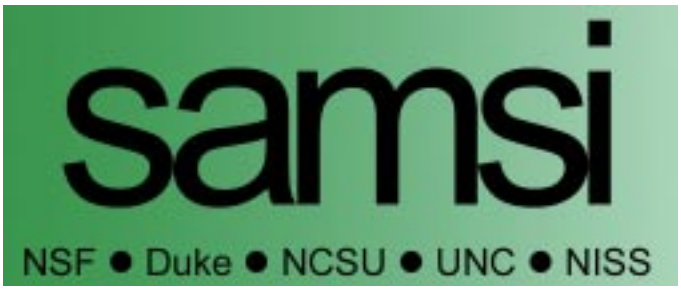


# Simulation in Industrial Statistics

David Rios Insua, Jorge Muruzabal, Jesus Palomo,
Fabrizio Ruggeri, Julio Holgado, Raul Moreno

Technical Report #2005-8
July 28, 2005

This material was based upon work supported by the National Science Foundation under Agreement No. DMS-0112069. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Statistical and Applied Mathematical Sciences Institute
PO Box 14006
Research Triangle Park, NC 27709-4006
www.samsi.info

# Simulation in Industrial Statistics

David Ríos Insua, Jorge Muruzabal, Jesus Palomo,
Fabrizio Ruggeri, Julio Holgado, Raul Moreno
Statistics and Decision Sciences Group, U. Rey Juan Carlos, Spain
Duke University-SAMSI, USA
CNR-IMATI, Italy
Computing Services, U. Rey Juan Carlos, Spain
Telefonica I+D, Spain

**Abstract**

We provide a brief introduction to simulation methods and tools within Industrial Statistics. We illustrate both Monte Carlo and discrete event simulation, within a common framework, through several motivating examples showing the relevance of simulation. We then describe random number and variate generation, and how to obtain and analyse output data from a simulation model. A case study in modelling a workflow line within the digitalisation industry guides our discussion. Relevant pointers to references and url's are provided.

Keywords: Simulation, Monte Carlo methods, Discrete event simulation, Industrial statistics

## 1 Introduction

Tipically, once an organisation has realised that a system is not operating as desired, it will look for ways to improve its performance. To do so, sometimes it is possible to experiment with the real system and, through observation and the aid of Statistics, reach valid conclusions towards future system improvement. Many chapters in this book have illustrated this point. However, experiments with a real system may entail ethical and/or economical problems, which may be avoided dealing with a prototype, a physical model.

Sometimes, it is not feasible or possible, to build a prototype, yet we may obtain a mathematical model describing, through equations and constraints, the essential behaviour of the system. This analysis may be done, sometimes, through analytical or numerical methods, but the model may be too complex to be dealt with. In such extreme cases, we may use simulation. Large complex system simulation has become common practice in many industrial areas such as the prediction of the performance of integrated circuits, the behavior of controlled nuclear fusion devices, the properties of thermal energy storage devices or the stresses in prosthetic devices. Essentially, simulation consists of (i) building a computer model that describes the behaviour of a system; and (ii) experimenting with this computer model to reach conclusions that support decisions. In this chapter, we introduce key concepts, methods and tools from simulation with the industrial statistics practitioner in mind.

In order to give a flavour of various Simulation applications and distinguish between discrete event simulation (DES) and Monte Carlo (MC) simulation, we describe several real cases. We outline an example in gas pipeline reliability to illustrate simulation methods in Bayesian statistics; an example in vehicle crash-tests modeling, to illustrate large scale models, and a detailed description of a DES model to predict the behaviour of a complex workflow line.

**Designing a workflow line** The study of moderately complex workflow lines, so as to optimize their performance, quickly leads to mathematical models which may be only dealt with through simulation. The specific context we consider here is that of modelling massive data capturing lines for financial institutions. This task has been outsourced by banks over companies working in a very competitive environment, which, therefore, look for gains in efficiency.

The process specifically considered here is as follows. Every day several bags arrive to the line containing bank documents to be digitized and recorded. Such documents must be stored in CD's for later use in the bank with several financial checks in between. The line is meant to deliver the completed task at a given time window. Otherwise, the company will be heavily penalized economically. From input (bags with financial documents) to output (CD's) the process goes through several stages. Our aim here is to forecast whether the company will deliver the work before the deadline and, if not, detect bottlenecks and suggest line reconfigurations.

We now briefly describe a typical configuration and the whole process within this massive data capturing line, with the various stages and the involved operations, resources, objectives and intervening agents:

- Paper preparation. The documents received are arranged and organised, taking into account the different types of documents, by several persons for later processing.
- Document scanning, to produce digitized documents. Two scanners are used (one for backup). Some of the documents could be illegible, and we would need to mark such documents for later special treatment.
- Optical recognition (OCR) of relevant features, like dates, imports and so on, so as to reduce later workload. Some of them may be wrongly recognised. Several workstations are involved at this stage.
- Recording. Documents are classified as legible or illegible; these are filled manually. Up to forty people may take part in this process.
- Resolution of illegible documents. In this phase, a person is in charge of looking for the original document, to digitize it correctly. An operator controls a workstation dedicated to this task.
- Coding. A second image recording takes place with all images that the recorder could not deal with. An operator controls a workstation dedicated to this task.
- Date verification. If necessary, dates are corrected. An operator controls a workstation dedicated to this task.
- Field verification. If necessary, fields are corrected. An operator controls a workstation dedicated to this task.
- Final verification. Every now and then, a document is inspected for correct recording. If necessary, they are corrected. An operator controls a workstation dedicated to this task.
- Balancing the totals. All documents within a lot are checked to verify their imports. A person with a workstation and a balancing program takes care of this.

- Modification. Those documents which did not pass the balancing phase are corrected by contrast with the original documents.
- Export. A file is generated and recorded on CD-ROMs.

On each of the stages of the scenario described above, the difficulties to deal with are: the resources are shared by processes, several random processes are involved, several parameters are under control and others need to be estimated. The basic problem we want to solve is whether the company will be able to finish the work before a limit time $h$, given the current system configuration, defined by the parameters under control $z$ (number of OCR's, the number of workers at each workstage,...). Formally, if $T$ designates the actual completion time window, we want to find out whether

$$Pr(T \leq h|z)$$

will be big enough so as to guarantee the completion of work. Recall that, otherwise, there is a big economic penalty. In case this probability is not big enough, we must reconfigure the system, modifying $z$. Because of the complexity of the model, we are unable to compute such probability analytically and, consequently, simulation is used.

**Reliability of a gas pipeline system** Gas distribution networks can be considered repairable systems whose failures (gas escapes) may be described by nonhomogeneous Poisson processes, see the review paper by Ruggeri (2005) and references therein, which address various aspects in the statistical analysis of gas escapes.

In a consulting case, both homogeneous and nonhomogeneous Poisson processes were used for "old" cast iron and steel pipes, respectively, since the former pipes are not ageing, whereas the latter are subject to corrosion. Both maximum likelihood and Bayesian estimation were used on data collected from a gas company. The final objective was to fit a model suitable for the description of the network so as to predict which pipes and operating conditions were more likely to lead to gas escapes, identifying influential factors on escapes and related costs, due, e.g., to preventive maintenance and repair. An important issue to note was that many data was missing and was only fairly complete for recent years. We had also available experts which we interviewed to obtain their prior beliefs on the parameters of interest, tested through sensitivity analysis.

Simulation techniques were fundamental in this study. We just quote several applications:

- Modelling. Typical repairable systems have a "bath-tub" shaped intensity function, whereas more complex behaviour is possible. It is quite common to consider different models in different time intervals; this is a typical change-point problem. The estimation of change points and their number and changes in the parameters have been addressed in a Bayesian framework using a simulation technique called Reversible Jump Markov Chain Monte Carlo, due to Green (1995), in which samples from the posterior distributions of the parameters are obtained along with the distribution of the number and the location of the change points.

- Parameter estimation. Standard optimisation techniques are used to get maximum likelihood estimates, whereas Markov Chain Monte Carlo techniques (mostly Gibbs and Metropolis-Hastings) are now routine in Bayesian analysis and they were used in our study.

- Missing data. Installation dates of failed pipes were sometimes missing and they were drawn from a probability distribution.

- Comparison between estimates. MLE and Bayes estimates were compared through extensive simulations, drawing data from known processes and trying to estimate their parameters.

**Reducing vehicle-crash tests** Traditionally, the automobile and aeronautic industries have been using physical experiments for design and security test purposes, such as car crash tests, wind tunnels,... However, as the market becomes more competitive, the time to deliver gets shorter and budgets get tighter. This implies that the number of prototypes built on the scale required to gather sufficient information for decision or research purposes is reduced and are not sufficient to produce good estimates. In theses cases, simulation becomes essential to fill the gap of lack of information. A fully Bayesian approach would account for the uncertainty on the simulation input parameters in an MCMC fashion. A problem in this case is that, since to ensure convergence we have to run the Simulation a long number of iterations, but, on the other hand, the simulation model is very complex, this is computationally intensive and slow. Therefore, due to time restrictions, a

proper experimental design should be performed in order to cover the input parameter space with the smallest amount of iterations, see e.g., Atkinson and Donev (1992).

