STOCHASTIC SIMULATION,MONTE CARLO METHODS AND APPLICATIONS[1]

Ion Văduva,
University of Bucharest, Romania
e-mail: **vaduva@fmi.unibuc.ro: oriv@clicknet.ro**

Key words: Random numbers, Random variates, Random number generators, Liniar congruential generators, Inverse method, Composition method, Acceptance-rejection method, Accepting probability, Ratio-of-uniform method, Bootstrap method, Monte Carlo method, Monte Carlo procedure, Primary estimator, Secondary estimator, Crude Monte Carlo, Variance reduction techniques, Importance sampling, Antithetic variates, Control variates, Operational equations, Simulated annealing, Markov Chain Monte Carlo (MCMC) method, Metropolis-Hastings algorithm, Gibbs sampler, Queueing models, Birth and death processes, Constant/variable increment clock time in simulation models, Models with parallel stations, Machine interference problem, Inventory models, Economic lot model,Shortage model, Distribution of total demand,Stochastic inventory model, Feedback rule, Reserve stock in production series, Stochastic processes simulation, Uniform Poisson Processes, Uniform binomial processes, Scan statistics.

Contents

[1]This paper contains lectures given at Polytechnic University of Toluca Valey, Mexic, in November 2009

3.2.2 Simulation of Some Particular Distributions
3.2.3 Simulation of Some Multivariate Distributions

4 The Use of Simulation in Statistics

4.1 Estimation of Parameters via Simulation
4.2 Use of Simulation in Hypothesis Testing
4.3 The Bootstrap Technique

5 Use of Simulation in Numerical Calculations

5.1 Generalities on the Monte Carlo Method
5.2 Evaluating an Integral
5.2.1 Crude Monte Carlo
5.3 Variance Reduction Techniques
5.3.1 Importance Sampling
5.3.2 Antithetic Variates
5.3.3 Control Variates
5.4 Solving Operatorial Equations
5.4.1 Solving Systems of Linear Equations
5.4.2 Solving Integral Equations
5.5 Monte Carlo Optimization
5.6 Markov Chain Monte Carlo

6. Introduction to queueing models
6.0 Preliminaries
6.1 Birth and Death processes
6.2 Simulation of a queueing system with N parallel stations.
6.3 Simulation of the machine interference problem

7. Introduction to inventory models
7.0 Preliminaries
7.1 Simple one product models
7.2 Multiproduct models
7.3 Stochastic models
7.4 A simulation model

8.Simulation of some discrete uniform processes
8.1 Poisson uniform bivariate Process
8.2 Binomial uniform bivariate Process
8.3 An application to a healthcare problem

Glossary
Random number - a sampling value (or a realization) of a random variable

U uniformly distributed on $(0,1)$ (denoted $U \sim unif(0,1)$;

Random variate - a sampling value of a non-uniform distribution;

Random number generator - an algorithm which produces an uniform $(0,1)$ random number;

Linear congruential generator - a random number generator based on using a linear function modulo(m);

Inverse method - a method for simulating a random variate by inverting the cumulative distribution function (*cdf*) of the variate;

Composition method - a method for simulating a random variate based on representing the *cdf* of the variate;

Acceptance-rejection method - a method for simulating a random variate

as a function of some set of "simpler" random variates satisfying some specific condition;(the random set of variates which do not satisfy the condition are rejected and another set is tried);

Ratio-of-uniform method - a method for simulating a random variate as a ratio of uniformly distributed variates on some bivariate set;

Bootstrap method - a resampling method which produces a new sample from a given sample by extracting "with replacement" sampling values from the original sample;

Monte Carlo method - a method for solving a numerical problem by using random variates to estimate the solution to the problem;

Monte Carlo procedure-an algorithm producing a solution based on the Monte Carlo method;

Primary estimator-a random variable which estimates the solution to a numerical problem that is solved with a Monte Carlo procedure;

Secondary estimator-the arithmetic mean of the primary estimator giving the numerical solution based on the Monte Carlo method;

Simulated annealing-a method for solving optimization problems based Multiple integral- an integral from a function of several variables defined on on a domain in a multivariate spece.

on an algorithm that simulates the annealing physical process;

Markov Chain Monte Carlo (MCMC)-a simulation method based on simulating a statistical distribution as an ergodic distribution of a Markov chain;

Metropolis-Hastings algorithm-an algorithm used in the MCMC method;

Gibbs sampler- a particular form of the Metropolis-Hastings algorithm.

Queueing model- a model to analyze a queueing system.

Queueing model with parallel stations- a model for analyzing a queueing system with several parallel stations.

Machine interference problem- a model to analyze the maintenance of a system with N machines and M repair units.

Inventory system- a system which conserves a storage.
Economic lot model- a model defining the optimum order of a stock.
Shortage model- a model assuming shortage of stock.
Multistation inventory model - a model analyzing several types of products in a storage.
Stochastic inventory modrl- model with random demand, random lead time and/or random reorder cycle.
Uniform bivariate Poisson Process- a discrete stochastic process based on Poisson distribution of random points in a domain $D \subset R^2$.
Uniform bivariate binomial Process- a discrete process based on binomial distribution of random points in a domain $D \subset R^2$.
Bivariate scan statistics- a statistic counting the number of random points in a scanning window while scanning a given domain $D \subset R^2$.

Summary

The paper presents in short the main questions related to the use of simulation in studying statistical problems, solving some classes of numerical problems and analysing the behaviour of some particular systems (such as queueing and inventory systems). Methods for simulating various types of statistical distributions are presented first . Then, some applications of simulation in statistics, including bootstrap techniques, are also discussed. A special attention is paid to Monte Carlo techniques and the Markov Chain Monte Carlo method. Some simple mathematical models related to queueing and inventory systems are presented. Finaly, algorithms for simulating discrete uniform bivariate Poisson or Binomial processes are presented and a practical application involving scan statistics in healthcare is presented. References contain only some of the old representative or recent publications.

1 Introduction

The term *simulation* represents in our days science a wide class of problems, solved or analysed via computers. There are many ways to define and understand *simulation* but all of them assume the use of *random numbers* to perform a *computer experiment* for solving some mathematical or practical problem. In this papere, the word simulation is also associated with terms like *Monte Carlo* techniques and *resampling* techniques, the last one involving statistical problems.The word *simulation* can be also understood in this paper as a *mathematical experiment.*

Random numbers are sampling values on the uniform distribution over $(0, 1)$ which has the probability density function (*pdf*)

$$f(u) = \begin{cases} 1, if\ u \in (0,1) \\ 0, otherwise. \end{cases}$$

(Note that it makes no difference if we consider the interval $(0,1)$ as beeing open or closed at any of its limits).
The following proposition (due to Khintchine, see Ermakov-1971, Vaduva-1977) plays a great role in simulating non uniform random variates.

Theorem 1. 1 *If X is a random variable having the cummulative distribution function (cdf) $F(x), x \in R$, and U denotes the random variable uniformly distributed over $(0,1)$, then the random variable $F^{-1}(U)$ (with F^{-1}- the inverse of F), has the cdf F.*

In other words, this theorem gives a general method for simulating a sampling value x of the random variable X when we have a sampling value u of U, namely, $x = F^{-1}(u)$. That is why the next chapter will be dedicated to simulating random numbers. The same theorem suggests that there could be various methods which transform sequences of random numbers into non uniform variates. Thus, another chapter will discuss these methods.

Since one purpose of this paper is to discuss the use of simulation in solving statistical problems, one section is devoted to the *bootstrap* method and some applications. The Monte Carlo method for solving various numerical problems is introduced in another chapter. Some applications of the so called *Markov Chain Monte Carlo* are also presented. Then, some applications of simulation and stochastic modeling in queueing problems, inventory models and scan statistics are presented.

2 Random Number Generation

Random numbers, i.e. sampling values on the random variable U uniformly distributed over $(0,1)$ (denoted $U \sim (0,1)$), are very important for the problems to be treated in this paper.

The aim of this chapter is to present in short some methods for generating with the computer sampling values on the random variable U, which are independent and uniformly distributed over $[0,1)$. As Knuth and other authors have shown, the computer calculations necessary to produce *good* random numbers, require first to generate an uniform integer over some interval $[0,m)$, and then to divide this by m in order to obtain the required random number. The calculations needed to produce an uniform integer in $[0,m)$ must be simple. In other words, the generation algorithm must have a low complexity, both regarding computing time and memory complexities. Details on random number generation are found in many books (see for instance Devroye-1986, Ermakov-1971, Gentle-1998, Ripley-1986 and Ross-1997).

2.1 Linear Congruential Generators

A *linear congruential generator* is of the form

$$x_n = (\sum_{i=1}^{k} a_i x_{n-i} + c)(mod\, m), x_n \in \mathcal{N}, \tag{2.1}$$

where m is a large positive integer, $k \leq n$, and $a_i, x_i, i = 1, ..., k, c$ are given constants, all chosen such that the produced numbers $x_n, n > k$ are integers uniformly distributed over the interval $[0, m-1)$. Then the uniform $U[0,1)$ random numbers are obtained as

$$u_n = x_n/m. \tag{2.1'}$$

The usual linear (*mixed*) congruential generator is the one with $k = 1$, i.e. $x_{n+1} = (a * x_n + c)(mod\, m)$. If a, c and m are properly chosen, then, in this case, $u_n's$ "look like" they are randomly and uniformly distributed between 0 and 1. Even if this linear congruential generator has a low complexity, the most used is the *multiplicative congruential generator*

$$x_{n+1} = (ax_n)(mod\, m). \tag{2.2}$$

It is shown (Knuth-1981) that if $x_0 \neq 0$ is prime to m, and a is a *primitive root mod m*, close to $\sqrt{m}$, then the numbers u_n produced by this generator have a large *period* λ, (defined as the minimum λ such as $x_n = x_{n+\lambda}$), they are aproximately uniform $(0,1)$ distributed and have a small *serial correlation coefficient* $\rho = corr(u_n, u_{n+1}) \forall n$ (i.e are *almost independent*). Of course, the *modulus* m must be very large (usualy close to the computer word, i.e. close to 2^{31} for usual computers).

In simulation we use sequences of random numbers $u_1, u_2, ..., u_n$ produced by a random number generator. These numbers must pass any test which assumes that they are uniformly distributed and stochastically independent. It is obvious that a random number generator cannot produce "pure" random numbers to pass the mentioned tests. (see knuth-1981). Therefore we call them *pseudo-random numbers.* A "good" random number generator must produce sequences close to pure random numbers. A linear congruential generator cannot produce good random numbers. It can be used when there is no need to perform *very accurate calculations* or to obtain *exact* solutions to the problems.

One trouble with using pseudorandom numbers produced by a linear congruential generator is that pairs (u_i, u_{i+1}) or triplets (u_i, u_{i+1}, u_{i+2}) are lying on lines or planes (i.e have a *latice structure*). This means that these generators must be used with care in numerical calculations. In order to obtain

"better" random numbers from an uniform pseudo-random number generator, the numbers produced by the generator must be transformed. If we consider the *binary representation* of the numbers u_i, then one way to obtain better numbers is to use the *bit stripping*, i.e. to obtain the new numbers by selecting some bits from the sequences of bits representing previous given numbers (e.g. odd bits or even bits, etc).

2.2 Other Sources of Uniform Random Numbers

Note that if in (2.2) we take a^k instead of a and start with x_s, then the sequence of pseudo-random numbers obtained is $x_s, x_{s+k}, x_{s+2k}, ...$ and therefore, for various values of s, the corresponding stream can be used by one processor in a parallel architecture.

Shuffling random numbers. A way of improving the quality of an uniform pseudo-random number generator is to define the *new* number y by mixing (or *shuffling*) two generators $G1, G2$. One mixing algorithm (due to MacLaren and Marsaglia) is:

Take an array (i.e. a table) $T[1..k], k = fixed$, *and initialize (fill in) it using* G_1;
generate with G_2 *a random index* $j \in \{1, 2, ..., k\}$;
take $y := T[j]$; *generate* x *with* G_1, *and put* $T[j] = x$.

(The notation $a := b$ means that b is assigned to a). The *better* generated number is y. This mixed generator can have a larger period and can break up the lattice structure of the generated sequence $\{y_i\}$. If instead of two generators we use only one, $G = G_1 = G_2$, then the above algorithm (called Bays-Durham shuffling of random numbers) can be easily changed by generating only one x in the initial step and determining j by the "*bit stripping*"procedure mentioned before.

Lagged Fibonacci sequences. Apart from linear congruential generators, another way of generating random numbers is to use the *lagged Fibonacci generator,* defined as

$$x_i = (x_{i-j} + x_{i-k})(mod\, m) \tag{2.3}$$

which, when m is prime and $k > j$, gives a period close to $m^k - 1$.

Inversive congruential generators (due to Eichenauer Hermann). This method produces uniform integers over $[0, m-1]$ by the relation

$$x_i = (ax_{i-1}^{-1} + c)(mod\, m) \tag{2.4}$$

where x^{-1} denotes the multiplicative *inverse* modulo m if it exists, or else is 0. Even if these inversive generators imply computational difficulties, they promise to give high quality random sequences.

Matrix congruential generators. Such a generator is of the form

$$x_i = (Ax_{i-1} + C)(mod\, m)$$

where x_i are vectors of dimension d and A and C are $d \times d$ matrices. This kind of generators are important when parallel computers are used to produce correlated random vectors.