As an example, we considered the simulation of a model of a vehicle being driven over a road with two major potholes, see Bayarri et al. (2004) for full details. The model included thirty parameters but, to simplify, we just consider nine of them uncertain, the rest being assumed known. The uncertainty on these parameters comes from deviations of manufacturing properties of the materials from their nominal values (measured parameters but with uncertainty), and calibration parameters model (non-measured input parameters). The outputs of the simulation were the vertical tensions experimented on the suspension system of a particular vehicle while hitting two consecutive potholes. Each simulation model run takes around four days to produce one output for a particular experiment design vector, and sometimes, due to inconsistencies with it, the run may fail (does not converge). As a result, we will obtain curves for each of the design experiment vectors, for the vertical tension experimented at different spots of the vehicle along the distance that contains the two potholes.

The advantages of using a simulation approach in this problem are that the physical experiment is expensive, and that it provides insights about testing a potential vehicle with specifications that are currently infeasible. For example, it could be of interest for the car maker to test the behavior of a current vehicle model with particular springs or tires that have not been developed yet, but could be ordered to a vendor.

The simplifications usually assumed on these large simulation models cause big differences with the physical experiments. To overcome this drawback, we need to develop methods to estimate the bias. This information can be used in different useful ways. For example, by using it as a target (bias equal to zero) to improve the simulation model, say by variable and model selection; to produce bias corrected prediction curves of reality, without performing the physical experiment, for the same physically tested vehicle, another vehicle of the same type, or another vehicle of different type stemming from the simulation curves obtained,... See Berger et al. (2004) for more details.

## 1.1 Outline of chapter

We have sketched several real case studies concerning applications of simulation in Industrial Statistics. We shall use the first one to illustrate the basic four-step process in any simulation experiment, once we have estimated the corresponding simulation model:

1. Obtain a source of random numbers.
2. Transform them into inputs to the simulation model.
3. Obtain outputs of the simulation model, stemming from the inputs.
4. Analyse the outputs to reach conclusions.

We devote a section to each of these four steps. Pointers to recent and/or relevant literature and urls are given.

# 2 Random number generation

We start by introducing key concepts in random number generation, the basic ingredient in any simulation analysis. The issue is how to choose a source that provides a sufficient amount of (pseudo)random numbers to carry out a simulation experiment.

Sometimes, we may appeal to historical data from the relevant process. As an example, in a reservoir management problem, see Ríos Insua and Salewicz (1995), we used the available historical time series of inflows for simulation purposes, as such series was long enough and we had no possibility of controlling the source of randomness. However, very frequently, we either lack a sufficiently long series or we want to, somehow, control the source of randomness. As an example, if we are simulating a computer centre, one of the typical sources of uncertainty will be the arrival process, modelled, say, through a Poisson process. Typically, we shall be interested in studying the performance of the system for a long enough period and under different conditions. The first issue suggests that we might need to store an excessive amount of historical data; the second one may imply costly transformations, when we modify the parameters. One possibility would be to use some physical mechanism to generate random numbers and store them for later access in some storage device. However, this procedure is too slow in many application areas.

As an alternative, we may use algorithmic procedures for random number generation. The idea, due to Von Neumann (1951), is to produce numbers which seem random using arithmetic computer operations: starting from an initial *seed* $(u_0, u_{-1}, ..., u_{-p+1})$, generate a sequence through $u_i = d(u_{i-1}, ..., u_{i-p})$, for a certain function $d$. Clearly, once the seed is chosen, the sequence is determined. For that reason, we adopt the following criteria:

**Definition 1** *A sequence $(u_i)$ is of random numbers if nonoverlapping h-uples of subsequent numbers are approximately uniformly distributed in $(0,1)^h$, for $h = 1, 2, ..., n$, and $n$ sufficiently large for the application of interest.*

Formally, any test applied to a finite part of $(u_i)$ which would try to detect relevant deviations from randomness would not reject the null hypothesis that the numbers are random.

These properties of uniformity in (0,1) and independence are complemented with others referring to computational efficiency, including, speed; little memory consumed; portability; implementation simplicity; reproducibility; mutability; and, long enough period. The last one is becoming more and more important as we aim at undertaking more detailed simulations of more complex systems, which demand longer and longer random number series.

The most popular random number generators are the *congruential* family, due to Lehmer (1951). They follow the recursion

$$x_{n+1} = (ax_n + b) \text{mod } m,$$

$$u_n = x_n/m,$$

for a multiplier $a$, a bias $b$, a module $m$ and a seed $x_0$. When $b = 0$, they are called *multiplicative*. In spite of their apparent simplicity and previsibility, a careful choice of ($a$, $b$, $m$) allows us to provide sufficiently large and random series for many purposes. In such sense, the generator

$$x_{n+1} = (16807x_n) \text{mod } (2^{31} - 1),$$

$$u_n = x_n/(2^{31} - 1),$$

originally proposed by Lewis, Goodman and Miller (1969), has become a minimal standard generator, implemented, e.g. in Press et al (1992).

However, a number of powerful tests have been designed to reject the randomness of congruential generators (L'Ecuyer, 1998). Such tests explode

the reticular structure of the generators. For that reason, and because of the need to have longer period generators (as an example, the minimal standard has period $2^{31} - 2$, which is insufficient for many purposes), many other generators have been designed including perturbations of random number generators, such as shuffling, as in Bays and Durham (1976), shift register generators, Fibonacci lagged generators, nonlinear generators or mixtures of generators. As an example, the Splus statistical package implements, among others, Marsaglia's Super-Duper algorithm, which combines a multiplicative and a Tausworthe generator modified to skip outcomes of zero.

In the basic generators introduced so far, the theoretical properties are easy to analyse because of the highly regular structure. However, this is not desirable in terms of randomness. Recently, methods that combine generators from different families have been proposed. They aim at providing better uniformity properties by reducing the regularity of their structure. Examples of these methods are combinations of linear congruential/multiple recursive or Tausworthe generator with any linear or non-linear generator type through the functions $u_n = (u_{Gen_1} + u_{Gen_2}) \, mod \, 1$ or $u_n = BIN(u_{Gen_1}) \, XOR \, BIN(u_{Gen_2})$, respectively. See L'Ecuyer and Granger-Piche (2003) for a full description and their statistical properties. As an example, one of the most powerful generator is the *Mersenne twister*, proposed by Matsumoto and Nishimura (1998), with period length of $2^{19937} - 1$.

**Pointers to the literature** In general, the practitioner should be really careful with the random number generator provided in the commercial software. They could give wrong answers if the period length is not big enough for a particular application. For example, SUN's Java standard library, available at `http://java.sun.com/j2se/1.3/docs/api/java/util/Random.html`, has period length of $2^{48}$, Visual Basic's is $2^{24}$,... See L'Ecuyer (2001) and McCullough (1999) for a set of tests performed to these popular random number generators.

Good sources of random number generators are at Statlib at `http://www.stat.cmu.edu/`. Other important sites in relation with random number generation are L'Ecuyer's page at `http://www.iro.umontreal.ca/~lecuyer` and `http://random.mat.sbg.ac.at`. A set of statistical tests for random number generators are available at `http://csrc.nist.gov/rng/rng5.html`.

**Case study** In our case study, we used the simulation tool Extend, from Imagine That Inc., which we used as a discrete event simulation environment. The version we used includes as default random number generator the minimum standard and Schrage's (1979) generator as an alternative.

# 3 Random variate generation

The next step in a simulation experiment is to convert the random numbers into inputs appropriate for the model at hand. As an example, in our case study we had to generate variates from a binomial distribution to simulate the number of illegible documents within a batch; variates from a gamma distribution to simulate times in modifying a document, and so on. Random variate generation is essentially based on the combination of six general principles: the inversion method, the composition method, the rejection method, the ratio-of-uniforms method, the use of pretests, the use of transformations and Markov chain Monte Carlo methods.

The most popular traditional method is based on *inversion*. Assuming we are interested in generating from distribution $X$ with distribution function $F$, we have available a source $U$ of random numbers and we have an efficient way to compute $F^{-1}$, the inversion method goes through

`Generate` $U \sim \mathcal{U}(0,1)$
`Output` $X = F^{-1}(U)$

Within Bayesian statistics, see French and Rios Insua (2000), the most intensely used techniques are Markov chain Monte Carlo (MCMC) methods, which assume we have found a Markov chain $\{\theta^n\}$ with state $\theta$ and with its stationary distribution being the (posterior) distribution of interest. The strategy is then to start from arbitrary values of $\theta$, let the Markov chain run until practical convergence, say after $t$ iterations, and use the next $m$ observed values from the chain as an approximate sample from the distribution of interest.

The key question is how to find Markov chains with the desired stationary distribution. There are several generic strategies to design such chains. One of them is the popular Gibbs sampler. Suppose that $\theta = (\theta_1, \ldots, \theta_k)$. The simplest version of the Gibbs sampler requires efficient sampling from the conditional distributions $(\theta_1 \mid \theta_2, ..., \theta_k)$, $(\theta_2 \mid \theta_1, \theta_3, ..., \theta_k)$,..., $(\theta_k \mid \theta_1, \theta_2, ..., \theta_{k-1})$. Starting from arbitrary values, the Gibbs sampler iterates through the

conditionals until convergence:

1. Choose initial values $(\theta_2^0, \ldots, \theta_k^0)$. $i = 1$

2. Until convergence is detected, iterate through
   . Generate $\theta_1^i \sim \theta_1 | \theta_2^{i-1}, .., \theta_k^{i-1}$
   . Generate $\theta_2^i \sim \theta_2 | \theta_1^i, \theta_3^{i-1}, ..., \theta_k^{i-1}$
   . ...
   . Generate $\theta_k^i \sim \theta_k | \theta_1^i, ..., \theta_{k-1}^i$.
   . $i = i + 1$

This sampler is particularly attractive in many scenarios, because the conditional posterior density of one parameter given the others is often relatively simple (perhaps after the introduction of some auxiliary variables). Given its importance, we provide a simple example from Berger and Ríos Insua (1998).

**Example** Suppose the posterior density is

$$p_\theta(\theta_1, \theta_2 \mid x) = \frac{1}{\pi} \exp\{-\theta_1(1 + \theta_2^2)\}$$

over the set $\theta_1 > 0$, $-\infty < \theta_2 < \infty$. The posterior conditional distribution of $\theta_2$, given $\theta_1$, is normal with mean zero and variance $1/2\theta_1$, since

$$p_{\theta_2}(\theta_2 \mid \theta_1, x) \propto p(\theta_1, \theta_2 \mid x) \ \propto \exp(-\theta_1 \theta_2^2).$$

Similarly, given $\theta_2$, $\theta_1$ has exponential distribution with mean $1/(1 + \theta_2^2)$. Then, a Gibbs sampler for this problem iterates through:

1. Choose initial value for $\theta_2$; e.g., the posterior mode, $\theta_2^0 = 0$. $i = 1$

2. Until convergence, iterate through
   . Generate $\theta_1^i = \mathcal{E}/(1 + [\theta_2^{i-1}]^2)$, ($\mathcal{E}$, standard exponential).
   . Generate $\theta_2^i = Z/\sqrt{2\theta_1^i}$, ($Z$, standard normal).

Other strategies to design Markov chains with a desired stationary distribution are the *Metropolis-Hastings algorithm*, the *perfect sampler*, the *slice sampler*, *random direction interior point samplers*,... Complex problems will

typically require a mixture of various MC algorithms, known as hybrid methods. As an example, Müller (1991) suggests using Gibbs sampler steps when conditionals are available for efficient sampling and Metropolis steps otherwise. For variable dimension problems, reversible jump (Green, 1995) and birth-death (Stephens, 2000) samplers are very relevant strategies.

**Pointers to the literature** The literature in modern MCMC methods is vast. Good introductions to Bayesian computations methods may be seen in Smith (1990) and Johnson and Albert (1999). Extensive developments may be seen in French and Ríos Insua (2000), Tanner (1996) and Gamerman (1997). Cheng (1998) contains information about standard methods.

As far as software is concerned, the *Numerical Recipes* (Press *et al*, 1992, or `http://www.nr.com`) include code to generate from the exponential, normal, gamma, Poisson and binomial distributions, from which many other distributions may be sampled based on the principles outlined above. Many generators are available at `http://www.netlib.org/random/index.html`. WINBUGS (Spiegelhalter *et al*, 1994) and OpenBUGS is downloadable from `http://www.mrc-bsu.cam.ac.uk/bugs`, facilitating MCMC sampling in many applied settings. Another useful library is GSL, available at `http://www.gnu.org/software/gsl`.

**Case study** As mentioned, we have used Extend for our case study, which includes samplers for the beta, binomial, constant, Erlang, exponential, gamma, geometric, hyperexponential, discrete uniform, lognormal, normal, Pearson, Poisson, continuous uniform, triangular and Weibull distributions. This allows us to deal with most of the distributions in our model, except for cases like mixtures.

In general, a mixture $f$ is expressed as

$$f(x) = \sum_{i=1}^{n} p_i \cdot g(x|y=i) = \sum_{i=1}^{n} p_i \cdot g_i(x)$$

with $p_i = P(Y=i) > 0$, for all $i = 1, ..., n$, $\sum_i p_i = 1$ and $g_i$, density functions. The procedure to generate from such distribution is

`Generate` $i \sim \begin{pmatrix} p_1 & p_2 & \cdots & p_n \\ 1 & 2 & \cdots & n \end{pmatrix}$

`Output` $X \sim g_i$

# 4 Obtaining model outputs

The third step in a simulation process consists of passing the inputs through the simulation model to obtain outputs to be analysed later. We shall consider the two main application areas in Industrial Statistics: Monte Carlo simulation and discrete event simulation.

## 4.1 Monte Carlo simulation models

A key group of simulation applications in Industrial Statistics use Monte Carlo simulation. By these we understand standard deterministic problems whose analytic solution is too complex, but such that by introducing some stochastic element we are able to obtain a solution with reasonable computational effort. Within statistics, we may use MC methods for optimisation purposes (say to a obtain an MLE or a posterior mode); for resampling purposes, as in the bootstrap; within MC hypothesis tests and confidence intervals; for computations in probabilistic expert systems,... the key application being Monte Carlo integration, specially within Bayesian statistics. We therefore illustrate it in some detail.

Suppose we are interested in computing

$$I_S = \int_{[0,1]^s} f(u)du$$

where $[0,1]^s$ is the $s$-dimensional unit hypercube. We have many numerical methods for such purpose, but they tend to be inefficient as the dimension $s$ grows. As an alternative, we may use simulation based integration methods, or Monte Carlo integration methods, whose (probabilistic) error bound is dimension independent, therefore making them competitive as the integral dimension grows. To wit, note first that

$$I_S = E(f)$$

where the expectation is taken with respect to the uniform distribution. We, therefore, suggest the strategy, based on the Strong Law of Large Numbers:

`Sample` $u_1, \ldots, u_N \sim U[0,1]^s$
`Approximate` $\hat{I}_S = \frac{1}{N}\sum_{i=1}^{N} f(u_i)$

Within Bayesian analysis, we are frequently interested in computing posterior moments as in

$$E_{\theta|x}(g(\theta)) = \int g(\theta)\pi(\theta|x)d\theta.$$

where

$$\pi(\theta|x) = \frac{p(x|\theta)\pi(\theta)}{\int p(x|\theta)\pi(\theta)d\theta},$$

is the posterior distribution for an observation $x$, $\pi(\theta)$ is the prior, $p(x|\theta)$ is the model. As an example, when $g(\theta) = \theta$, we have the posterior mean, whereas when $g(\theta) = I_A(\theta)$, we have the posterior probability of $A$. To compute them, we may use an MC approximation as in

`Sample` $\theta_1, ..., \theta_N \sim \pi(\theta|x)$
`Do` $\widehat{E_{\theta|x}(g(\theta))} = \frac{1}{N}\sum_{i=1}^{N} g(\theta_i)$

Sampling will generally be done through a Markov chain algorithm.

To sum up, assume we want to estimate $\theta = E(X)$ through Monte Carlo integration. If it is simple to sample from $X$, our output process consists of the values $X_i$ sampled from the distribution of $X$. We have that $F_X = F_{X_i}$, so that $F_X = \lim_{i\to\infty} F_{X_i}$ and $\theta = E_{F_X}(X)$. If, on the other hand, it is not easy to sample from $X$, we may define a Markov chain $X_i \xrightarrow{d} X$, so that $F_X = \lim_{i\to\infty} F_{X_i}$, our output process being again $X_i$.

**Example** Following with our Gibbs sampler example, typically, we shall be interested in approximating posterior expectations of functions $g(\theta_1, \theta_2)$ through

$$\begin{aligned} E_\theta[g(\theta_1, \theta_2) \mid x] &= \int_{-\infty}^{\infty}\int_{0}^{\infty} g(\theta_1, \theta_2)p(\theta_1, \theta_2 \mid x)d\theta_1 d\theta_2 \\ &\cong \frac{1}{m}\sum_{i=1}^{m} g(\theta_1^i, \theta_2^i). \end{aligned}$$

For example, under squared error loss, we estimate $\theta_1$ by its posterior mean, approximated by $\hat{\theta}_1 = \frac{1}{m}\sum_{i=1}^{m}\theta_1^i$. The output process in this case will be $(\theta_1^i, \theta_2^i)$.

## 4.2 Discrete event simulation models

The other big application area refers to discrete event simulation (DES), which deals with systems whose state changes at discrete times, not continuously. These methods were initiated in the late 50's; for example, the first DES-specific language, GSP, was developed at General Electric by Tocher and Owen to study manufacturing problems. To study such systems, we build a discrete event model. Its evolution in time implies changes in the attributes of one of its entities, or model components, and it takes place in a given instant. Such change is called event. The time between two instants is an interval. A process describes the sequence of states of an entity throughout its life in the system.