Feedback shift register generators. Such a generator takes into consideration the binary representation of integers in registers of the computer. If $a_i, i = 1, ..., p$, denote the binary digits of the random number, and c_i are given (not all zero) binary digits, then the digits a_i of the *new* generated numbers are produced by

$$a_i = (c_p a_{i-p} + c_{p-1} a_{i-p+1} + ... + c_1 a_{i-1})(mod\, 2). \tag{2.5}$$

This generator was introduced by Tausworthe. In practice it has the form

$$a_i = (a_{i-p} + a_{i-p+q})(mod\, 2) \tag{2.5'}$$

or, if we denote $\bigoplus$, the *binary exclusive-or operation,* as addition of $0's$ and $1's$ modulo 2, equation $(2.5')$ becomes

$$a_i = a_{i-p} \bigoplus a_{i-p+q}. \tag{2.5"}$$

Note that this recurrence of bits $a_i's$ is the same as the recurrence of random numbers, (interpreted as $l-$tuples of bits), namely,

$$x_i = x_{i-p} \bigoplus x_{i-p+q}. \tag{2.6}$$

If the random number has l binary digits ($l \leq p$), and l is relatively prime to $2^p - 1$, then the period of the $l-$tuples (i.e. of the sequence of generated numbers) is $2^p - 1$. A variation of the Tausworthe generator, called *generalized feedback shift register (GFSR)*, is obtained if we use a bit-generator in the form (5') to obtain an $l-$bit binary number and next bit-positions are obtained from the same bit-positions but with delay (by *shifting* usualy to the left). A particular GFSR is $x_i = x_{i-3p} \bigoplus x_{i-3q}, p = 521, q = 32$ which gives a period $2^{521} - 1$. Another generator of this kind is the so called *twisted GSFR generator,* which recurrently defines the random integers x_i as

$$x_i = x_{i-p} \bigoplus Ax_{i-p+q} \tag{6'}$$

where A is a properly chosen $p \times p$ matrix .

A practical remark. Apart from *shuffling* random numbers as mentioned above, some other simple combinations could be used to produce "good" random numbers. Thus, if we use the following three generators

$$x_i = 171x_{i-1}(mod\,30269), y_i = 172y_{i-1}(mod\,30307), z_i = 170z_{i-1}(mod\,30323)$$

with positive initializations (*seeds*) $(x_0, y_0, z_0,)$ and take uniform $(0, 1)$ numbers such as

$$u_i = (\frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323})\ (mod\,1)$$

it can be shown that the sequence of u_i's has a period of order 10^{12}.

3 Non Uniform Random Variate Generation

In this chapter we assume that an uniform $(0, 1)$ random number generator called **rnd** is given. The aim of this chapter is to present methods and algorithms which transform sequences of random numbers $u_1.u_2, ..., u_n, n \geq 1$ into a sampling value of a given random variable X which has a cdf $F(x)$. (For further information see Devroye-1986,Gentle-1998 and Ross-1997).

3.1 General Methods

3.1.1 The Inverse Method

Theorem 1.1 leads to the following algorithm (the *inverse method*):

generate u *with* **rnd**; *take* $x := F^{-1}(u)$.

The following list gives some examples of the inverse method:

Distribution	*cdf*	*Inverse*
$Exp(\lambda)$	$F(x) = 1 - e^{-\lambda x}, x > 0, \lambda > 0$	$x := -ln(u)$
$Weib(0, 1, \nu)$	$F(x) = 1 - e^{-x^{\nu}}, \nu > 0$	$x := (-ln(u))^{1/\nu}$
$Cauch$	$F(x) = \frac{1}{\pi}(arctan\, x + \frac{\pi}{2}), x \in R$	$x = tan\, \pi(u - \frac{1}{2})$
$Pears\, XI$	$F(x) = 1 - \frac{1}{(1+\alpha x)^{\nu}}, x > 0, \nu > 0$	$x = \frac{1}{u^{1/\nu}}$

(The abbreviations are: Exp for exponential; Weib for Weibull; Cauch for Cauchy; Pears XI for Pearson type XI).

In the multivariate case, there is a generalization of Theorem 1.1 (see Ermakov-1981), which gives a similar algorithm for simulating a sampling value $\mathbf{x} = (x_1, x_2, ..., x_k)'$ of the k-dimensional random vector $\mathbf{X}$ which has the cdf $F(\mathbf{x})$. Let us denote

$F_1(x_1) = P(X_1 < x_1), F_j(x_j|x_{j-1}, ..., x_1) = P(X_j < x_j|X_{j-1} = x_{j-1}, ..., X_1 = x_1), 1 < j \leq k.$

The algorithm is (the *multivariate inverse method*):

generate u *with* **rnd;** *take* $x_1 = F_1^{-1}(u|x_{j-1}, ..., x_1)$;
for $i := 2$ **to** k **do**
begin
Generate u *with* **rnd**; *take* $x_i = F_i^{-1}(u)$;
end.

An inverse algorithm for simulating a finite discrete random variate having probability distribution

$$X : \begin{pmatrix} a_1, & a_2, & ..., & a_n \\ p_1, & p_2, & ..., & p_n \end{pmatrix}$$

is:

calculate $F_i = \sum_{\alpha=1}^{i} p_\alpha$, $1 \leq i \leq n$; *take* $i := 0$;
generate u with **rnd;**
repeat
$i := i + 1$;
until $u < F_i$;
take $x := a_i$.

The loop in the algorithm searches for the value of index i; this can be better done by using the *binary search* technique.

3.1.2 The Composition (Mixture) Method

If the cdf of the random variable X is of the form

$$F(x) = \sum_{i=1}^{k} p_i F_i(x),\ p_i > 0,\ \sum_{i=1}^{k} p_i = 1 \tag{3.1}$$

then one says that F is a *mixture* (or *composition*) of $F_i(x), i = 1, ..., k$. Note that $p_i = P(X = X_i)$, where $X_i \sim$ *cdf* $F_i(x)$. If x_i denotes the random variate corresponding to X_i, $1 \leq i \leq k$, which is assumed can be simulated, then the algorithm for simulating the random variate x is:

generate a random index i, *such as* $P(i) = p_i$;
generate a random variate x_i *having the cdf* $F_i(x)$;
take $x := x_i$.

Example 3.1. Assume that X has a *mixed exponential distribution* i.e. its pdf is:

$$f(x) = \sum_{i=1}^{k} p_i \lambda_i e^{-\lambda_i x}.$$

As x_i (which are exponential) can be generated by the inverse method, x can be generated by the previous algorithm.

A composition algorithm can be built up also in the case of a *continuous mixture* i.e.

$$F(x) = \int_{-\infty}^{\infty} G(x,y)dH(y) \tag{3.1'}$$

where for each $y, G(x,y)$ (as a function in x) is the cdf of a random variable Z_y, and $H(y)$ is the cdf of a random variable Y. (It is assumed that Y and Z_y are random variables that can be simulated).

Example 3.2 Assume that Z_λ is a rv exponentially distributed with parameter λ and that the parameter λ is random being $Gamma(0,a,b)$ distributed, i.e. has the pdf

$$h(\lambda) = \frac{a^b}{\Gamma(b)} \lambda^{b-1} e^{-a\lambda},\ a,b > 0. \tag{3.2}$$

In this case, the pdf of the continuous mixture (i.e. the pdf of the rv $X = Z_\lambda$) is

$$f(x) = \frac{ba^b}{(a+x)^{b+1}},\ x > 0 \tag{3.3}$$

and the composition algorithm for simulating x is

generate $\lambda \sim Gamma(0,a,b)$;
generate $z_\lambda \sim Exp(\lambda)$;
take $x = z_\lambda$.

One says that X has a *Lomax distribution* (Pearson XI) and it is used in reliability as a *life time* distribution.

3.1.3 The Acceptance-Rejection Method

This method (simply called the *rejection* method sometimes), consists in using some random variates which can be simulated, say $s_1, s_2, ..., s_n$ and, when some predicate $\mathcal{P}(s_1, s_2, ..., s_n)$ is *true*, the random variate x is $x := \Psi(s_1, s_2, ..., s_n)$. Of course, for a given probability distribution (i.e. for a given X), the predicate $\mathcal{P}$ and the function Ψ must be defined in a suitable

manner. The index n itself could be a rv which can be generated. The attribute "rejection" for the method means that, for some set of generated variates $s_1, s_2, ..., s_n$, the predicate could be *false*; in this case, the set is rejected and another set is tried. That is why, for the rejection method, the probability $p_a = P(\mathcal{P}(s_1, ..., s_n) = true)$ (called *the acceptance probability*) must be large in order to give a good rejection algorithm.

In the following we will present some theorems leading to rejection algorithms.

Theorem 3. 1 *(The enveloping rejection method). Let X be a rv with the pdf $f(x)$ and assume that Y is another rv with pdf $h(x)$ and both $f(x), h(x)$ are non negative on the same set in R. If there is a constant $\alpha > 1$ such that for an uniform $(0, 1)$ random variable U, independent from Y, we have $0 \leq U \leq f(Y)/(\alpha h(Y))$, then the pdf of this Y is f.*

Example 3.3 Let X be the normal $N(0, 1)$ rv and assume that $X_1 > 0$ is the positive normal deviate which has the pdf

$$f(x) = \sqrt{\frac{2}{\pi}} e^{-x^2/2}, x > 0. \tag{3.4}$$

Take as envelope $Y \sim Exp(\lambda = 1)$. Then, one can easily find that $\alpha = \sqrt{2e/\pi}$ and $p_a = 1/\alpha = \sqrt{\pi/(2e)}$. The rejection algorithm for simulating x_1 is

repeat
generate $u \sim unif(0, 1)$ and generate y independent from u;
until $u < e^{-(y^2)/2+y-0.5}$;
take $x_1 := y$.

For simulating the rv X we must add the following steps:

generate u with **rnd**;
if $u < 0.5$ **then** $s := 1$ **else** $s := -1$; { s is a *random sign;*}
take $x := s.x_1$.

In order to generate a normal $N(\mu, \sigma)$ random variate w, the following step should be added to the preceding algorithm:

take $w := \mu + \sigma x$.

Theorem 3. 2 *(See Vaduva-1977). Assume that the pdf of X is of the form*

$$f(x) = cQ(\varphi(x))r(x) \tag{3.5}$$

where $Q(z)$ is the cdf of a rv Z, $M > Z > 0$, $r(y)$ is the pdf of a rv Y, (Y stochastically independent from Z), and φ is a function such that $0 \leq \varphi(z) \leq M$. Then, the conditional pdf of Y, given that $Z \leq \varphi(Y)$, is f.

The acceptance probability is $p_a = P(Z \leq \varphi(Y))$. Note that if X and Y are random vectors, Theorem 3.1, as well as Theorem 3.2, are valid. Note also that there is an *alternative* of Theorem 3.2, when

$$f(x) = c(1 - Q(\phi(x)))r(x). \tag{3.5'}$$

In this case, the statement of the theorem remains valid if the predicate is changed as $Z \geq \varphi(Y)$.

Example 3.4. (See Vaduva-1977). Let X^* be a $Gamma(0, 1, \nu)$ random variable, $0 \leq \nu \leq 1$ and take $X = X^* > 1$. Then the pdf of X is of the form (12) with

$$\varphi(x) = x,\ Q(x) = 1 - 1/(x^{\nu-1}), x \geq 1, r(x) = e^{-x+1}, x \geq 1, c = \frac{1}{e(\Gamma(\nu) - \Gamma(1;\nu)}.$$

In the previous formula $\Gamma(\nu)$ and $\Gamma(1;\nu)$ (*the incomplete gamma function*) are

$$\Gamma(\nu) = \int_0^\infty x^{\nu-1}e^{-x}dx\,, \Gamma(1;\nu) = \int_0^1 x^{\nu-1}e^{-x}dx. \tag{3.6}$$

From the mentioned *alternative theorem* the following algorithm for simulating $X \sim Gamma(0, 1, \nu)$, $X \geq 1$ is derived:

repeat
generate a random variate $z \sim Q$
{*i.e. generate* $u \sim unif(0, 1)$ *and take* $z := u^b, b = \frac{-1}{1-\nu}$};
generate $y_0 \sim Exp(\lambda = 1),\ y_0 \in [1, \infty)$;
until $z > y_0$;
take $x := y_0$.

Theorem 3. 3 *Assume that* $Z_1, Z_2, ...$ *are iid (short for "independent and identically distributed") random variables with cdf* $G(x)$, *and that* Z_0 *is independent from* Z_i *having the cdf* $G_0(x)$. *Then the following assertions are valid:*
1^0. $P(x > Z_1 \geq ... \geq Z_{k-1} < Z_k) = \frac{(G(x))^{k-1}}{(k-1)!} - \frac{(G(x)^k}{k!}$ *for a fixed* k *and* x;.
2^0. *If* K *is a rv such as* $x \geq Z_1 \geq Z_2 \geq ... \geq Z_{K-1} < Z_k$, *then* $p_a = P(K =$ *odd integer*$) = e^{-G(x)}$;
3^0. *If* Z_0 *is the above mentioned rv and the descending sequence from* 1^0, *starting with* Z_0, *breaks at* K, *which is an odd integer, then*

$$F(x) = P(Z_0|K = odd) = \frac{1}{p_a}\int_{-\infty}^{x} e^{-G(t)}dG_0(t), \quad p_a = \int_{-\infty}^{+\infty} e^{G(t)}dG_0(t). \tag{3.7}$$

For the rejection algorithm deriving from this theorem, the p_a is the acceptance probability.

Example 3.5. Theorem 3.3 (due to Forsythe) leads to the following John von Neumann's algorithm for simulating a random variate $x \sim Exp(\lambda = 1)$.

$N := 0$;
repeat
generate $u_0, u_1, iid \sim unif(0,1)$; *take* $u^* := u_0$; $k := 1$;
while $u_0 > u_1$ **do**
begin $u_0 := u_1; k := k + 1$; *generate* u_1; **end;**
if $k \, mod \, m = 0$ **then** $N := N + 1$;
until $k \, mod \, m = 1$;
take $x := u^* + N$.

According to Theorem 3.3, if G_0 and G are iid uniform $(0,1)$ then u^* in the algorithm is $Exp(\lambda = 1)$ distributed, truncated on $[0,1]$. The theorem of John von Neumann says that $x = N + u^* \sim Exp(\lambda = 1)$, which is the output of the previous algorithm.

Example 3.6. If $G_0(x) := u^\nu, u \in [0,1]$ and $G(x)$ is the uniform $(0,1)$ cdf (i.e if in Theorem 3.3, $Z_0 = U^{1/\nu}, Z_i = U_i$- uniform $(0,1)$), then Z_0 in the accepted descending sequence has a $Gamma(0,1,\nu)$ distribution truncated on $[0,1]$. Combining this result with example 5 one can derive a *composition-rejection* algorithm for simulating the $Gamma(0,1,\nu)$ distribution when $0 < \nu < 1$.

3.2 Other Methods

3.2.1 Ratio-of-uniform Method

This method, due to Kinderman and Monahan (see Devroye-1986), was used to simulate various particular distributions. The following theorem (see Vaduva-1993) gives a general form of the method.