There are several strategies to describe such evolution, which depend on the mechanism that regulates time evolution within the system. When such evolution is based on time increments of the same duration, we talk about *synchronous simulation*. When the evolution is based on intervals, we talk about *asynchronous simulation*.

We illustrate both strategies describing how to sample from a Markov chain with state space $S$ and transition matrix $P = (p_{ij})$, with $p_{ij} = P(X_{n+1} = j | X_n = i)$. The obvious way to simulate the $(n+1)$-th transition, given $X_n$, is

`Generate` $X_{n+1} \sim \{p_{x_n j} : j \in S\}$

This synchronous approach has the potential shortcoming that $X_n = X_{n+1}$, with the corresponding computational effort lost. Alternatively, we may simulate $T_n$, the time until the next change of state and, then, sample the new state $X_{n+T_n}$. If $X_n = s$, $T_n$ follows a geometric distribution of parameter $p_{ss}$ and $X_{n+T_n}$ will have a discrete distribution with mass function $\{p_{sj}/(1-p_{ss}) : j \in S \setminus \{s\}\}$. Should we wish to sample $N$ transitions of the chain, assuming $X_0 = i_0$, we do

```
Do t = 0, X_0 = i_0
   While t < N
       Sample h ~ Ge(p_{x_t x_t})
       Sample  X_{t+h} ~ {p_{x_t j}/(1 - p_{x_t x_t}) : j ∈ S \ {x_t}}
       Do  t = t + h
```

There are two key strategies for asynchronous simulation. One is that of event scheduling. The simulation time advances until the next event and the corresponding activities are executed. If we have $k$ types of events $(1, 2, ..., k)$, we maintain a list of events, ordered according to their execution times $(t_1, t_2, ..., t_k)$. A routine $R_i$ associated with the $i$-th type of event is started at time $\tau_i = \min(t_1, t_2, ..., t_k)$. An alternative strategy is that of *process interaction*; a process represents an entity and the set of actions that experiments throughout its life within the model. The system behaviour may be described as a set of processes that interact, for example, competing for limited resources. A list of processes is maintained, ordered according to the occurrence of the next event. Processes may be interrupted, having their routines multiple entry points, designated reactivation points.

Each execution of the program will correspond to a replication, which corresponds to simulating the system behaviour for a long enough period of time, providing average performance measures, say $X_n$, after $n$ customers have been processed. If the system is stable $X_n \xrightarrow{w} X$. If, e.g., processing 1000 jobs is considered long enough, we associate with each replication $j$ of the experiment the output $X^j_{1000}$. After several replications, we would analyse the results as described in the next section.

### 4.2.1 Discrete event simulation software

The implementation of complex simulation models with several types of events and processes in standard programming languages may be very involved. This would explain the emergence of numerous simulation environments, as shown in the recent software simulation review in OR/MS Today (2003), which is periodically adapted. These include simulation languages and simulators.

Simulation languages are general purpose languages which include tools and utilities specific for simulation, such as

- A general framework to create and describe a model in terms of processes, events, entities, attributes, resources and interactions between model components.
- A mechanism to control the evolution of time.
- Methods to schedule event occurrences.
- Random number and variate generators.
- Tools to collect and analyse data.
- Tools to describe and display graphically the model and its simulation.
- Tools to validate and verify the model.

Simulators are software packages which allow for the simulation of complex models in specific application areas, such as manufacturing, supply chain management, material handling, workflow management, of interest in industrial statistics. Examples of this packages are *Arena*, *Taylor II*, *Extend*, *AutoMod*,...

**Case study** We now illustrate the implementation of our reference example in a DES environment, specifically in Extend. In the DES terminology, the processes that we shall consider are the lots and the lot arrival generator and the resources will be the scanners, the program, the operators and their workstations. To facilitate implementation we associate seven numerical attributes, which establish the different types of documents, that a lot may include within its life in the system:

- Lot headers (R).
- Bills and receipts (L).
- Promissory notes (P).
- Bill and receipts wrongly scanned or illegible (LC).
- Bills and receipts wrongly recognised (LD).
- Promissory notes wrongly scanned or illegible (PC).

- Promissory notes wrongly recognised (PD).

For each stage in the line, we must model the process occurring at that stage, the involved random processes, the parameters to be estimated and the controllable parameters. As an example, we provide that of the scanning process:

- *Description.* Scanning is done by lots. Once scanned, there is a waiting period of around 10 seconds, before we may feed them in a new lot. The technical specifications suggest a scanning speed of 4.17 documents per second.

- *Random processes.* We need to consider the failure distribution for the scanner. We use a binomial model for each type of document, $x \sim BIN(L, P_s)$ , $y \sim BIN(P, P_s)$ (where $P_s$, is the probability of a document incorrectly scanned, $x$ is the number of illegible bills and $y$ is the number of illegible promissory notes). The estimated scanning time per lot is, therefore: $\frac{L+P+R}{4.17} + 10$ seconds.

- *Parameter estimation* We use a beta-binomial to estimate $P_s$. With a uniform prior, the posterior is beta (84, 9916), the posterior mode being $\hat{P}_s = 0.0085$.

- *Controllable parameters* One person and two scanners are used in this phase. One of the scanners is for backup.

The above stage may be described in Extend as follows The first block in the figure is called (in the Extend jargon) Dequation, which computes an expression given the input values; in this case it will compute the scanning time for the lot, as indicated above. Then, the items go through an element called Queue-Fifo, which represents a queue in which we store and leave elements according to a FIFO strategy. Then, they go through an Activity Delay, which will keep elements for some time, in this case the scanning time. Then, the item leaves that block and goes through a Dequation used to obtain the time that the element remained in queue before leaving the Activity Delay. Items go afterwards through a Dequation which computes the attribute LC, through a binomial model. The next Dequation modifies the attribute L. The next two Dequation blocks affect PC and P.

The whole line is described in the following diagram

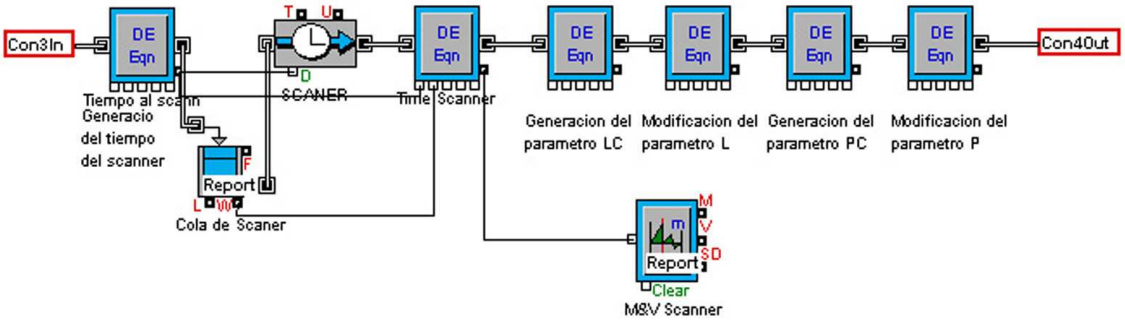


Figure 1: Block modelling the scanning stage

Clearly, this is too complicated to describe to a practicioner, but we may use appropriate icons to summarise the model as presented in the diagram

Each run of this simulation model will provide outputs in relation to a day of operation.

# 5 Output analysis

The final stage of a simulation experiment consists of the analysis of the output obtained through the experiment. To a large extent, we may use standard estimation methods, point estimates and precision measures, with the key observation that the output might be correlated. Clearly, as we deal with stochastic models, each repetition of the experiment will lead to a different result, provided that we use different seeds to initialise the random number generators at each replication.

The general issue here is to provide information about some performance measure $\theta$ of our system. We shall only comment in some detail the case of univariate performance measures. We assume that the simulation experiment provides us with an output process $X = \{X_i\}$, so that $\theta$ is a property of the limit distribution $F_X = \lim_{i\to\infty} F_{X_i}$. In fact, most of the times, we shall be able to redefine the output process so that $\theta$ is the expected value of $X$, that is, $\int_{-\infty}^{\infty} x dF_X(x)$, or the $p$-th quantile, that is, the value $x_p$ such that $F_X(x_p) = p$, for $p \in (0, 1)$. For example, if we are interested in estimating the covariance between two variables $X$, $Y$, $\theta = \sigma_{XY} = \int\int (X - \mu_X)(Y - \mu_Y) dF_{XY}(x, y)$,

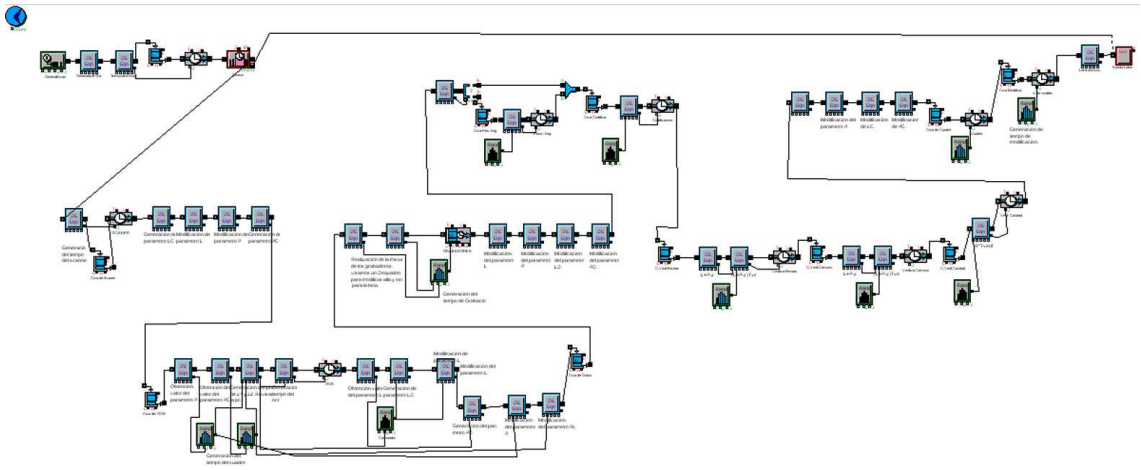

Figure 2: The whole workflow line in Extend

we may define the bivariate output process $\left\{(X_i - \bar{X}), (Y_i - \bar{Y})\right\}$, where $\bar{X}$, $\bar{Y}$ are the sample means of $X$, $Y$. Sometimes, we are interested in estimating the whole distribution of a system performance measure, which may be done by estimating the parameters of a (parametric) distribution. Alternatively, we may estimate a sufficient number of quantiles so as to approximate the distribution function.

Another important issue is the distinction between stationary and transition behaviour. In fact, it determines the way to carry out the simulation experiment and the way of analysing data. Transition behaviour refers to short-term system performance. In our example, we are interested in whether we shall be able to complete the workload before the deadline. Stationary behaviour refers to long term performance. In our example, we are interested in the fraction of bags not processed on time due to the system being busy, or we may be interested in determining the long-term fraction of lost messages due to the system being saturated (see e.g. Conti et al., 2004).

## 5.1 Point estimation

As far as point estimation is concerned, standard methods and concepts like unbiasedness and consistency apply. For obvious reasons, the concept of asymptotic unbiasedness is specially relevant, when dealing with stationary

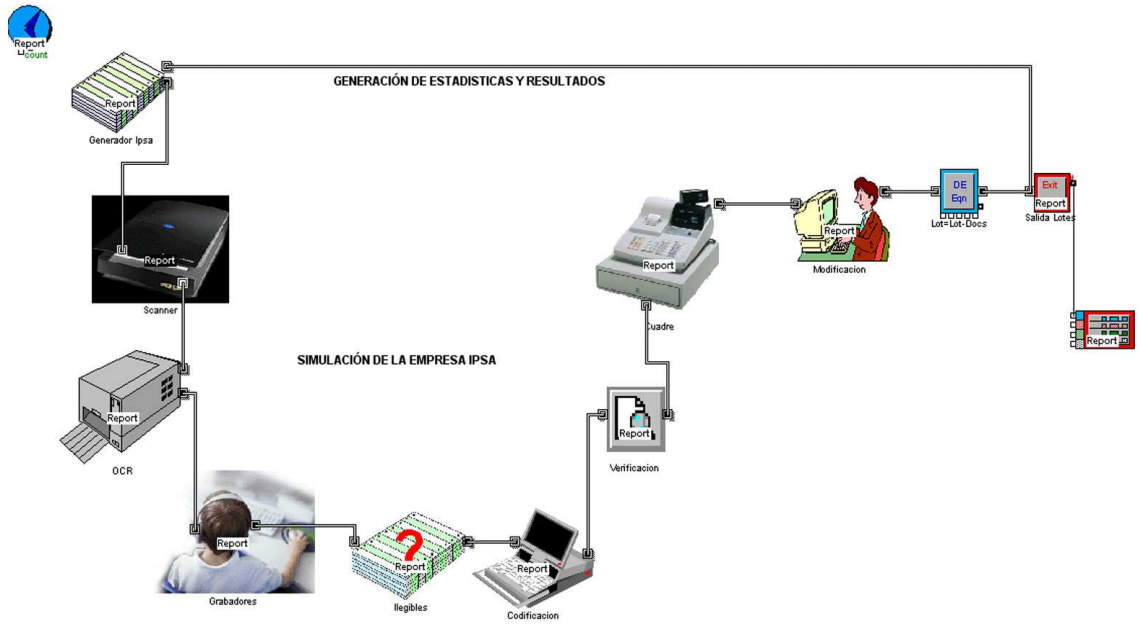


Figure 3: The workflow line

performance.

In that sense, in order to estimate the mean $\mu_X = \lim_{i\to\infty} \mu_{X_i}$, we shall normally use the sample mean

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

To estimate proportions or probabilities, we use the sample mean of the indicators of the event of interest. Specifically, if $A$ is such event, we define $Z_i = I_{\{X_i \in A\}}$, and use

$$\bar{Z} = \frac{\sum_{i=1}^{n} Z_i}{n} = \frac{\#\{X_i \in A\}}{n}$$

to estimate $P(X \in A)$. In order to estimate the variance

$$\sigma_X^2 = \int_{-\infty}^{\infty} (x - \mu_X)^2 dF_X(x),$$

we shall compute $\bar{X}$, $\{(X_i - \bar{X})^2\}$ and use

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

which is unbiased for iid observations, and asymptotically unbiased in general; for small samples, and correlated output we may use various methods to correct the bias.

As far as quantiles are concerned, if $\{X_{(i)}\}$ is the order statistic associated with the output $\{X_i\}$, a simple estimator of $F_X^{-1}(p)$ is

$$(1-\alpha)X_{(r)} + \alpha X_{(r+1)},$$

with $\alpha = p(n+1) - int(p(n+1))$ and $r = int(p(n+1))$. An alternative for simultaneous estimation of several quantiles is to use the histogram. Although we loose some information while computing the counts in the histogram cells, we usually obtain good results with small cells, in spite of the inconsistency of histograms of fixed width, see Hartigan (1996) for references on histograms.

Before finishing this section, we would like to point out three possible sources of bias in these estimators: the initial transition (linked with the problem of convergence detection, see Cowles and Carlin (1996) for a review), nonlinearity of transformations and random sample sizes.

## 5.2 Precision estimation

We also need to estimate the precision of the estimator. We shall use the mean square error which, when the bias is negligible, will coincide with the variance. As basic measure, we shall use the standard deviation of $\hat{\theta}$, $EE(\hat{\theta}) = (Var(\hat{\theta}))^{1/2}$ and we aim at estimating $\widehat{EE}(\hat{\theta})$ or, equivalently, $\widehat{Var}(\hat{\theta})$. In such a way, when we say that $\hat{\theta} = 16.3289$ with $\widehat{EE}(\hat{\theta}) = .1624$, we may find meaningful 16, give some validity to .3 (for example, 16.3 is more meaningful than 15.8) and consider 0.0289 as random digits.

In the iid case, we use the standard variance estimation theory. For example, when $\hat{\theta} = \bar{X}$, with fixed $n$, $Var(\bar{X}) = \frac{Var(X_i)}{n}$ and an estimator is $S^2/n$, where

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

is the sample variance. If $N$ is random and independent of the observations, $Var(\bar{X}) = \frac{Var(X_i)}{E(N)}$ and an unbiased estimator is $\hat{V}_0 = S^2/N$, with $S^2$ as before and $N$ in place of $n$, with $N \geq 2$.

If we are estimating $p = P(A)$, we use

$$\hat{\theta} = \hat{p} = \frac{\sum_{i=1}^N I_{\{X_i \in A\}}}{N}$$

with $N$, possibly random, but independent of the observations. We have that $Var(\hat{p}) = (p(1-p))/E(N)$ and

$$\hat{Var}(\hat{p}) = \frac{\hat{p}(1-\hat{p})}{N-1}.$$

is an unbiased estimator.

Similarly, we would proceed for other estimators. A shortcoming of this approach is that it is, somewhat, ad hoc in the sense that we need to develop methods for each estimator. An alternative general (and popular) variance estimation method in simulation is that of macro-micro replications. Given $n$ replications, we actually assume that it consists of $k$ independent macroreplications with $m$ micro replications $(X_{1j}, ..., X_{mj})$, $j = 1, ..., k$, and $km = n$. Each micro replication provides an observation of the output process; each macro replication provides an estimator $\hat{\theta}_j$, $j = 1, ..., k$, based on the $m$ observations of such replication, with the same expression as $\hat{\theta}$. The mean of the $k$ macro replications, that is,

$$\bar{\theta} = \frac{1}{k}\sum_{j=1}^{k}\hat{\theta}_j,$$

is an alternative to the estimator $\hat{\theta}$. Clearly, when $m = n$, $k = 1$, $\hat{\theta} = \bar{\theta}$. As $\bar{\theta}$ is a sample mean, its variance will be estimated through

$$\hat{V}_1 = \frac{1}{k}\frac{\sum_{j=1}^{k}\hat{\theta}_j^2 - k\bar{\theta}^2}{k-1}.$$

For a discussion on how to choose $m$ and $k$, see Schmeiser (1990). Recall also, that the bootstrap and the jackknife, see Efron (1982), provide methods to estimate the variance.