Theorem 3. 4 *Assume that the pdf of an* $m-$*dimensional random vector* $\mathbf{X}$ *has the pdf in the form*

$$f(x) = \frac{1}{H} h(\mathbf{x}), \mathbf{x} \in R^m, \, H = \int\limits_{R^m} h(\mathbf{x}) d\mathbf{x} \tag{3.8}$$

and consider the mapping $\varphi : R^{m+1} \to R^m$ *defined as*

$$\varphi(v_0, v_1, ..., v_m) = \left(\frac{v_1}{v_0^c}, ..., \frac{V_m}{V_0^c}, c > 0 \right). \tag{3.9}$$

Consider the set $C \subset R^{m+1}$,

$$C = \{(v_0, v_1, ..., v'_m | \gamma(v_0, v_1, ..., v_m) \leq 0\}, \tag{3.10}$$

where $V_0 > 0$ *and*

$$\gamma(v_0, ..., v_m) = log\, v_0 - d.log(h\varphi(v_0, v_1, ..., v_n)),\ d = \tfrac{1}{mc+1}. \tag{3.10'}$$

If the set C *is bounded on* R^{m+1} *and* $\mathbf{V}$ *in an uniform random vector over* C*, then the random vector* $\mathbf{X} = \varphi(\mathbf{V})$ *has the pdf* f*.*

This theorem leads to the following general algorithm:

generate $\mathbf{v} \sim unif(C)$;
take $\mathbf{x} := \varphi(\mathbf{v})$.

In order to simulate the sampling value $\mathbf{v} \sim unif(C)$, the following rejection procedure is used:

find a minimum interval $I = [a_0, b_0] \times [a_1, b_1] \times ... \times [a_m, b_m]$, $C \subset I$;
repeat
generate a random vector $\mathbf{w} \sim unif(I)$;
{*This will be done in section 3.2.3*};
until $\mathbf{w} \in C$;
take $\mathbf{v} = \mathbf{w}$.

The *minimum* interval I is obtained from the conditions

$$\min_{(v_0,v_1,...,v_m)\in C} v_i,\ \max_{(v_0,v_1,...,v_m)\in C} v_i, i = 0, 1, ..., m. \tag{3.11}$$

The acceptance probability of the previous algorithm is

$$p_a = \frac{mesC}{\prod_{i=0}^{m}(b_i - a_i)},\quad mesC = \int\limits_{R^m} d\mathbf{x}. \tag{3.11'}$$

Example 3.7. Applying Theorem 3.4 for generating the $Gamma(0, 1, \nu)$ distribution, we obtain the limits of I as

$$a_0 = 0,\ b_0 = (\nu - 1)^{\frac{\nu-1}{c+1}} e^{-\frac{\nu-1}{c+1}};\ a_1 = 0, b_1 = (\tfrac{c\nu+1}{c})^{\frac{c\nu+1}{c+1}} e^{-\frac{c\nu+1}{c+\nu}}. \tag{3.12}$$

The interval I is bounded and the algorithm is obvious.

Example 3.8. For simulating the normal $N(0, 1)$ distribution, the limits of the interval I, when $c = 1/2$, (the value which maximizes the probability p_a), and p_a are

$$a_0 = 0,\ b_0 = 1,\ b_1 = \sqrt{\tfrac{3}{e}},\ a_1 = -b_1,\ \ p_a = \sqrt{e/(12\pi)}. \tag{3.13}$$

3.2.2 Simulation of Some Particular Distributions

Apart from the mentioned examples, various properties, sometimes combined with the general methods, can give us algorithms for simulating particular random variates. We will list up some of these methods

• **Normal distribution** could be simulated (*approximately*) by using the *central limit theorem* as follows:

$z := 0$;
for $i := 1$ **to** 12 **do**
begin
generate $u \sim unif(0,1)$; $z := z + u$;
end;

z is a normal $N(0,1)$ random variate.

• The **Gamma**$(0,1,\nu)$ **distribution,** when $\nu > 1$, can be easily simulated by the following simple procedure:

generate an Erlang variate as $E = \sum_{i=1}^{k} z_i, k = [\nu]$ (*integerpart*) *where* z_i *are iid* $Exp(1)$ distributed;
generate $G \sim Gamma(0,1,p), 0 < p < 1$;
take $x = E + G$.

(Here the fact that the sum of independent gamma random variables is still gamma distributed was used).

• The **Beta distribution** has the pdf in the form

$$f(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1},\ x \in [0,1],\ a > 0, b > 0;\ f(x) = 0, x \notin [0,1]. \quad (3.14)$$

One simple method for simulating a Beta distributed random variate x is based on its property which says that if w_1, w_2 are independent and $Gamma(0,1,a), Gamma(0,1,b)$ distributed respectively, then $x = w_1/(w_1 + w_2)$, is Beta distributed.

•**Distributions based on Bernoulli trials** are *binomial(n,p), geometric(p) and negative binomial(k,p) (or Pascal).* A *Bernoulli trial* is an experiment which involves a fixed event A, with a constant probability $p = P(A)$; in an experiment the event may occur (i.e. *succes* takes place) or may not occur (i.e. *failure* takes place). If we associate to the Bernoulli trial the random variable Z such that $P(Z = 1) = p = P(succes)$ and $P(Z = 0) = 1 - p = q = P(failure)$, then this can be simulated as:

generate $u \sim unif(0,1)$; **if** $u < p$ **then** $z := 1$ **else** $z := 0$.

The **Binomial**$(n,p), n \in N^+$ rv X is the number of successes in n independent Bernoulli trials, this being one method for generating it.
It is known that the binomial distribution is connected with an urn containing balls of two colors, white and black, such that $p = P(to\, extract\, one\, white)$, the ball being returned to the urn. The rv X is the number of white balls from n independent extractions. If the extracted balls are not returned and extractions start with a given composition of the urn (say, $A = p.N$-white balls, N-the total number of balls in the urn), then the number Y of white balls extracted out of n, has a **hypergeometric distribution** and it is simulated as:

given N, p, n *initialize* $i := 0; y := 0;$
repeat
generate u *with* **rnd**, *take* $i := i + 1;$
if $u < p$ **then** $s := 1$ **else** $s := 0;$
update $p := \frac{Np-s}{N-1}$; $N := n - 1$; $y := y + s;$
until $i = n$.

The **geometric**(p) rv Y is the number of failures until one success occurs in several Bernoulli trials.

The **Pascal**$(k,p), k \in N^+$ rv T is the number of failures until k successes occur.When $k = 1$, T is a geometric rv. An algorithm (based on counting Bernoulli trials) for simulating T is:

$T := 0; s := 0;$
repeat
generate $u \sim unif(0,1)$; **if** $u > p$ **then** $T := T + 1$ **else** $s := s + 1;$
until $s = k$.

• **Poisson distribution.** The rv $X \in N^+$ has a $Poisson(\lambda)$ distribution if its frequency function is

$$f(k) = P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}. \tag{3.15}$$

Taking into account the fact that, if some events occur at random time intervals which are $Exp(\lambda)$ distributed, then the *number* of events occuring on the unit time is $Poisson(\lambda)$, it results that the $Poisson(\lambda)$ random variate x is simulated as

$x := 0;\ P := 1; L := e^{-\lambda};$
repeat
generate $u \sim unif(0,1)$; *take* $P := P * u;$
if $P \geq L$ **then** $x := x + 1;$

until $P < L$;

3.2.3 Simulation of Some Multivariate Distributions

The *multivariate inverse* method and Theorem 5 are general methods for simulating random vectors. All such methods consist in simulating the *components* of the random vectors. Here we will present some algorithms for simulating particular multivariate distributions.

• **Uniform random vectors.** If $\mathbf{W} = (W_1, W_2, ..., W_m)'$ is uniformly distributed on the interval $I = [a_1, b_1] \times ... \times [a_m, b_m]$, then the components $W_i, i = 1, 2, ..., m$ are independent and uniformly distributed over $[a_i, b_i]$. Therefore the algorithm for simulating $\mathbf{W}$ is

for $i := 1$ **to** m **do**
begin *generate* $u \sim unif(0, 1)$; *take* $w_i := a_i + (b_i - a_i) * u$ **end.**

An algorithm for simulating an m-dimensional random vector $\mathbf{V}$ uniformly distributed over some domain $C \subset R^m$ was presented in connection with Theorem 3.4.

• **The multivariate normal distribution** $N(\mu, \Sigma)$ has the pdf

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{m/2} det(\Sigma)^{1/2}} e^{-Q(\mathbf{x})}, Q(\mathbf{x}) = (\mathbf{x} - \mu)' \Sigma^{-1} (\mathbf{x} - \mu) \tag{3.16}$$

where μ is the mean vector and $\Sigma = (\sigma_{ij})$ is the covariance matrix. The random vector $\mathbf{X}$, distributed $N(\mu, \Sigma)$ can be simulated by using Theorem 5 (as in examples 3.8,3.9). The *optimum* interval $I \supset C$ obtained for $c = 1/2$ is

$$b_i = \sqrt{\frac{(m+2)\sigma_{ii}}{e}}, \; a_i = -b_i, \; 1 \leq i \leq m. \tag{3.17}$$

Another method uses the following property of the $N(\mu, \Sigma)$ distribution: if $\mathbf{Y}$ is $N(\mu, \Sigma)$ and D is a $m \times m$ matrix, then the random vector $\mathbf{X} = D\mathbf{Y}$ is $N(\mu, D\Sigma D')$. Therefore, to generate $\mathbf{X} \sim N(\mu, \Sigma)$, one first determines the lower triangular matrix D such that $DD' = \Sigma$. Then $\mathbf{X}$ is generated as

generate $\mathbf{z} \sim N(0, I)$ (I *is the unit matrix*);
take $\mathbf{x} = \mu + DD'$.

• The **Multivariate Lomax distribution** $ML_m(a, \theta_1, ..., \theta_m)$ has the pdf in the form

$$f(x_1, ..., x_m) = \frac{(\prod_{i=1}^{m} \theta_i) a(a+1)...(a+m-1)}{(1 + \sum_{i=1}^{m} \theta_i x^i)^{a+m}}, \theta_i, a > 0. \tag{3.18}$$

It is known that, if $\mathbf{X} = (X_1, ..., X_m)'$, and $X_i, 1 \leq i \leq m$ are independent and $Exp(\eta\lambda_i)$ distributed with η a random variable, $\eta \sim Gamma(0, b, a)$, then $\mathbf{X}$ is Lomax $ML_M(a, \theta_1, ..., \theta_m)$ with $\theta_i = \lambda_i/b, 1 \leq i \leq m$. This gives an obvious *composition* procedure for simulating $\mathbf{X}$. The multivariate inverse method induces the following algorithm for simulating $\mathbf{X}$:

generate $u \sim unif(0,1)$ *and take* $x_1 := \frac{1}{\theta_1}(u^{-1/a} - 1)$;
for $k = 2$ **to** m **do**
begin
generate $u \sim unif(0,1)$;
take $x_k = \frac{1+\sum_{i=1}^{k-1} \theta_j x_j}{\theta_k}(u^{-1(a+k-1)} - 1)$
end.

• The **Multinomial distribution** $MD(n, p_1, p_2, ..., p_m)$ is a m-dimensional extension of the binomial distribution. For $\mathbf{X} \sim MD(n, p_1, ..., p_m)$, the frequency function is

$$P(X_1 = n_1, ..., X_m = n_m) = \frac{n!}{n_1!...n_m!} p_1^{n_1} ... p_m^{n_m}, n = n_1 + ... + n_m. \tag{3.19}$$

If $A_1, A_2, ..., A_m$ are exclusive events, $p_i = P(A_i), 1 \leq i \leq m$, (i.e. in one experiment only one of the A_i is produced), then the component X_i of $\mathbf{X}$ is the number of realizations of A_i in n independent experiments. This gives the hint for simulating $\mathbf{X}$.

4 The Use of Simulation in Statistics

4.1 Estimation of Parameters via Simulation

A simulation study is frequently performed to determine the value of some parameter θ connected to a particular stochastic model such as $\theta = E(X)$, X being a random quantity which can be simulated. Of course, if we generate a sample $x_1, ..., x_n$ on X, the estimate of θ is the arithmetic mean $\overline{x} = (\sum_{i=1}^n x_i)/n$ which is an unbiased and consistent estimator of θ. We can operate in a similar manner if θ is the expectation of a function of X, $\theta = E[f(X)]$. One question when estimating a parameter is first to determine the sample size n. This can be done by using the Tchebyshev's inequality. More precisely, if the variance σ^2 of X (or $f(X)$) is known, and we wish to estimate θ with a given error ϵ, such as

$$P(|\overline{x} - \theta| < \epsilon) \geq \delta, \delta \in (0,1) \tag{4.1}$$

(with δ large enough), then one can see that

$$n \geq \frac{\sigma^2 t_\delta^2}{\epsilon_2}, t_\delta = \sqrt{\frac{1}{1-\delta}}. \tag{4.2}$$

If σ^2 is unknown, then it is estimated by a preliminary computer run, namely: k variates $x_1, ..., x_k$ are generated ($k \approx 30$) after $\overline{x}$ is calculated, $\sigma^2 \approx s^2 = (\sum_{i=1}^{k}(x_i - \overline{x})^2)/(k-1)$ (s^2 is un unbiased and consistent estimate of σ^2).

A good estimate of θ can be obtained if one first determines k such that $Var(\overline{x})$ is less then some constant d, as follows:

Take $k = 30$ *and generate the sample* $x_1, ..., x_k$; *calculate* s^2;
repeat
generate a new x *and include it in the sample, then again calculate* s^2;
take $k := k + 1$;
until $s/\sqrt{k} < d$.

Now, with the resulting k we can determine a good estimate $\overline{x}$ of θ.

If $\theta = p$ (i.e. X is a Bernoulli random variable taking values 0 or 1), then the previous algorithm gives the estimate of p with a required error $\epsilon = d$. In statistics parameters are also estimated by a *tolerance interval,* which is an interval in the form $[\hat{t}_1, \hat{t}_2]$ with $\hat{t}_i = \hat{t}_i(x_1, ..., x_n), i = 1, 2$ and satisfying the property that $P(\theta \in [\hat{t}_1, \hat{t}_2]) = \delta$ where $\delta \in (0, 1), \delta$ *large* being the *tolerance level.* If we are interested to estimate θ by a *short* tolerance interval, then we must also use one suitably large sample size. But for a large sample size n, a tolerance interval is obtained by taking into account the fact that the arithmetic mean $\overline{x}$ is asymptotically normal $N(\theta, \sigma/\sqrt{n})$. This interval is

$$[\hat{t}_1, \hat{t}_2], \quad \hat{t}_1 = \overline{x} - z_{\delta/2}\frac{s}{\sqrt{n}}, \quad \hat{t}_2 = \overline{x} + z_{\delta/2}\frac{s}{\sqrt{n}}, \tag{4.3}$$

where $z_{\delta/2}$ is defined as

$$\frac{1}{\sqrt{2\pi}} \int_0^{z_{\delta/2}} e^{-\frac{z^2}{2}} dz \tag{4.4}$$

and $\overline{x}, s$ are estimates of θ and σ respectively.

4.2 Use of Simulation in Hypotheses Testing

In many situations, in order to test some statistical hypothesis, we need the *critical values* of the test statistic or, for some tests, we need to estimate *test power.*

Assume that we are interested in testing some hypothesis H on some random variable X and that we use some sample $x_1, x_2, ..., x_n$ with a given n. If the test uses the statistic $t = t(x_1, ..., x_n)$, for a given significance level α, then we need the *critical value* t_α such that $P(t > t_\alpha) = \alpha$ (for an one side test!). In order to estimate t_α we simulate N replicates of t and build-up a histogram as follows:

input N=*the sample size used to estimate* t_α;
input n *and* k=*the number of intervals of the histogram of* t;
for $i := 1$ **to** k **do** $\nu_i := 0$;
{ν_i *are the frequencies of the histogram*};
input a sample size $n_1, n_1 << N$;
for $i := 1$ **to** n_1 **do**
begin
generate under the hypothesis H *the sample* $x_1, ..., x_n$;
calculate the test statistic $t_i := t(x_1, ..., x_n)$;
end;
order the vector $(t_1, ..., t_{n1})$; *take* $a_1 := \min_{1 \le i \le n_1} t_i$; $a_{k-1} := \max_{1 \le i \le n_1} t_i$;
take $h := (a_{k-1} - a_1)/(k-2)$;
{ h *is the common length of the inner* $k-2$ *intervals of the histogram*};
for $i := 1$ **to** n_1 **do**
begin *calculate* $j := [(t_i - a_1)/h] + 2$; *take* $\nu_j := nu_j + 2$; **end;**
{$[z]$ *is the integer part of* z} *take* $a_0 := a_1$; $a_k := a_{k-1}$;
initialize $i := n_1 + 1$;
repeat
generate a sample of size n *on* X, $x_1, ..., x_n$;
calculate $t := t(x_1, ..., x_n)$; *take* $i := i + 1$;
if $t < a_0$ **then begin** $a_0 := t$, $\nu_0 := \nu_0 + 1$; **end;**
if $t > a_{k-1}$ **then begin** $a_k := t$, $\nu_k := \nu_k + 1$; **end**
else begin *take* $j := [(t - a_1)/h] + 2$, $\nu_j := \nu_j + 1$;**end;**
until $i = N$;

By a liniar interpolation between the values a_{i_0}, a_{i_0+1}, for which $\sum_{i=1}^{i_0} \frac{\nu_i}{N} \le 1 - \alpha < \sum_{i=1}^{i_0+1} \frac{\nu_i}{N}$, one can estimate the upper quantile t_α. If the test power is needed, the procedure is similar: the *"generate"* statements in the previous algorithm assume that X_i are simulated under the *alternative* hypothesis $nonH$.

A special case arises for the *Kolmogoroff-Smirnoff* test of goodness of fit. In this case the statistic is $t = \sup_x |F(x) - F_n(x)|$, where $F(x)$ is the *theoretical* cdf of X and $F_n(x)$ is the *empirical* cfd of X i.e. $F_n(x) = (\sum_{i, x_i < x} \nu_i)/n$. It is known that the cdf of t does not depend on F. Therefore, to estimate the *test power* $\pi = P(t > t_\alpha | nonH)$, we need to simulate $x_i, 1 \le i \le N$, in any alternative hypothesis, estimate $F_n(x)$ and then calculate t with F =the real (*null*) cdf. The previous algorithm will produce a histogram on this t. Then, with t_α known, the test power π is easily estimated from this histogram.

For the problem of the two samples, (i.e. to test $H : F_0 = G_0$), the test power is estimated by also using a histogram of $t_{n,m} = \sup_x |F_n(x) - g_m(x)|$, where $F_n(x), G_m(x)$ are empirical cdf's obtained by simulating two different (known) cdf's F and G. The same technique will be used for the *Cramer-von Mises* test or *Anderson-Darling* test.

4.3 The Bootstrap Technique

The bootstrap method, introduced by Efron, consists in *resampling* (or reuse of a sample) as it will be underlined in the following. Let $\mathbf{x} = (x_1, x_2, ..., x_n)$ be a sample on the rv X which has the cdf $F(x)$ and assume that we have to estimate a parameter $\theta(F)$ of X. *The bootstrap* technique consists in *resampling* the initial sample. In other words, if $\hat{F}(x)$ is the empirical cdf of X, obtained with the given sample, a *bootstrap sample* is $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)$, obtained by simulating $\hat{F}(x)$. Usually the bootstrap sample is obtained by the following algorithm:

take $i := 0$
repeat
generate $u \sim unif(0,1)$; *take* $j := [nu] + 1$; $i := i + 1$; $x_i^* := x_j$;
until $i := n$.

Therefore the bootstrap sample is obtained by extracting with *replacement* n values from the initial sample. Usually, B bootstrap samples $\mathbf{x_1^*}, \mathbf{x_2^*}, ..., \mathbf{x_B^*}$ are considered. The bootstrap samples are used in solving various statistical problems. Here we will give only some examples.

Estimating the mean square error. Assume that the parameter $\theta(F)$ can be estimated by $g(x_1, ..., x_n)$. If the parameter $\theta(F)$ is estimated by $\theta(\hat{F})$, we say that the *plug-in* principle of estimation has been used.The mean square error of the estimate is

$$MSE_F(g) = E_{\hat{F}}[(g(x_1, ..., x_n) - \theta(\hat{F}))^2]. \tag{4.5}$$

The algorithm for estimating $MSE_F(g)$ is:

for $b = 1$ **to** B **do**
begin *generate a bootstrap sample* $\mathbf{x}_b^*$; *calculate* $\hat{\theta}(b) = g(\mathbf{x}_b^*)$; **end;**
calculate the arithmetic mean and the variance of the variates $\hat{\theta}(b), 1 \leq b \leq B$

$$\overline{\theta}^* := \frac{\sum_{b=1}^{B} \hat{\theta}(b)}{B}; \ \widehat{MSE}_{\hat{F}}(g) := \sqrt{\frac{\sum_{b=1}^{B} (\hat{\theta}^*(b) - \overline{\theta}^*)^2}{B-1}}. \tag{4.6}$$

Confidence intervals based on bootstrap percentiles. A confidence interval for $\theta(F)$ can be obtained according to the following procedure:

Generate a large number B of bootstrap samples $\mathbf{x}_b, 1 \leq b \leq B$; calculate the bootstrap estimates $\hat{\theta}^*(b)$;
build-up a histogram of $\hat{\theta}^*(b)$;
for a given confidence level $\delta = 1 - 2\alpha$, determine the lower and upper α-quantiles $\hat{\theta}_\alpha, \hat{\theta}_{1-\alpha}$ as

$$\hat{\theta}_\alpha = \hat{G}^{-1}(\alpha),\ \hat{\theta}_{1-\alpha} = \hat{G}^{-1}(1-\alpha) \tag{4.7}$$

{ G is the cdf of $\hat{\theta}$ and G^{-1} is the inverse of G}.

The calculation of quantiles was illustrated in section 4.2.

Many bootstrap applications are based on the fact that, for large B, the arithmetic mean of bootstrap estimates $\hat{\theta}(b), 1 \leq b \leq B$ is *assymptotically normal.* This can induce a bootstrap confidence interval for $\theta(F)$ (with quantiles of the $N(0,1)$ distribution) namely

$$\overline{\theta}^* - \widehat{MSE}.z_{\alpha/2} \leq \theta(F) \leq \overline{\theta}^* + \widehat{MSE}.z_{\alpha/2},\ \frac{1}{2\pi}\int_{-z_{\alpha/2}}^{z_{\alpha/2}} e^{-u^2/2} = 1-\alpha. \tag{4.7'}$$

Bootstrap in linear models. Assume we have the liniar model

$$\Omega : \mathbf{y} = \mathbf{C}\beta + \mathbf{e} \tag{4.8}$$

where $\mathbf{C}$ is a $n \times p$ (design matrix), $\mathbf{y}$ is the *response* vector (of sampling values) and $\mathbf{e}$ is the error vector assumming to have the covariance matrix $\Sigma = \sigma^2\mathbf{I}$ with $\mathbf{I}$ the unit $n \times n$ matrix. The problem is to estimate the vector β and the variance σ from the observed data set $\mathbf{x} = (\mathbf{C}, \mathbf{y})$. Usually, the estimate $\hat{\beta}$ of β is obtained from the *least squares method* as the solution of the system of *normal equations*

$$\mathbf{C^T C}\hat{\beta} = \mathbf{C^T y}. \tag{4.9}$$

Let us denote $\mathbf{c_i}$ the lines of $\mathbf{C}$ (called covariates) and e_i the components of $\mathbf{e}$. Here the bootstrap samples could be obtained in two ways:by bootstrapping the data set $\mathbf{x}$ obtaining the bootstrapped data set $\mathbf{x}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, ..., \mathbf{x}_n^*), \mathbf{x}_i^* = (\mathbf{c}_i, y_i)$, or by first estimating β from the original data set $\mathbf{x}$, then calculating the residuals $\hat{e}_i = y_i - \mathbf{c}_i\hat{\beta}$, bootstrapping $\hat{e}_i$ as $\hat{e}_i^*$ and taking as bootstrap sample

$$\mathbf{x}^* = \{(\mathbf{c_1}, \mathbf{c_1}\hat{\beta} + \hat{\mathbf{e}}_\mathbf{1}^*), ..., (\mathbf{c}_n, \mathbf{c}_n\hat{\beta} + \hat{e}_n^*)\}. \tag{4.10}$$

The first method consists in bootstrapping pairs, while the second consists in bootstrapping residuals.

It is reported that, when the probability distribution of errors is depending on the covariates $\mathbf{c}_i$ (which might be the general case), the pairs bootstrapping is better to be used.

5 Use of Simulation in Numerical Calculation

5.1 Generalities on the Monte Carlo Method

The Monte Carlo method consists in estimating the solution θ of a numerical problem by using a suitable sample of a random variable or of a stochastic process ξ.(See Ermakov-1971, Fishman- 1996, Gentle 1998, Ripley-1986, Ross-1997, Roberts and Casella-1999). The function $\tau(\xi)$ such as $E[\tau(\xi)] = \theta$, is called a *primary estimator* of θ. The solution θ is then estimated by

$$\overline{\tau} = \frac{1}{n} \sum_{i=1}^{n} \tau(\xi_i) \tag{5.1}$$

where $\xi_i, 1 \leq i \leq n$ is a sample of ξ simulated by the computer. The $\overline{\tau}$ is called a *secondary estimator.* The minimum sample size n_0 can be calculated by using Tchebysheff's inequality as in section 4.1. If the approximation error ϵ is given and δ is a given large probability to obtain an error less then ϵ, then the minimum sample size is

$$n_0 = \{\frac{t_\delta^2 \sigma^2}{\epsilon^2}\} + 1,\ t_\delta = \sqrt{\frac{1}{1-\delta}}\ Var[\tau(\xi)] = \sigma^2 \tag{5.2}$$

where $\{z\}$ is the nearest integer to z.

5.2 Evaluating an Integral

5.2.1 Crude Monte Carlo

Assume, without loss of generality, that we have to calculate the integral

$$\theta = \int_0^1 f(x)dx. \tag{5.3}$$

Note that this can be written as $\theta = E[f(U)]$, where U is uniform $(0, 1)$. Therefore, a primary estimator is $\tau = f(U)$, and θ can be estimated by the corresponding secondary estimator. This method (based on $\xi = U$), is called the *crude Monte Carlo method.* Of course, for calculating an integral over some real interval (a, b), a change of variable can reduce the problem to the interval $(0, 1)$. Note also that if $\sigma^2 = Var(f(U))$ exists, then the minimum

sample size n_0 can be calculated. Note also that the Monte Carlo method is highly recommended for calculating a multiple integral in the form

$$\theta = \int\limits_D f(x)dx,\ D \in R^k,\ \theta = mes(D)\int_D \frac{f(x)dx}{mes(D)} = mes(D)E[f(\mathbf{V})] \qquad (5.3')$$

where $\mathbf{V}$ is a random vector uniformly distributed on D and $mes(D)$ is the measure of D.

Example 5.1 Let us calculate the m ultiple integral

$$I = \int_D f(x,x,z;x',y',z')dxdydzdx'dy'dz',\ f(x,y,z;x',y',z') =$$

$$\alpha \log d + \beta,\ d = \sqrt{(x-x')^2 + (y-y')^2 + (z-z')^2},$$

and $D = \Delta \times \Delta,\ \delta = [a,a_1] \times [b,b_1] \times [c,c_1],\ D \subset R^3$.

(Such a calculus can be reqired in Geostatistics). Such an integral is difficult to be calculated by quadrature formulae. Therefore it will be calculated by using Monte Carlo methods.

The integral I can be written in the form

$$I = (mes(\Delta)^2)\int_D \frac{1}{mes(\Delta)^2} f(x,y,z;x'y'z')dxdydzdx'dy'dz' = mes(\Delta)^2 I_1, \qquad (5.3")$$

$$mes(\Delta) = |x-x'||y-y'|z-z'| > 0.$$

The integral in the last formula can be written in the form

$$I_1 = E\left[f(V;V')\right],\ V,V' - uniform\ on\ \Delta,\ independent. \qquad (5.3''')$$

Now, the crude Monte Carlo procedure for calculating I_1 (and I) is obvious.