## 5.3 Dependent output

We need to be specially careful when simulation output is correlated, as it happens in an MCMC sampler. To illustrate the issues involved, assume that $(X_1, ..., X_n)$ are observations from a stationary process and we estimate $\theta$ through $\bar{X}$. If $Var(X) = \sigma_X^2$ and $\rho_j = Corr(X_i, X_{i+j})$ we have

$$Var(\bar{X}) = \frac{d\,\sigma_X^2}{n}$$

with

$$d = 1 + 2\sum_{j=1}^{n-1}\left(1 - \frac{j}{n}\right)\rho_j.$$

In the i.i.d., $d = 1$. When the process is positively correlated $Var(\bar{X}) > \sigma^2/n$. Moreover,

$$E\left(\frac{S^2}{n}\right) = \frac{e\,\sigma_X^2}{n}$$

with

$$e = 1 - \frac{2}{n-1}\sum_{j=1}^{n-1}\left(1 - \frac{j}{n}\right)\rho_j$$

so that we underestimate the variability of $\bar{X}$. Similarly, if the process in negatively correlated, we shall overestimate it.

To mitigate the problem, several methods have been devised. The most popular is that of macro-micro replications, known as batch method for dependent data; correlation substitution; time series methods; regenerative simulation; and, thinning. Further details may be seen in Balci (1994) and Ríos Insua et al (1997).

## 5.4 Confidence intervals

We shall usually employ $Var(\hat{\theta})$ to obtain a confidence interval for $\theta$: we aim at using the output process $(X_1, ..., X_n)$ to obtain a random interval $[L, U]$ so that $P(\{L \leq \theta \leq U\}) = 1 - \alpha$ for some $\alpha$. The standard confidence interval theory applies, with convenient modifications to take into account the issue of dependent data and the various methods used to estimate the precision.

**Pointers to the literature** We have concentrated on analysing only one performance measure. Multiple performance measures require special care when defining multiple confidence intervals, which is done through Bonferroni type inequalities, see Rios Insua et al (1996) for further details.

Special care must be adopted when using simulation software, since, rather frequently, it provides precision measures under the hypothesis of independence. Beware. Alternatively, you may use your favourite statistical environment to undertake this task, if it allows for time series analysis data, after preparation of simple programs. Convergence detection is an important issue, for which CODA `http://www.mrc-bsu.cam.ac.uk/bugs/classic/coda04/readme.shtml`

is useful. Fishman (1996) provides ample information about his LABATCH package available at `http://www.unc.edu/ gfish/labatch.2.html`. The MCMCpack also contains some useful utility functions, and is available at `http://www.r-project.org`.

**The example** Assume we want to approximate $I = \int_{-\infty}^{\infty}(x + x^2)f(x)dx$, where $f$ is a normal density with mean 1 and standard deviation 2. We used Monte Carlo integration and repeated the experiment 50 times with sample size 50 at each iteration. The obtained values are

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 6.769 | 5.603 | 5.614 | 9.229 | 7.189 | 3.277 | 4.312 | 7.070 | 5.195 | 4.496 |
| 5.775 | 4.646 | 5.670 | 7.134 | 4.931 | 4.403 | 6.783 | 7.152 | 5.834 | 4.958 |
| 7.159 | 7.270 | 8.379 | 5.037 | 5.143 | 5.757 | 7.399 | 5.236 | 4.749 | 5.729 |
| 7.015 | 6.156 | 3.985 | 5.643 | 5.720 | 6.878 | 6.367 | 7.520 | 7.093 | 6.605 |
| 6.356 | 6.567 | 7.784 | 5.256 | 6.302 | 5.460 | 4.808 | 5.880 | 3.846 | 5.962 |

Table: approximations of $I$

Note that if we use just one replication, we may obtain very disparate values, from 3.277 to 9.229. We use the sample mean as estimator, which is, 5.983. The mean square error is 1.475. Note that $I = 6$.

**Case study** As we indicated, our interest is in estimating the probability that the process is completed before the deadline, in our case 28800 seconds, that is we need to estimate $Pr(T \leq 28800)$ for each configuration line. For that, we shall perform 1000 replications, equivalent to 1000 days of operation, and we record whether the process finishes before 28800 seconds. The output process is therefore $N_i$, where $N_i = 1$ if $T_i \leq 28800$ and 0 otherwise, where $T_i$ is the final time of the $i$-th iteration. Therefore, we estimate the probability through

$$\frac{\sum_{i=1}^{1000} N_i}{1000}$$

To complete the study, we record other output values, specifically average waiting times at each queue and average times at each process, twenty two outputs on the whole.

We show below the statistics generated by Extend after simulating 1000 days under the current configuration,

- Probability of finishing work on time: 0.441

- Average finishing time: 31221.1 sec.

| Device | Lot Av. que. time | Lot Av. resp. time |
|---|---|---|
| Scanner | 5.93 | 41.03 |
| OCR | 3779.69 | 3861. 32 |
| Recording | 0 | 1888.72 |
| Illegible resolution | 0.75 | 11.21 |
| Coding | 961.33 | 1038.89 |
| Date verification | 59.83 | 87.23 |
| Field verification | 6183.34 | 6325.18 |
| Quality verification | 44.79 | 1836. 54 |
| Balancing | 1.75 | 1.77 |
| Modification | 5.64 | 123.23 |

Note that the probability of timely completion is too low (0.44) and the average finishing time (31221) is much bigger than the allowed one (the respective mean square errors, were 0.12 and 925). Note that the biggest waiting times hold for the OCR and field verification, the main bottlenecks.

We, therefore, reconfigure the system, doubling the resources in the OCR phase and field verification. This leads to the following results:

- Probability of finishing process on time: 0.941
- Average finishing process time: 21881.1 sec.

| Device | Lot Av. que. time | Lot Av. resp. time |
|---|---|---|
| Scanner | 6.05 | 41.15 |
| OCR | 454.17 | 541.34 |
| Recording | 369.27 | 2465.18 |
| Illegible resolution | 1.79 | 12.18 |
| Coding | 2608.79 | 2685.28 |
| Date verification | 83.29 | 110. 64 |
| Field verification | 1351.67 | 1478.59 |
| Quality verification | 2179.06 | 2280.67 |
| Balancing | 0.154 | 1.73 |
| Modification | 68.93 | 131.21 |

As we see, the modifications induce a reduction in the modified stages and, more importantly, have been sufficient to guarantee completion of the process (probability .94). Note, as well, that the average queueing times have been balanced, therefore attaining a more balanced load. Note that some processes, like recording, have increased their time. However, the design is globally much better.

As we have indicated, the previous results allow us to answer the initial question. To complete the study, we could investigate the performance under more extreme work conditions. For example, we could study the possibility of expanding our business, looking for new customers.

As an example, suppose that five vans od documents arrive every day (the current workload is randomly distributed between one and four). The resulting statistics would be:

- Probability of finishing on time: 0.26
- Average finishing time: 30569

| Device | Lot Av. que. time | Lot Av. Resp. time |
|---|---:|---:|
| Scanner | 5.81 | 40.91 |
| OCR | 638.89 | 723.92 |
| Recording | 629.56 | 2748.63 |
| Illegible resolution | 1.91 | 12.51 |
| Coding | 3766.47 | 3842.59 |
| Date verification | 87.56 | 114.95 |
| Field verification | 1823.13 | 1952.27 |
| Quality verification | 3228.38 | 3330.5 |
| Balancing | 0.13 | 1.84 |
| Modification | 72.99 | 135.37 |

As we see under such extreme conditions, the average processing time is much worse and there is little guarantee of completing the process on time (0.264). Hence it does not seem advisable to expand business so much.

# 6 Tactical questions in simulation

We end up this review by pointing out issues concerning tactical questions in simulation: how do we design simulation experiments, how do we combine simulation with optimisation; the issue of variance reduction.

## 6.1 The number of iterations

We briefly discuss now the issue of determining the appropriate number $n$ of replications, which, on one hand, will affect the estimation precision and, on the other, will affect computational cost. As normally, variance estimators are $O(n^{-1})$, the basic idea will be to take $n$ sufficiently large to guarantee a certain precision. Note that an initial idea about $n$ somehow limits our computational effort, whereas if we view it excessive, we may try to apply variance reduction techniques. We briefly illustrate the ideas in the simplest case.