The sample size n can be previously determined according to formula (4.2) or (5.2) and discussions regarding it.

Exercice. Calculate the integral

$$I = \int_D f(x,y,z)dxdydz, f(x,y,z) = \sqrt{x^2+y^2+z^2}(cosx + siny + cosz)$$

where D is the sphere $D = \{(x,y,z); x^2+y^2+z^2 \leq 1\}$.

Hint. Note that $|f(x,y,z)| \leq m = 3\sqrt{3}$. Now, determine the sample size n according to (4.2) with $\sigma = m$. Then, generate points $P_i = (x_i,y_i,z_i)', 1 \leq i \leq n$ and apply crude Monte Carlo method.

Finally, note that the error of the estimator $\overline{\tau}$ depends on the variance of the primary estimator $\sigma^2 = Var(\tau)$. If, for a given problem, one can find a primary esimator having a smaller variance then the variance of the

crude Monte Carlo, then one says that this last primary estimator produces a *variance reduction.*

5.3 Variance Reduction Techniques

5.3.1 Importance Sampling

Note that the variance of τ in the case of crude Monte Carlo depends on the *variation* of the function $f(x)$ on the interval (a, b). One primary estimator which can reduce this variation can be obtained if we choose a pdf $p(x)$ over $(0, 1)$ which has a *shape* similar to $f(x)$. The pdf $p(x)$ is called *importance function* and the sample of this pdf is called *importance sample.* Note that

$$\theta = \int_0^1 p(x)\frac{f(x)}{p(x)}dx = E[\frac{F(Y)}{p(Y)}], \tag{5.3}$$

where Y is the random variable having the pdf p. If $p(x)$ is properly chosen, then $f(x)/p(x) \approx const$ and the primary estimator $\tau_i(Y) = f(Y)/p(Y)$ has a variance $Var(\tau_i) = \sigma_i^2 < \sigma^2$. Note that the method is highly recommended in the multivariate case.

Example 5.2. Let us calculate the integral

$$I = \int_0^\infty \int_0^\infty (cos(x) + sin(x))^3 e^{-x^2-y^3}dxdy = \int_0^\infty \int_0^\infty f(x, y)dxdy. \tag{5.3'}$$

This is a double improper integral and it is convergent because

$$|(cos(x) + sin(x)^3)e^{-x^2-y^3}| < 8e^{-x^2-y^3}$$

and the last function is integrable.

In order to calculate I by importance sample, we must select a **good** importance function which is to be a pdf, easy to simulate. Note that such a function could be

$$p(x, y) = e^{-x-y}.$$

The random vector (X, Y) withe pdf $p(x, y)$ has components exponential $Exp(1)$ distributed and independent, therefore (X, Y) can be straightforward generated. Because I is in the form

$$I = E\left[\frac{f(X, Y)}{p(X, Y)}\right] \tag{5.3"}$$

it results that a primary estimator is

$$(cos(x) + sin(x)^3)e^{-x^2-y^3+x+y}.$$

Note that this primary estimator is integrable, (because the previous function is bounded on $(0, \infty) \times (0, \infty)$), therefore the importance sampling procedure can be applied.

Exercice. Calculate the integral

$$I = \int_0^\infty \int_0^\infty f(x,y)dxdy : f(x,y) = \frac{1}{x^{2a}+y^{2a}}, a > 1.$$

Hint. Use the importance sampling. Take as importance density

$$p(x,y) = \frac{a(a-1)}{1+x+y)^{a+2}}.$$

The primary estimator is

$$\frac{f(x,y)}{p(x,y)} = \frac{1+x+y)^{a+2}}{a(a+1)(1+x^{2a}+y^{2aq})}$$

which is an integrable function on $(0,\infty)\times(0,\infty)$.

5.3.2 Antithetic Variates

Another variance reduction technique (called *antithetic variates*) is obtained as follows: consider the primary unbiased estimator τ and another primary unbiased estimator τ_1 and take as a new primary unbiased estimator

$$\psi = \tfrac{\tau+\tau_1}{2}. \tag{5.4}$$

If $Cov(\tau,\tau_1) = \sigma_{\tau,\tau_1} < 0$ (Cov denotes the *covariance*), then the variance $Var(\psi) = \sigma_\psi^2$ can be made less then σ^2, i.e. the variance reduction is obtained. If $\tau = f(U)$ is the crude Monte Carlo unbiased estimator, then one can take $\tau_1 = f(1-U)$, for which $Var(\tau_1) = \sigma^2$. This gives a negative covariance $\sigma_{\tau,\tau_1} < 0$, implying the variance reduction: $\sigma_\psi^2 < \sigma^2/2$.

Exercice. Calculate the integral

$$I = \int_0^1 f(x)dx : f(x) = e^{x^a}(cos^2x + sinx), a > 0.$$

Hint. Apply the method of antithetic variates. The procedure is the following:

- Calculate n using (4.2) with $\sigma + m = 2e$ where $|f(x)| \le m$;
- Generate random numbers $U_i, 1 \le i \le n$;
- Calculate $V_i = 1 - U_i$ and

$$\tau_{1i} = e^{U_i^a}(cos^2U_i + sinU_i); \tau_{2i} = e^{V_i^a}(cos^2V_i + sinV_i);$$

- Calculate the primary estimator $\tau_i = \frac{\tau_{1i}+\tau_{2i}}{2}, 1 \le i \le n$;

The secondary estimator of I is

$$I_n = \frac{1}{n}\sum_{i=1}^n \tau_i.$$

5.3.3 Control Variates

Another variance reduction technique (called *control variates*) consists in using an approximate integrable function ϕ, such as $Var(\phi(U)) < \sigma^2$, $\mu = \int_0^1 \phi(x)dx$, where μ is finite. Then $\theta = \int_0^1 f(x)dx - \int_0^1 \phi(x)dx + \mu$ and therefore it is necessary to estimate only the parameter $\theta - \mu = \nu$. The primary estimator in this case can be taken $\tau_a = f(U) - \phi(U)$, which also gives a variance reduction.

5.4 Solving Operatorial Equations

In order to solve operatorial equations, some notions related to *Markov Chains* are necessary. Consider a sequence $X_1, X_2, ...$ of random variables. Interpret the value of X_n as the *state of the system* at time n and assume that the set of possible states is $\{1, 2, ..., N\}$. If the conditional probability $P(X_{n+1} = j|X_n = i, X_{n-1} = i_1, ...) = P(X_{n+1} = j|X_n = i) = P_{ij}$ (i.e. the probability to pass from state i at time n to state j at time $n + 1$ depends only on the previous state i), we say that $\{X_n, n \geq 0\}$ is a *Markov chain* with transition probabilities $P_{ij}, i, j = 1, ..., N$. The elements of the transition matrix $P = P_{ij}$ satisfy the condition

$$\sum_{j=1}^{N} P_{ij} = 1,\ i = 1, 2, ..., N. \tag{5.5}$$

As the transition probabilities P_{ij} do not depend on time n, the Markov chain is *homogeneous.* The previous formula leads to the following algorithm for simulating a random state of a Markov chain.

Assume the previous state is i. Calculate $F_k = \sum_{s=1}^{k} P_{is}, k = 1, ..., N$;
Generate $u \sim unif(0, 1)$ *and take* $j = 0$;
while $F_j > u$ **do** $j := j + 1$; {*j is the generated state.*}

If the algorithm is repeated (with a previous update $i := j$) one obtains a *trajectory* of the process, $i, j_1, j_2, ...$ If, for a Markov chain, there is a state i_0 such that $P_{i_0 i_0} = 1$, then i_0 is an *absorbing* state. (If the chain enters this state it will never leave it).
A Markov chain is ergodic if, when the time $n \to \infty$, the probability distribution of states is stationary (i.e. is *unchanged* in time).

5.4.1 Solving Systems of Liniar Equations

Every system of liniuar equations can be written in the form

$$\mathbf{x} = \mathbf{Hx} + \mathbf{b}, \mathbf{H} = |h_{ij}|, \tag{5.6}$$

where $\mathbf{x}$ and $\mathbf{b}$ are $k \times 1$ matrices, and $\mathbf{H}$ is a $k \times k$ nonsingular matrix with $\| \mathbf{H} \| < 1$ ($\| \mathbf{H} \|$ is *the norm* of the matrix $\mathbf{H}$). The Monte Carlo procedure for solving the system consists first of associating a Markov chain with $k+1$ states having transition probabilities satisfying conditions:

$$P_{ij} \neq 0 \, if \, h_{ij} \neq 0, 1 \leq i, j \leq k, \; \sum_{j=1}^{k} P_{ij} = p_i < 1 \tag{5.7}$$

$$P_{i,k+1} = p_i, 1 \leq i \leq k, \; P_{k+1,k+1} = 1, \; P_{k+1,i} = 0, i \leq k. \tag{5.7'}$$

Consider now

$$v_{ij} = \begin{cases} \frac{h_{ij}}{P_{ij}}, \; if \; P_{ij} \neq 0 \\ 0, \; if \; P_{ij} = 0. \end{cases} \tag{5.7"}$$

The Monte Carlo algorithm for estimating the solution $x_i, i - fixed$, is the following:

input N; $\{N$ *is the sample size*$\}$; $j := 0; x_i := 0$;
repeat
generate a trajectory $\gamma = (i, i_1, i_2, ..., i_m, k+1)$ *of the Markov chain (i.e. until absorbtion)*;
calculate $V_m(\gamma) = v_{ii_1} v_{i_1 i_2} ... v_{i_{m-1} i_m}; X(\gamma) = V_m(\gamma) \frac{a_{i_m}}{p_{i_m}}$;
take $x_i := x_i + V_m(\gamma); j := j + 1$;
until $j = N$;
take $x_i := x_i / N$.

One theorem (see Ermakov-1971) says that $V_n(\gamma)$ is a primary estimator of the component x_i and the algorithm gives the estimate of x_i as the arithmetic mean of the primary estimator. This algorithm is mainly used when it is necessary to obtain only the component x_i of the solution.

In order to obtain other components $x_j, j \neq i$, the algorithm can be applied for each $j, 1 \leq j \leq k$. An alternative is to use an uniform distribution for the initial states, calculate the primary estimator for each trajectory starting in the initial random state and then calculate the secondary estimators for all the states. (In this case, the sample size N must be large enough to make sure that several trajectories starting from each state are obtained, and these must be counted in $N_1, N_2, ..., N_k$).

5.4.2 Solving Integral Equations

In order to find a numerical solution of the *integral equation:*

$$f(x) = g(x) + \int_a^b K(x, y) f(y) dy \tag{5.8}$$

where g is a known function and K is a known *kernel* (f is the unknown function), one method is to start with the discretization of the equation (on a gird of points $a = a_1 < a_2 < ... < a_k = b$), reducing the problem to solving a system of liniar equations in the form (5.6) where $\mathbf{x} = (f(a_1), ..., f(a_k))$, $\mathbf{H} = \| K(a_i, a_j) \|$, and $\mathbf{b} = (g(a_1), ..., g(a_k))$.

When the functions involved belong to the space of functions $L^2[a, b]$ (i.e. are squared integrable), the problem can also be studied in a similar manner. Assume that $\{\phi_0, \phi_1, ...\}$ is a *base* of functions of L^2 and consider the expansions

$$\begin{array}{l} g(x) = a_0\phi_0(x) + a_1\phi_1(x) + ... \\ \int\limits_a^b K(x, y)\phi_j(y)dy = h_{0j}\phi_0(x) + h_{1j}\phi_1(x) + ... \\ f(x) = x_0\phi_0(x) + x_1\phi_1(x) + ... \end{array} \tag{5.9}$$

If the sums in formulae (5.9) have a finite number of terms (i.e. they approximate the functions with a finite number of terms, say k), then the problem is again reduced to a system of linear equations of the form (5.6), for which the Monte Carlo solution is known from the previous section.

Another method consists in using an *absorbing Markov process* having a *transition* pdf $P(x, y)$ such as

$$P(x, y) > 0 \ if \ K(x, y) \neq 0, \quad \int_a^b P(x, y)dy < 1. \tag{5.10}$$

Consider now the notations

$$p(x) = 1 - \int\limits_a^b P(x, y)dy, \quad v(x, y) = \frac{K(x,y)}{P(x,y)}. \tag{5.10'}$$

Generate a trajectory $\gamma = (x_0, x_1, ..., x_k)$ of states for the absorbing Markov chain as follows. Generate first an initial state x_0 with the initial distribution $\pi(x)$ (this could be a density). Then generate the next state x_1 with pdf $P(x_0, x)/(1-p(x))$. Given the non-absorbing state x_{i-1}, generate the state x_i with pdf $P(x_{i-1}, x)/(1-p(x))$, which is absorbing with probability $p(x_i)$ (and non-absorbing with probability $1 - p(x_i)$). When the last state is absorbing (say x_k), then the trajectory is completed. The primary estimator is

$$X(\gamma) = \frac{V_k(\gamma)}{p(x_k)}, \ where \ V_k(\gamma) = v(x_0, x_1)...v(x_{k-1}x_k).$$

One can show that if

$$\| K \| = \sup_x \int\limits_a^b |K(x, y)|dy < 1 \tag{5.11}$$

then $X(\gamma)$ is an unbiased estimator for $f(x_0)$.
The algorithm is similar to the previous one. Details on these methods are found in the mentioned literature.

5.5 Monte Carlo Optimization

Another type of numerical problem which can be solved via Monte Carlo is the problem of optimization in the following form: find the optimum of a function $h(\mathbf{x}) \in R$ when $\mathbf{x} \in D \subset R^k$. In general, the set D is bounded. (See Robert and Casella-1999 for details). Without loss of generality assume that the problem is: find $\mathbf{x}^* \in D$ such that

$$h^* = \max_{\mathbf{x} \in D} h(\mathbf{x}) = h(\mathbf{x}^*),\ h(\mathbf{x}) = E[h(\mathbf{x}, Z)] \tag{5.12}$$

where Z is a random variable. Any deterministic maximization problem can be formulated as (48) (except for the last formula involving expectation). This suggests that the maximum point $\mathbf{x}^*$ can be obtained by a *random search* procedure such as:

$i := 1$; *generate* $\mathbf{v}_1 \sim unif(D)$; *take* $M = h(\mathbf{v}_1)$; *input* $\epsilon > 0$;
repeat
take $\mathbf{v}_1 = \mathbf{v}_i$; $i := i + 1$; *generate* $\mathbf{v}_i \sim unif(D)$;
calculate $M := max[M, h(\mathbf{v}_i)]$; *if* $M = h(\mathbf{v}_i)$ *then* $\overline{\mathbf{x}}^* = \mathbf{v}_i$;
until $|\mathbf{v}_i - \mathbf{v}_1| < \epsilon$.

One theorem (of Gnedenko-1943) says that if h is continuous on D, then M converges to h^* and $\overline{\mathbf{x}}^*$ converges to $\mathbf{x}^*$.
When $h(\mathbf{x})$ is defined as an expectation and Z has the pdf $f(z, \mathbf{x})$ then, instead of using uniform points $\mathbf{v}_i$ on D we can use *search* technique, based on random variates $z_1, z_2, ..., z_n$, from the pdf $f(z, \mathbf{x})$, involved in definition of h. Now the maximization algorithm becomes:

take $i := 1$; *select* $\mathbf{x}_1 \in D$, uniformly distributed on D; *input* m *and* $\epsilon > 0$;
repeat
generate $z_1, ..., z_m$ with the pdf $f(z, \mathbf{x}_i)$;
calculate $\hat{h}_i = \dfrac{\sum_{j=1}^{m} h(z_j, \mathbf{x}_i)}{m}$;
calculate $\hat{h}^* = \max_i \hat{h}_i$;
if $\hat{h}^* = h(\mathbf{x}_{i_0})$ *then* $\hat{\mathbf{x}}^* = \mathbf{x}_{i_0}; \mathbf{x}_1 := \mathbf{x}_i$; *update* $i := i + 1; \mathbf{x}_i := \hat{\mathbf{x}}^*$;

until $|\mathbf{x}_1 - \mathbf{x}_i| < \epsilon$;

This kind of optimization problem is frequently used in statistics when the *maximum likelihood method* is applied for estimating parameters. In this

case Z is the rv and $\mathbf{x}$ is the parameter to be estimated. A similar problem arises when we estimate parameters of a *posterior* distribution in *Bayesian* statistical analysis.

In the deterministic case, when $h(\mathbf{x}) > 0$ is integrable, one can build up an *importance sampling* procedure using the importance pdf defined as:

$$g(\mathbf{x}) = \begin{cases} \frac{h(\mathbf{x})}{H} \; if \; \mathbf{x} \in D \\ o, \; otherwise, \; H = \int\limits_D h(\mathbf{x}) d\mathbf{x}. \end{cases} \tag{5.13}$$

The previous algorithm is then simply modified just by using vectors $\mathbf{z}_i$ simulated with pdf $g(\mathbf{x})$. One can prove that, in this case (for h-continuous), the $\hat{h}^*$ converges to h^*, faster than M.

One connected optimization problem (which also uses the notion of a Markov chain) is the one called *simulated annealing.*(See Pham and Karaboga-2000, Robert and Casella-1999 and Ross-1997). Annealing is a physical process of heating up a solid and then cooling it down slowly until it cristallyses. An optimization algorithm based on simulating the annealing process first needs to use the probability distribution of system *energy* E at a given *temperature* T. (Usually this distribution is $P(E) = e^{-E/(kT)}$, where k is Boltzmann's constant). Then, the representation of the solutions to the problem must be established. Of course, the definition of the *cost* function E (which is to be minimized), is required . The definition of the generation algorithm for the neighbours is another requirement.(In the physical annealing case this distribution is Boltzmann's probability from above; otherwise it is a state of a Markov chain). Finally, a *cooling* schedule of the "temperature" T is required. The general form of the algorithm is:

select an initial solution, x_0 *i.e. an initial state of the chain, maybe from an initial distribution; take* $n = 0$; *input* T_0;
repeat
generate y, *maybe from Boltzmann's distribution or from another fixed distribution, conditionally on* x_n; *then, the new state of the chain is:*

$$x_{n+1} = \begin{cases} y, \; with \; probability \; min(1, e^{-\Delta E/T_n}) \\ x_n \; otherwise, \Delta E = E(y) - E(x_n); \end{cases} \tag{5.14}$$

evaluate the energy as a function $E(x_{n+1})$;
update T (*cooling schedule*) *which is usually in the form* $T_{n+1} := cT_n, 0 < c < 1$); *take* $n := n + 1$;
until *energy is minimum.*

5.6 Markov Chain Monte Carlo

A *Markov Chain Monte Carlo (MCMC) method* for the simulation of a distribution f is any method producing an ergodic Markov chain $\{X_n\}$ whose stationary distribution is f. (See Robert & Casella-1999 and Ross-1997).
• **The standard Metropolis-Hastings (M-H) algorithm** for simulating a random vector $\mathbf{X} \sim pdf\, f(\mathbf{x})$ is based on a *conditional density* $q(\mathbf{y}|\mathbf{x}), \mathbf{x}, \mathbf{y} \in A \subset R^k$ which is symmetric and easy to be simulated. It may be a transition density of a Markov process. Denote

$$\rho(\mathbf{x}, \mathbf{y}) = min\{\frac{f(\mathbf{y})}{f(\mathbf{x})} \frac{q(\mathbf{x}|\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}, 1\}$$

The M-H algorithm is the following:

take $t = 0$ *-the initial time,* $\mathbf{x}_t$ *-the initial state; input a large* N;
repeat
generate $\mathbf{y}_t \sim q(\mathbf{y}|\mathbf{x}_t)$;
generate $u \sim unif(0, 1)$; **if** $u < \rho(\mathbf{x}_t, \mathbf{y}_t)$ **then** $\mathbf{x}_{t+1} := \mathbf{y}_t$ **else** $\mathbf{x}_{t+1} := \mathbf{x}_t$;
take $t := t + 1$;
until $t = N$.

The sampling (simulated) value of $\mathbf{X} \sim f$ is $\mathbf{x} := \mathbf{x}_N$. There are many particular forms of the Metropolis-Hastings algorithm; one is when $q = q(\mathbf{x}, \mathbf{y})$ is the transition density of the Markov chain; some others refer to transition kernels in the form $q(\mathbf{x}, \mathbf{y}) = g(\mathbf{x} - \mathbf{y}), g$-symmetric (the Markov process is, in this case, a *random walk*) and so on. Partricular forms are obtained when f is a particular univariate distribution.

•**The Gibbs sampler** is one particular form of M-H algorithm. It is assumed that, in order to simulate the $k \times 1$ vector $\mathbf{X} \sim$ *pdf* $f(\mathbf{x}), \mathbf{x} \in A, \mathbf{x} = (x_1, ..., x_k)$, one can simulate the univariate conditional densities $f_1, f_2, ..., f_k$, such as $\{X_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_k\} \sim f_i(x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_k)$. If for a fixed i one denotes $\mathbf{y} = (x_1, ..., x_{i-1}, x, x_{i+1}, ..., x_k)$ and takes

$$q(\mathbf{x}, \mathbf{y}) = \frac{1}{k} f_i(x|x_j, j \neq i) \tag{5.15}$$

in the Metropolis-Hastings algorithm, then one obtains the Gibbs sampler as:

take $\mathbf{x} \in A$, for which $f(x) > 0$
for $i = 1$ **to** k **do**
begin
generate $x \sim f_i(x|x_j, j \neq i)$
if $\mathbf{y} \in A$ **then** *take* $x_i := x$ **else leave the value of** x_i *as it is;*
end.

Note that the output of this algorithm is a sampling value $\mathbf{x}$ of $\mathbf{X}$.
There are many applications of the MCMC method. (See the mentioned literature: Robert-Casella-1999,Ross-1997). Here we will mention an usual one for statistics.
Assume we have to estimate $\theta = E[X]$ and X is simulated with the M-H algorithm. Denote $x_1, x_2, ..., x_N$ (with large N) a trajectory of the Markov chain produced by the M-H algorithm. Assume that states $i \geq p$ are almost stationary.Then the estimate of θ and its mean square error MSE are

$$\hat{\theta} = \frac{1}{n-p} \sum_{i=p+1}^{N} h(x_i), \quad MSE = E\left[\left(\hat{\theta} - \theta\right)^2\right].$$

To estimate the MSE the *batch means* method is used, namely, break up the $N - p$ states intyo s batches of size $r, s = (N - p)/r$ (assuming that s is an integer!). Denote by $Y_j, j = 1, 2, ..., s$ the sampling means of the batches, which are identically distributed with $Var(Y_j) = \sigma^2$, and calculate its estimate as $\hat{\sigma}^2$. Then the estimate of MSE is $\widehat{MSE} = \hat{\sigma}^2/s$.

6. Introduction to queueing models

6.1 Preliminaries

A queueing system is ome in which *arrive* discrete units (called *customers*, which take a *service.* When the flow of customers arriving in the system is large and the system service is slow, then, the surplus of customers in the system join a *wayting queue* where they wait until the service is availlable. Such a queing system can be a shop selling goods for people, a bank which serves its customers, a raylway station which sels travelling tickets, a naval station (or port) which gives service to ships, n asembler line in a factory and so on. The manager of a queueing systemIs interesed both in decreasing the wayting time of customers in the system, and in decreasing the the "idle time " of service stations. But these two objectives of the manager are in strife:when one decreases, the other one increases.In this cae the idea is to introduce some costs related to the two elements (wayting time and idle time) and to tray to ballance them (i. e. the to costs to be equal or close each other).In order to built up a mathematical model for a queueing system, it is necesary to define first some variables and parameters of the system.(Difference between varisbles and parameters is obvious: parameters keep constant values over large periods of time, while variables change their value during short intervals of time). Some of the variables are known-input (e.g. arrivals and services) and some other are unknown-output (e.g waiting time and idle time). The purpose of the model should be to determine the output elements in term of input ones. The problem now

is how to measure the arrivals and services. Both could be measured either by discrete (integer) numeric values or by continuous values.Arrivals could be measured by arrival time (AT), i.e. the interarrivel time of customers, or by integer number (NA), i.e. the mumber of units arriving per unit of time (the flow of arrivals). Similarly, the services could be measured by the duration of one service (ST) or by the number of customers served per unit time (SN). These are *imput* variables. All these input variables are random variables for which the probability distributions are assume to be known. The output variables, in the continuous version, are WT the curent wayting time of a customer and TID the current idle time of a service station (in the case when there are several stations!). The discrete output variables are WL-the current queue length and NID- the current number of idle stations. These autput variables are also random, but their probability distributions are unknown and in ideal situation they must be determined depending on distributions of input variables. Practically this is not allways possible and therefore we need to use moments (or expectations) of these distributions, which are *parameters.*

Input parameters could be expected arrival time $E[AT]$ or $E[NA]$-the expected number of customers arriving per unit of time, and, the expected service time $E[ST]$ or $E[SN]$- the expected number of customers served per unit time. The following relations are valid for input parameters:

$$E[AT] = \frac{1}{E[NA]},\ E[ST] = \frac{1}{E[SN]}. \tag{6.1}$$

Similarly, bewtween output parametrs there are relations:

$$E[WT] = \frac{1}{E[WL]},\ E[TID] = \frac{1}{E[NID]}. \tag{6.1'}$$

For calculating output parameters of a queueing model we use a discrete stochastic process $N(t)$ which is the number of customers in the system at the time $t, t > 0$. Note that $N(t)$ consists in both the number of customers in wayting queues, plus the number of customers under service.

The key element in studying a queueing system is to derive the probability distribution of $N(t)$ in terms of input elements. The probability distribution of $N(t)$is in the form

$$N(t) : \begin{pmatrix} 0, & 1, & 2, & ..., & n, & ... \\ P_0(t), & P_1(t), & P_2(t), & ..., & P_n(t), & ... \end{pmatrix} \tag{6.2}$$

where $P_n(t) = P[N(t) = n]$, $n = 0, 1, 2, ...$ We shall denote a queueing model as folows

$$A/S/c : (L_q, d) \tag{6.3}$$

where A is an information on the probability of arrivals, S is an information on the distribution of services, c is the number of service station (chanels!) together with the topology of these stations when are many, L_q is the maximum length of the queue and d is the *discipline of service*. The topology concerning the $c > 1$ service stations could be *serial*, when the service is performed by crossing all stations, *parallel* when the service is fulfilled by only one of the service stations, or *network* when the sevice stations are the nodes of a directed graph. The disciplene of service could be FIFO (in order of arrivals), LIFO (in the inverse order of arrivals), with priorities (when some important customers are served before others), or some other rules of service (e.g. when some costomers do not wait in the queue more than some maximum time). Note that all elements in the previous notation of a queueing model are known and we must take them in consideration when searching for the probability distribution of $N(t)$.

If we know probabilities $P_n(t)$ we can determine some interesting output parameters as follows

$$E[N(t)] = \sum_{n=0}^{\infty} nP_n(t), E[WL] = \sum_{n=c}^{\infty}(n-c)P_n(t), E[NID] = \sum_{n=0}^{c}(c-n)P_n(t). \tag{6.4}$$

In the following we introduce the *birth snd death process*, which is the mathematical instrument to solvr a queueing model.

6.2 Birth and Death Processes

The discrete stochastic process $N(t)$, $t > 0$ is a birth and death process if it has the following properties

(a). $N(t)$ counts random events occuring in time such as $N(t)$ is the number of events occuring on time interval $[0, t]$; furhtermore for $t_1 < t_2$, $N(t_2 - t_1) = N(t_2) - N(t_1)$ and for $t_1 < t_2 < t_3 < t_4$ we have $N(t_2 - t_1)$ is stochasticaly independent fron $N(t_4 - t_3)$; (one says that the process is with *independent stochastic increments.*

(b). $P([N(t + \Delta t) = n + 1]/[N(t) = n]) = \lambda_n \Delta t + O(\Delta t)$, $n \geq 0$;

(c). $P([N(t + \Delta t) = n - 1]/[N(t) = n]) = \mu_n \Delta t + o(\Delta t)$, $n \geq 0$;

(d). For $i > 1$, $P([N(t+\Delta t) = n \pm i]/[N(t) = n]) = O(\Delta t)$. In (b),(c), (d) the notation $o(\Delta t)$ denotes a quantity which is *neglected* (i.e. very small) for a small increment $\Delta t > 0$. The class of functions $\mathcal{O} = \{o(\Delta t)\}$ has properties

$$f_1(t)o_1(\Delta t) + f_2(t)o_2(\Delta t) \in \mathcal{O}, \forall |f_i(t)| < \infty, o_1, o_2 \in \mathcal{O}.$$

$$o_1 + o_2 + ... + o_n \in \mathcal{O}, \forall o_i \in \mathcal{O} \tag{6.5}$$

$$\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$$

(i.e. $o(\Delta t)$ tends to zero faster then Δt.).