Suppose we may observe $n$ times the output process $X_i$ and $X_1, ..., X_n$ are i.i.d. with $E(X) = \theta$, $Var(X) = \sigma^2$. We use $\bar{X} = (1/n)\sum_{i=1}^{n} X_i$ and $S/\sqrt{n}$, with $S^2 = (1/(n-1))\sum_{i=1}^{n}(X_i - \bar{X})^2$, respectively as the estimator and the precision of the estimator. To determine $n$, we fix $1-\alpha$ as an acceptable confidence level. For $n$ sufficiently large

$$\left[\bar{X} - z_{\alpha/2}\frac{S}{\sqrt{n}},\ \bar{X} + z_{\alpha/2}\frac{S}{\sqrt{n}}\right]$$

is a confidence interval of level $(1-\alpha)$ and width $2z_{\alpha/2}S/\sqrt{n}$. If we fix the maximum allowed width as $d$, we just need to do

$$2z_{\alpha/2}\frac{S}{\sqrt{n}} \ \leq \ d$$

or

$$\left(2z_{\alpha/2}\frac{S}{d}\right)^2 \ \leq \ n.$$

Sometimes we have an initial idea about $\sigma$ or $S$, which we may use to suggest an initial size. If this is not the case, we may use a pilot sample to estimate $S$ and apply the previous method; typically we shall need to iterate with an argument such as follows, where $S_n$ is the sample variance when the size is $n$

`Do` $n_0 = 30$
`Generate` $\{X_j\}_{j=1}^{30}$
`Compute` $S_{30},\ i = 0$
`While` $2z_{\alpha/2}S_{n_i}/\sqrt{n_i} > d$
    $i = i + 1$

`Compute min` $n_i$ : $\left(2z_{\alpha/2}\frac{S_{n_{i-1}}}{d}\right)^2 \leq n_i$
`Generate` $\{X_j\}_{j=n_{i-1}+1}^{n_i}$
`Compute` $S_{n_i}$

**Example** We apply the previous procedure to the case in which we try to estimate

$$\int_{-\infty}^{\infty}(x+x^2)f(x)dx,$$

with $f$ the normal density $N(1,4)$. The table includes the results when $\alpha = 0.005$ and $d = 0.1$.

| $n$ | $\hat{I}$ | $S$ | Width |
|---|---|---|---|
| 30 | 5.06 | 5.22 | 5.216 |
| 85849 | 5.996 | 8.2 | .158 |
| 212521 | 6.005 | 8.27 | .101 |
| 216225 | 6.003 | 8.26 | .099 |

For the general case, we shall use confidence intervals

$$\left[\hat{\theta} - t_{\nu,\alpha/2}\sqrt{\hat{V}},\ \hat{\theta} + t_{\nu,\alpha/2}\sqrt{\hat{V}}\right]$$

where $\hat{\theta}$ is the estimator of interest and $\hat{V}$ is a variance estimator with $\chi^2$ distribution with $\nu$ degrees of freedom and independent of the distribution of $\hat{\theta}$. Typically, $\hat{V}$ depends on $n$, that is $\hat{V} = \hat{V}(n)$. Moreover, $\nu = \nu(n)$ frequently, so that

$$2t_{\nu(n),\alpha/2}\hat{V}(n) \leq d$$

which we solve for $n$. The former sequential procedure may be extended easily, specially if the distribution of $t_{\nu(n)}$ may be approximated through a normal distribution with $\nu(n) \geq 30$.

## 6.2 Regression metamodels

In most simulation based industrial statistical applications, we shall be interested in either understand the behaviour of a system, i.e. how changes in operation conditions affect performance, or improve its functioning. Algebraically, we describe the relation of the output $z_0$ of the real system with the inputs $y_1, y_2, ...$ through a function

$$z_0 = f_0(y_1, y_2, ...; R_0)$$

where $R_0$ designates the sources of randomness in a generic form. We identify those inputs that we consider relevant $y_1, y_2, ..., y_k$ and describe the relation between the model output $z_1$ and the inputs through the function

$$z_1 = f_1(y_1, y_2, ..., y_k; R_1)$$

where $R_1$ designates the randomness sources of the model, our objective being to estimate

$$\theta(y_1, ..., y_k) = E_{R_1}(f_1(y_1, ..., y_k; R_1))$$

when computing such expectation is difficult we may appeal to simulation to estimate $\theta(y_1, ..., y_k)$. For such purpose, the relation between the output $(X_i)_{i=1}^n$ and the inputs through a simulation program, is through a function

$$X(y_1, ..., y_k) = f_2(y_1, ..., y_k; R_2)$$

where $R_2$ designates the random seeds used to initialise the random number generators.

The tools previously described allow us to estimate the performance $\theta(y_1, ..., y_k)$, given the inputs, together with a precision measure. Analogously, should we be interested in determining optimal inputs we should solve

$$\min\ \theta(y_1, ..., y_k)$$

under appropriate constraints. One possibility would be to use an optimisation algorithm requiring only function evaluations, such as Nelder Mead's, estimating the function at each new input value, through the simulation algorithm, see Ríos Insua et al (1997), for an example.

This approximation may be extremely costly from a computational point of view. In this case, it may be more interesting to associate to the problem a new model called *regression metamodel*. For that, we just need to represent the estimation problem in such a way that if $Z_3(y_1, ..., y_k) = \hat{\theta}(y_1, ..., y_k)$ we introduce the representation

$$Z_3 = f_3(y_1, ..., y_k, \delta) + \epsilon \tag{2}$$

where $f_3$ represents a parametric function with parameters $\delta$ to be estimated and an error term $\epsilon$. An example, based on neural networks may be seen in Muller and Ríos Insua (1998). We may then use the regression metamodel for prediction and optimization purposes as wished.

The optimisation problem is described as

$$\begin{aligned} \min\ \theta_0 &= E_R(f_0(y_1, ..., y_k; R)) \\ \text{s.t.}\quad \theta_i &= E_R(f_i(y_1, ..., y_k; R)) \leq b_i,\ i = 1, ..., r \\ & (y_1, ..., y_k) \in S \end{aligned}$$

where $R$ designates the randomness sources, $r$ designate output constraints, and the last constraint refers to inputs. We have a simulation model with $m$ responses of interest $\hat{\theta}_i = f_i(y_1, ..., y_k; \underline{R}),\ i = 0, 1, ..., r$, where, as before, $\underline{R}$ designates the employed random numbers.

We end up this section mentioning that a number of specific simulation-optimisation methods have been developed. In the case of finite sets of alternatives, we should mention, on one hand, methods based on ranking and selection and, on the other, based on multiple comparisons. Among methods for continuous problems, we should mention response surface methods and stochastic approximation methods such as Robbins-Monro (1951) and Kiefer-Wolfowitz (1952). Other methods include algorithms based on perturbation analysis (Glasserman, 1991) and on likelihood ratios (Kleijnen and Rubinstein, 1996).

## 6.3 Experimental design and simulation

We have described simulation as a (computer based) experimental methodology. As such all principles for good experimentation, as reviewed in the design of experiments chapter in this book, seem relevant. Further details may be seen in Box *et al* (1978) and Chaloner and Verdinelli (1995).

An important difference with other types of experiments refers to the key point that we control the source of randomness, and we may take advantage of it. As an example, in simulation we have the common random number technique, which uses the same random numbers for simulations under different input conditions.

## 6.4 Variance reduction

We have emphasised the need to have quality measures of simulation estimators through precision estimates. In such respect, it seems natural to improve the quality of estimators, typically looking for estimators with similar bias

but smaller variance. The techniques addressed towards such purpose are called variance reduction techniques.

Given a basic simulation experiment, the idea is to introduce another experiment, in which sampling is done differently or in which we observe a different output variable, which leads to a better quality estimator. A trivial observation is that, we may reduce the variance by augmenting the sample size $n$, as we have seen that variance estimators are $O(n^{-1})$. But this entails an increase in computational effort, which may be unacceptable in many cases. The objective would be, on the other hand, to reduce the mean squared error, keeping the computational effort, or reduce the computational effort, keeping the mean squared error.

For that purpose, several techniques have been developed including antithetic variates, control variates, conditioning, importance sampling, common random numbers and stratified sampling. Computational savings may be tremendous, however its application is far from simple, frequently demanding ingenuity and small pilot samples to ascertain whether we may achieve, in effect, a variance reduction.

We end up this section mentioning that, again, variance reduction is a topic which is in the border between simulation and experimental design.

# 7 Discussion and conclusions

We have provided an illustrated introduction to key concepts and methods in simulation, as far as Industrial Statistics is concerned. Further details may be seen in various texts including Fishman (1996, 2000), Rios Insua et al (1996), Banks (1998) or Schmeiser (1990). We hope that these lines will provide the reader with a broad view of simulation methods and applications to eventually apply them at his industrial problem at hand.

Discrete event simulation is one of the most used techniques in Business and Industrial Statistics in problems such as manufacturing systems, LAN modelling,... in which performance of a system whose state evolves discretely in time may not be computed analytically. The standard approach proceeds by building a simulation model; estimating the model parameters; plugging the estimates into the model; running the model to forecast performance evaluation and analysing the output. However, although this has not been acknowledged in the DES literature, this approach typically greatly underestimates uncertainty in predictions, since the uncertainty in the model para-

meters is not taken into account, by assuming parameters fixed at estimated values. In other fields, this issue of input uncertainty influencing model uncertainty has generated a relevant literature, see, for example, Draper (1995) or Chick (2001). Applying Bayesian methods in discrete event simulation seems a fruitful area of research.