The prooerties (b),(c) say that on a short interval of time $[t, t+\Delta t]$ is born or die *only* one individual (i.e these are*random events* and the corresponding probabilities depend only on $\lambda_n, n \geq 0$ (the *intensity of birth* and on $\mu_n, n \geq 1$ (the intensity of death).

A birth and death process can describe the evolution of a queueing system, the evolution of a human population, the evolution of trafic of users into a computer network, and so on.

The properties (b),(c),(d) say that events counted by $N(t)$ are rare events. These properties of a birth and death process allow us to find a system of differential equations satisfied by the probabilities of $P_n(t) = P[N(t) = n]$. It results in the following theorem.

Theorem 6. 1 *The probabilities $P_n(t)$ of a birth and death process satisfy the following system of differential equations*

$$P_0'(t) = -\lambda_0 P_0(t) + \mu_1 P_1(t) \tag{6.6}$$

$$P_n'(t) = -(\lambda_n + \mu_n)P_n(t) + \lambda_{n-1}P_{n-1}(t) + \mu_{n+1}P_{n+1}(t),\ n \geq 1. \tag{6.6'}$$

Proof. For $n = 0$ if we use properties (a),(b),(c),(d) and properties (6.5) one obtain

$$P[N(t + \Delta t) = 0] = P[N(t) = 0](1 - \lambda_0 \Delta t - O(\Delta t))+$$

$$+P[N(t) = 1](\mu_1 \Delta t + o(\Delta t))(1 - \lambda_1 \Delta t - o(\Delta t)) + P[N(t) = n+i, i > 1]o(\Delta t)$$

and after some calculations we obtain

$$P[N(t+\Delta t) = 0] - P[N(t) = 0] = -\lambda_0 P[N(t) = 0]\Delta t + P[N(t) = 1]\mu_1 \Delta t + o(\Delta t)$$

and therefore

$$\frac{P_0(t + \Delta t) - P_0(t)}{\Delta t} = -\lambda_0 P_0(t) + \mu_1 P_1(t) + \frac{o(\Delta t)}{\Delta t}.$$

Making $\Delta t \to 0$ in the last formula we obtain (6.6). In a similar manner one can prove also (6.6'). The theorem is proved.

The process$\{N(t)\}$ is well defined by $\lambda_n > 0, n \geq 0$ (called *birth intensities*) and $\{\mu_n > 0, n \geq 1\}$ (called *death intensities*).

If we fix up *initial conditions* $P_0(0) = 1$, $p_i(0) = 0$, $i \geq 1$ then the system (6.6),(6.6') has a unique solution (a result known from the theory of differential equations. We need that $\{P_n(t)\}$ satisfy condition

$$\sum_{n=0}^{\infty} P_n(t) = 1,\ \forall t. \tag{6.6''}$$

The following theoren, due to Feller says

Theorem 6. 2 *The necessary and sufficient condition to be satisfied relation (6.6") is*

$$\sum_{n=o}^{\infty}\prod_{i=1}^{n}\frac{\mu_i}{\lambda_{i-1}}=\infty.$$

An interesting practical case is when the process becomes stationary, i.e. when

$$\lim_{n\to\infty}P_n(t)=p_n=const,\forall n\geq 0.$$

In the case of queueing models this means that after some period running in, the system becomes stable, i.e. $P_n(t)=p_n=const$. In this case, the system (6.6),(6.6') becomes

$$-\lambda_0 p_0+\mu_1 p_1=0 \tag{6.7}$$

$$-(\lambda_n+\mu_n)p_2+\lambda_{n-1}p_{n-1}+\mu_{n+1}p_{n+1}=0. \tag{6.7'}$$

If we denote $Z_k=-\lambda_{k-1}p_{k-1}+\mu_k p_k$, from (6.7) and (6.7') it results

$$Z_0=0,\ Z_{k+1}=Z_k=0,\ k\geq 1$$

and hence we have

$$p_k=\frac{\lambda_{k-1}}{\mu_k}p_{k-1}$$

i.e.

$$p_n=\prod_{i=0}^{n-1}\frac{\lambda_i}{\mu_{i+1}}p_0. \tag{6.8}$$

The condition $\sum_{n=0}^{\infty}p_n=1$ gives finaly

$$p_0=[1+\sum_{n=1}^{\infty}\prod_{i=0}^{n-1}\frac{\lambda_i}{\mu_{i+1}}]^{-1}. \tag{6.8'}$$

We note that is necessary to have

$$\sum_{n=0}^{\infty}\prod_{i=0}^{n-1}\frac{\lambda_i}{\mu_{i+1}}<\infty.$$

Example 6.1 We will show how the birth and death process can be used to solve the following queueing model

$$Exp(\lambda)/Exp(\mu)/1:(\infty;FIFO). \tag{6.9}$$

(When the distribution of arrivals and services are certain, the model is denoted

$$\bullet/\bullet/1:(\infty;FIFO). \tag{6.9'}$$

Therefore, the arrival time AT is exponentialy distributed $Exp(\lambda)$, the service time ST is exponentialy distributed $Exp(\mu)$, $\lambda \neq \mu$, there is only one service station, the maximum length of queue is ∞, and the discipline of service is FIFO (First In First Out).

Let us determine the intensities $\lambda_n, n \geq 0; \mu_n, n \geq 1$. Note that

$P[to\, arrive\, a\, customer\, on\, [0, \Delta t]$ is

$$P[AT \leq \Delta t] = 1 - e^{\lambda \Delta t} = 1 - (1 - \lambda \Delta t) + o(\Delta t) = \lambda \Delta t + o(\Delta t)$$

and therefore we have $\lambda_n = \lambda = const = \frac{1}{E[AT]}, \forall n \geq 0$.

In a similar manner one can found that $\mu_n = \mu, \forall n \geq 1$. Now if we denote $\rho = \frac{\lambda}{\mu}$ we obtain in the stationary case (according to (6.8),(6.8'))

$$p_n = \rho^n p_o,\ p_0 = [\sum_{i=0}^{\infty} \rho^i]^{-1} = 1 - \rho,$$

assuming $|\rho| < 1$ and therefore

$$p_n = (1 - \rho)\rho^n.$$

Note that ρ (called *traffic intensity*), is the ratio of the expected number of arrivals per unit time and the expected number of customers served per unit time; the queueing system can run *only* when $|\rho| < 1$.

Now we can calculate some interesting output parameters of the queueing model (6.9), namely

$$E[N(t)] = \sum_{n=0}^{\infty} n p_n = \rho(1 - \rho) \sum_{n=1}^{\infty} n \rho^{n-1} =$$

$$= \rho(1 - \rho) \frac{d}{d\rho} \left(\sum_{n=1}^{\infty} \rho^n \right) = \rho(1 - \rho) \frac{1}{(1 - \rho)^2} = \frac{\rho}{1 - \rho},$$

$$E[WL] = \sum_{n=2}^{\infty} (n - 1) p_n = \rho^2 (1 - \rho) \sum_{n=2}^{\infty} (n - 1) \rho^{n-2} =$$

$$= \rho^2 (1 - \rho) \sum_{n=1} \infty n \rho^{n-1} = \rho^2 (1 - \rho) \frac{d}{d\rho} \left(\sum_{n=1}^{\infty} \rho^n \right) = \frac{\rho^2}{1 - \rho},$$

$$E[WT] = E[ST] E[WL] = \frac{\rho^2}{\mu(1 - \rho)}, \tag{6.10}$$

$$E[NID] = \sum_{0}^{1} (1 - n) p_n = p_0 = 1 - \rho,$$

$$E[TID] = E[AT]E[NID] = \frac{1-\rho}{\lambda}. \qquad (6.10')$$

If we denote W the (random) time a customer spends in the queueing system then we have

$$E[W] = E[N(t)]E[ST] = \frac{\rho}{\mu(1-\rho)}.$$

For the queueing system is defined the so called *efficiency factor* as

$$I_e = \frac{E[W]}{E[ST]}.$$

Note that in this example AT and ST are exponentialy distributed. Usualy, in practice AT may be expinentialy distributed, while ST could have any other distribution. In this case the model (6.9) is diffiucult to be solved. Pollaczek stated a *conjecture* that for the model

$$Exp(\lambda)/B/1 : (\infty, FIFO)$$

where B means any distribution of ST for which $E[B], Var[B]$ exist, the efficiency factor I_e is

$$I_e = \frac{\rho}{1-\rho}(1 + C_s^2) \qquad (6.11)$$

where C_s^2 is the *variability coefficient* of ST defined as

$$C_s = \frac{\sqrt{Var[ST]}}{E[ST]}. \qquad (6.11')$$

This queueing model assumed exponential arrivals and services.

• **Use of discrete simulation.** In order to analyze the queueing system with any type of distribution for arrivals and services we need to built up a *simulation model.* Such a model consists in an algorithm which produces artifficial experiments via computer runs and then, these experiments (in fact samples on output variables) will be processed and analyzed giving solutions for the management of the real queuieing system described by the simulation model. The instruments used to built up the simulation model (i.e. algorithm) are *the clock time* and *agenda.*The clock time has a double purpose: to record the time elapsed of the system and to keep the correct order in time of the events produced by the simulation model. Agenda is a concept related to store the events produced by simulation. The clock time is increased by a finite number of values. After each increment of the clock, the simulation algorithm processes the events occured at that moment of time (these events define the *agenda of current events* -ACE). When the ACE is empty, the clock is incremented with some value and the process

is repeated for the new events fron ACE. When an event is processed, it cam produce another event (at a future moment of the clock, saying that the set of these events is the *agenda of future events*-AFE) or can cancel some events from ACE (this is the set of *canceled events*-CE). Therefore the agenda A has a dynamic evolution in the form

$$A = ACE \oplus AFE \ominus CE.$$

The clock time can be incremented in two main ways: with variable increments (called *variable increment clock time*) or with constant increments (called *constant increment clock time*). Initially the clock is zero. In the case of variable increment, the clock is incremented up to the time occurence of the first event in AFE. In the case of constant increment, the clock is incremented with a constant "hour" c. After incrementing the clock, the main cycle of the simulation algorithm selects the events from AFE having the occurence time equal to clock and introduces them in ACE. Then, the events from ACE are processed; when ACE is empty, the clock is advanced again and the process is repeated until clock takes a given T_{max} input value when the simulation ends. As an alternative, the simulation ends when the number of a type of simulated events takes a given value N_{max}. Sometime, instead of the variable clock tine we use the equivalent rule i.e. *next event rule*; in this case the end of simulation is determined by N_{max}

In the book by Zeigler and all (2000) is presented a general formal description of discrete simulation models, based on formal sytem theory, particularly on the so-called *system with external discrete events*. This formal description is a background for construction of discrete simulation languages. We do not come into details concerning this formal description of discrete simulation models; we resume description to the above short preserntation.

After this brief introduction to the ideas of building up simulation models we describe now the simulation model of the system (6.9'). Apart from the known variables AT, ST, WT, TID the simulation model uses variables:

TWT- total wayting time;

$TTID$-total idle time;

$ICOUNT$-an integer variable which counts the number of services performed;

$NS = N_{max}$-the numbes of customers to be served;

AWT- average wayting time=TWT/NS;

$ATID$-average idle time=$TTID/NS$;

The flowchart of the simulation algorithm is given in Fig.6.1. The rule of next event is used (i.e.an implicite variable clock time). The block (1) of the flowchart reads NS and input parameters for problability distributions

of AT. Initializations are

$$WT = TID = 0,\ TWT = TTID = 0,\ ICOUNT = 0,$$

Block (2) simulates an arrival. Because the rule of next event is used, block (3) fits the arrival time using wayting time, to allow selection of the first next event. Then, block (4) simulates a service time for the current customer. Block (5), according to discipline $FIFO$, selects the next event to be processed; when the arrival time is less then service time, then the customer has to wait (block (6) calculates current WT) or on the contrary block (7) calculates the idle time TID of the service station. Block (8) gives the condition to terminate the simulation. Finally, block (9) calculates output parameters AWT and $ATID$ and delivers results. After building up the simulation algorithm the problem is to transform it into a computer program. This can be done by using a general purpose language (like, Fortran, Pascal, C++, etc) or by using a specialized simulation language (like GPSS, Simula,etc). A general purpose langusge has the advantage that it alows more flexibility in processing and interpreting simulation experiments, while a simulation language allows a facile construction of the simulation algorithm (avoiding for instance the difficult handling with clock and agenda), but it has only limited possibility of analyzing simulated experiments. The flowchart presented is oriented to use a general purpose language.

The simulation model presented,can be changed for various hypotheses. For example AT and/or ST could have different distributions as; Gamma, Erlang, Lomax, Weibull, etc. They could have *mixture distributions* in the form

$$F(x) = \sum_{i=1}^{k} p_i F_i(x),\ p_i > 0, \sum_{i=1}^{k} p_i = 1. \tag{6.12}$$

In this case customers of the k classes coulkd have different priorities in service and the algorithm can be developed accordingly. To finish the construction of the simulation model, we need to answer the following question: is the maodel logically correct? Even if it is running well on the computer, this does not mean that it is solving well the problem. Therefore, we must do the *validation* of the model.

This can be done by using a mathematical model, whose solution is known in some particular case. For instance the mathematical model (6.9) gives formulae for $E[WT]$ and $E[TID]$ (see (6.10),(6.10')). If for NS large (to satisfy conditions of the law of large numbers) we obtain $E[WT] < approxAWT$, $E[TID] \approx ATID$, then the model is valid and it can be used for any types of distribution of AT and ST.