Note that we have not practically mentioned continuous time simulations, typically based on stochastic differential equations. Typically, a synchronous approach will be adopted. See Neelamkavil (1987) for further information.

We finally provide some additional pointers to urls of interest, referring to on-line executable simulations, `http://www.cis.ufl.edu/~fishwick/websim.html` and `http://www.national.com/appinfo/power/webench/websim/`

# Acknowledgments

This work was supported by funds under the European Commission's Fifth Framework 'Growth Programme' via Thematic Network "Pro-ENBIS", contract reference G6RT-CT-2001-05059, and by grants from MEC and the DMR Consulting Foundation.

# REFERENCES

Atkinson, A.C. and Donev, A.N. (1992). *Optimum Experimental Designs*, Oxford University Press, Oxford.

Balci, O. (ed) (1994). Simulation and Modelling, *Annals of Operations Research*, 53.

Banks, J. (1998). *Handbook of Simulation*, Wiley, New York.

Bayarri, M.J., Berger, J.O., Garcia-Donato, G., Palomo, J., Sacks, J. and Walsh, D. (2004). Computer model validation with functional output, *Tech. Rep.* NISS, Research Triangle Park.

Bays, C. and Durham, S. (1976). Improving a poor random number generator, *ACM Transactions in Mathematical Software*, 2, 59-64.

Berger, J.O., Garcia-Donato, G. and Palomo, J. (2004). Validation of Complex Computer Models with Multivariate Functional Outputs, *Tech. Rep.* SAMSI, Durham.

Berger, J.O. and Ríos Insua, D. (1998). Recent developments in Bayesian Inference, with applications in Hydrology, *Statistical and Bayesian Methods in Hydrology*, UNESCO Press.

Box, G.E.P., Hunter, W.G. and Hunter, J.S. (1978). *Statistics for Experimenters*, Wiley, New York.

Bratley, P., Fox, B. and Schrage, L. (1987). *A Guide to Simulation*, Springer, New York.

Chaloner, K. and Verdinelli, I. (1995). Bayesian experimental design: A review, *Statistical Science*, 10, 273-304.

Chick, S.E. (2001). Input Distribution Selection for Simulation Experiments: Accounting for Input Uncertainty, *Operations Research*, 49, 744-758.

Conti, P.L., Lijoi, A. and Ruggeri, F. (2004). A Bayesian approach to the analysis of telecommunication systems performance, *Applied Stochastic Models in Business and Industry*, 20, 305-321.

Cowles, K. and Carlin B. (1996). Markov chain Monte Carlo convergence diagnostics, *Journal of the American Statistical Association*, 91, 883-904.

Draper (1995). Assessment and propagation of model uncertainty (with discussion), *Journal of the Royal Statistical Society, B*, 57, 45-97.

Efron, B. (1982). *The Jackknife, the Bootstrap and other Resampling Plans*, SIAM, Philadelphia.

Extend, official web page. *http://www.imaginethatinc.com*

Fishman, G. S. (1996). *Monte Carlo: Concepts, Algorithms and Applications*, Springer, New York.

French, S. and Ríos Insua, D. (2000). *Statistical Decision Theory*, Arnold, London.

Fu, M.C. (1994). Optimization via simulation, *Annals of Operations Research*, 53, 199-248.

Gamerman, D. (1997). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, Chapman & Hall, New York.

Glasserman, P.(1991). *Gradient Estimation via Perturbation Analysis*, Kluwer, Boston.

Green, P. (1995). Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination, *Biometrika*, 82, 711-732.

Hartigan, J. (1996). Bayesian histograms., in *Bayesian statistics 5* (J. Bernardo, J. Berger, A. Dawid, and A. Smith eds.), 211-222, Oxford University Press, Oxford.

Imaginethatinc (2000). *Extend Manual*, Imagine That Inc.

Johnson, V.E. and Albert, J.H. (1999). *Ordinal Data Modeling*, Springer, New York.

Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function, *Annals of Mathematical Statistics*, 23, 462-466.

Kleijnen, J.P.C. (1987). *Statistical Tools for Simulation Practitioners*, Dekker, New York.

Kleijnen, J.P.C. and Rubinstein, R.Y. (1996). Sensitivity Analysis by the Score Function Method, *European Journal of Operations Research*, 88, 413-427.

Knuth, D. (1981). *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, Addison Wesley, New York.

Law, A. M. and Kelton, W. D. (1991). *Simulation Modeling and Analysis*, McGraw-Hill, New York.

L'Ecuyer, P. (1990). Random numbers for simulation, *Communications ACM*, 33, 85-97.

L'Ecuyer, P. (1994). Uniform random number generators, *Annals of Operations Research*, 53, 77-120.

L'Ecuyer (1998). Random Number Generators and Empirical Tests, *Lecture Notes in Statistics 127*, Springer-Verlag, 1998, 124–138.

L'Ecuyer, P. (2001). Software for Uniform Random Number Generation: Distinguishing the good and the bad, *Proceedings of the 2001 Winter Simulation Conference, IEEE Press*, Dec, 95-105.

L'Ecuyer, P. and Granger-Piche, J. (2003). Combined Generators with Components from Different Families, *Mathematics and Computers in Simulation*, 62, 395-404.

L'Ecuyer, P., Simard, R., Chen, E.J. and Kelton, W.D. (2002). An object-oriented random number package with many long streams and substreams, *Operation Research*, 50, 1073-1075.

Lehmer, D. H. (1951). Mathematical methods in large-scale computing units, *Proceedings of the Second Symposium on Large Scale Digital Computing Machinery*, 141146, Harvard University Press, Cambridge.

Lewis, P.A., Goodman, A.S. and Miller, J.M. (1969). A pseudo-random number generator for the system/360, *IBM System's Journal*, 8, 136-143.

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Transactions on Modelling and Computer Simulation*, 8, 3-30.

McCullough, B.D. (1999). Assessing the Reliability of Statistical Software: Part II, *The American Statistician*, 53, 149-159.

Müller, P. (1991). A generic approach to posterior integration and Bayesian sampling, *Technical Report 91-09*, Statistics Department, Purdue University.

Müller, P. and Ríos Insua, D. (1998). Issues in Bayesian Analysis of Neural Network Models, *Neural Computation*, 10, 571-592.

Neelamkavil, F. (1987). *Computer Simulation and Modelling,* Wiley, New York.

Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia.

Park, S. and Miller, K. (1988). Random number generators: good ones are hard to find, *Communications ACM*, 31, 1192-1201.

OR/MS Today (2003).

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992). *Numerical recipes in C*, Cambridge University Press, Cambridge.

R, official web page. *http://www.r-project.org*

RAND Co. (1955). *A Million Random Digits with 100000 Normal Deviates*, Free Press.

Richardson, S. and Green, P. (1997). On Bayesian analysis of mixtures with an unknown number of components, *Journal of the Royal Statistical Society, B*, 59, 731-792.

Ríos Insua, D. and Müller, P. (1998). Feedforward neural networks for non-parametric regression, in *Practical Nonparametric and Semiparametric Bayesian Statistics* (D.Dey, P. Müller and D. Sinha eds.), Springer, New York.

Ríos Insua, D., Ríos Insua, S. and Martin, J. (1997). *Simulacion, Metodos y Aplicaciones.*, RA-MA, Madrid.

Ríos Insua, D. and Salewicz, K. (1995). The operation of Kariba Lake: a Multiobjective decision analysis, *Journal of Multicriteria Decision Analysis*, 4, 203-222.

Ripley, B. (1987). *Stochastic Simulation*, Wiley, New York.

Robbins, H. and Monro, S. (1951). A stochastic approximation method, *Annals of Mathematical Statistics*, 22, 400-407.

Ross, S. (1991). *A Course in Simulation*, MacMillan, New York.

Rubinstein, R. and Mohamed, B. (1998). *Modern Simulation and Modeling*, Wiley, New York.

Ruggeri, F. (2005). On the Reliability of Repairable Systems: Methods and Applications. To appear in *Proceedings of ECMI 2004*, Springer, New York.

Schmeiser, B. (1990). Simulation Methods, in *Stochastic Models* (Heyman and Sobel eds), North Holland, Amsterdam.

Schrage, L. (1979). A more portable FORTRAN random number generator, *ACM Transactions on Mathematical Software*, 5, 132-138.

Spiegelhalter, D., Thomas, A., Best, N. and Gilks, W. (1994). BUGS: Bayesian inference using Gibbs sampling, version 0.30, MRC Biostatistics Unit, Cambridge.

Stephens, M. (2000). Bayesian analysis of mixture models with an unknown number of components: an alternative to reversible jump methods, *Annals of Statistics*, 28, 40-74.

Tanner, M.A. (1996). *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions, 3nd ed.*, Springer, New York.

Von Neumann, J. (1951). Various techniques in connection with random digits, *NBS Appl. Math. Ser.*, 12, 36-38.

Whitt, W. (1989). Planning queueing simulations, *Management Science*, 35, 1341-1366.