Finally we note that the same model can be developed by using *a constant increment clock time.* (See Vaduva-1977). The construction of such a

model is usually easier than the construction of the variable increment clock time model.

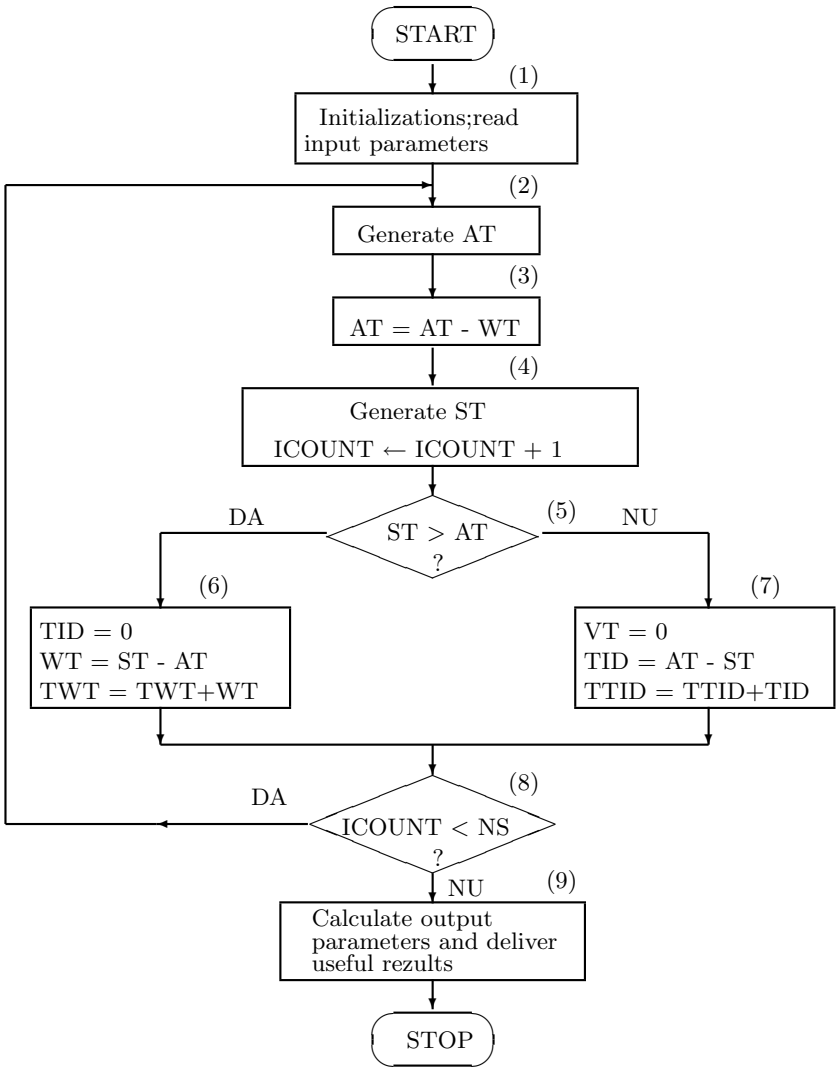

Fig. 6.1. The flowchart of the model ./1./1: (∞ FIFO), with variabile clock time

6.2 Simulation of a queueing system with N parallel stations

Now, we built up the simulation model

$$\bullet/\bullet/N : (\infty; FIFO). \tag{6.13}$$

i.e.the system contains N parallel stations. Such a system can be imagined

in relation with desks of a bank, the desks of a public service, the tanks of a petrol station, etc. We bult up the simulation model to be used via a general purpose language. The variables and parameters used are

AT-arrival time;

$ST(j)$- service time at the station $j, 1 \leq j \leq N$;

WT and TID were already used;

TAT- the time of the last arrival (the clock of arrivals);

$TT(j)$- the time of station j (i.e. the clock time of j);

$TTMIN$- the clock time of the system (the variable increment clock time);it is defined as

$$TTMIN = \min_{1 \leq j \leq N} TT(j).$$

(Discipline $FIFO$ is involved).

L- the current analyzed station, i.e the station for which $TTMIN = TT(L)$;

NS- the number of customers to be served (i.e. an input parameter giving the condition to finish simulation);

$ICOUNT$- integer variable which counts the served customers;

$SST(j)$- sum of service times fulfilled by the station j;

$SWT(j)$- sum of waiting times of customers served by the station j;

$TTID(j)$-total idle time of the station j;

$DIF = TAT - TTMIN$- a working variable.

The initial values of variables are

$$TAT = TT(j) = SST(j) = SW(j) = TTID(j) = 0,\ 1 \leq j \leq N.$$

Apart from NS, input parameters are also the parameters of AT and $ST(j)$. The flowchart of the simulation model ios given in Fig.6.2. We will underline the function of each block in Fig.6.2.

The block (1) reads input parameters and makes initializations. Then, the cycle (2) and the block (3) simulate $N - 1$ arrivals (it is assumed that one customer is already in the system). In a similar way the cycle (5) and the block (6) simulate the service of the first arived customers, while block (4) records the customers served. Block (7) determines the clock time and the station L with this clock. Block (8) records a new customer and block (9) checks the termination of simulation.

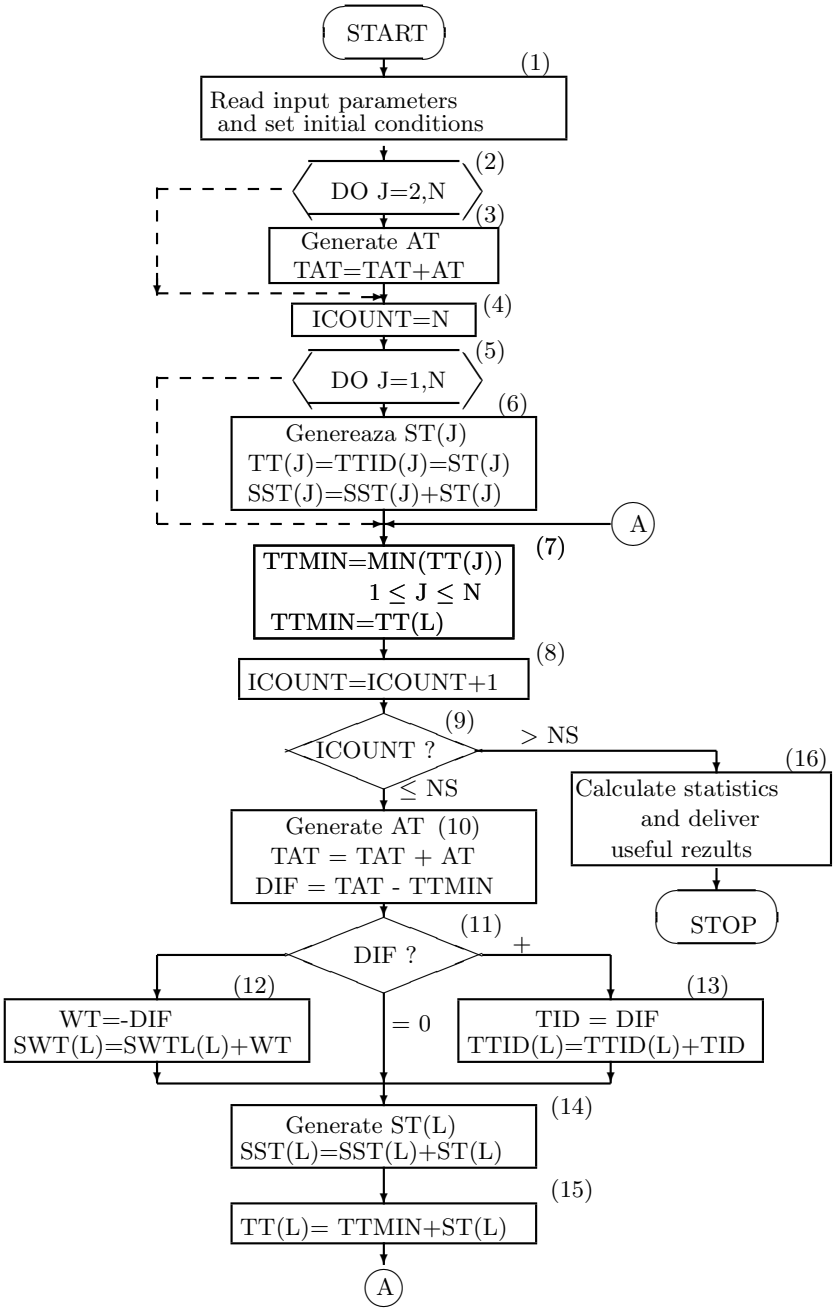

Fig. 6.2. Flow chart for simulation of a queueing system with N parallel stations.

If the simulation continues then block (10) generates AT, updates TAT and calculates DIF. Block (11) tests the sign of DIF; if this is negative then one calculates (in block (12)) WT and $SWT(L)$ is updated, while, on cotrary, when the sign is positive, then one calculates TID and $TTID(L)$ is updated (block (13)). Block (14) simulates a new service time (which was already recorded in block (8)) and the clock time of the station L is updated in block (15). Block (16) calculates final statistics such as

$$AWT = \frac{\sum_{L=1}^{NS} SWT(L)}{NS},\ ATID = \frac{\sum_{L=1}^{NS} TTID(L)}{NS}, \tag{6.14}$$

and delivers useful results.

Validation of the model. In order to validate rhe model (6.13), we present the solution of the queueing model

$$Exp(\lambda)/Exp(\mu)/N : (\infty; FIFO) \tag{6.13'}$$

where ist is assumed that all service stations have the same service distribution $Exp(\mu)$. Therefore, according to Example 6.1 we have in the stationary case

$$\lambda_n = \lambda,\ \forall n \geq 0, \quad \mu_n = \begin{cases} n\mu,\ for\ 0 \leq n \leq N-1 \\ N\mu,\ otherwise \end{cases}. \tag{6.15}$$

Then, denoting $\rho = \frac{\lambda}{\mu}$, one obtain

$$p_n = \begin{cases} \frac{\rho^n}{n!} p_0,\ for\ 1 \leq n \leq N-1, \\ \frac{\rho^n}{(N-1)!N^{n-N}} p_0\ for\ N \leq n \leq \infty \end{cases} \tag{6.15'}$$

and

$$p_0 = \left[\sum_{n=0}^{N-1} \frac{\rho^n}{n!} + \sum_{n=N}^{\infty} \frac{\rho^n}{(N-1)!N^{n-N}}\right]^{-1} = \left[\sum_{n=0}^{N-1} \frac{\rho^n}{n!} + \frac{\rho^N}{(N-1)!}\frac{N}{N-\rho}\right]^{-1}. \tag{6.15"}$$

Some characteristics of this queueing system are as follows.

The average queue length:

$$E[WL] = \sum_{n=N}^{\infty} (n-N)p_n = p_0 \left[\frac{N^N}{N!} \sum_{n=N}^{\infty} n\frac{\rho^n}{N^n} - \frac{N^{N+1}}{N!} \sum_{n=N}^{\infty} \frac{\rho^n}{N^n}\right].$$

Putting $\rho^* = \frac{\rho}{N}$ (i.e. *traffic intensity of the system*) we obtain

$$E[WL] = p_o \left[\frac{N^N}{N!}\rho^* \frac{d}{d\rho}(\sum_{n=N}^{\infty} n\rho*^{n-1}) - \frac{N^{N+1}}{N!}\frac{\rho^{*N+1}}{1-\rho^*}\right] =$$

$$= \frac{N^N}{N!}\frac{\rho^* N + 1}{1-\rho^*} = \frac{\lambda\mu\rho^N}{N(N-1)!}(N\mu - \lambda). \tag{6.16}$$

In order to calculate the average wayting time we note that the average service time of the whole system is $E[S] = N\mu$ and therefore

$$E[WT] = E[WL]E[S] = \frac{\lambda\rho^* N}{N(N-1)(N\mu-\lambda)}. \tag{6.16'}$$

The average number of idle stations is

$$E[NID] = p_0\left[\sum_{n=0}^{N-1}(n-N)\frac{\rho^n}{n!}\right] =$$

$$= p_0\left[\sum_{n=0}^{N-1} n\frac{\rho^n}{n!} - N\sum_{n=0}^{N-1}\frac{\rho^n}{n!}\right].$$

Beeing a finite sum, the $E[NID]$ can be easily computed and therefore

$$E[TID] = E[NID]E[AT] = E[NID]\frac{1}{\lambda}. \tag{6.17}$$

In order to validate the simulation model (6.16') we must compare the theoretical wayting time $E[WT]$ given by (6.16') with the simulated average wayting time AWT given by (6.14) and/or to compare the theoretical average idle time $E[TID]$ with the simulated averge idle time $ATID$. If, for a large number of simulated services NS the theoretcal and simulated values (in the hypotheses of (6.13')) are close each other , then the simulation model is valid.

Note. When the model requires service with priorities, the AT will have a mixture distribition (e.g. mixtexpinential). In this case for each priority, another queue length WL must be introduces and the simulation model will be completed with new commands to handle these quees.

Exercice. An investor intends to built-up a petrol station with N pomps (to sell petrol). He knows that rate of arrival of customers to the petrol station is λ (known) and the service rate of each pomp is μ and the trobability distributions of arrival time and sedrvice time are exponential. The question is the following: how many places for cars comming to be served the investor must preserve arround petrol station, to be able to keep at least 95% of the cars which arrive at the station? Which is the propbability for a car to wait for service?

Hint. Use the formulae (6.15),(6.15') anf for $\alpha = 0.95$ and determine N_α such as

$$\sum_{i=1}^{N_\alpha} p_n \geq \alpha.$$

N_α is the solution. Now the investor can calculate the area of land to be preserved for cars comming to get petrol.

The probability for a car to wait is

$$P(AT > 0) = P(WL > 0) = \sum_{i=N+1}^{\infty} p_n,$$

with p_n given by (6.15),(6.15').

6.3 Simulation of the machine interference problem

The problem is the following. There are N machines which run automatically, and there are M repair stations (e.g repairmen), such as $N > M$.. When a machine breaks down, a free repairman repairs it. The running times of the machines are random and the service times are also random. When all repairmen are busy and a new machine fails down, that machine must be wayting until a repair man becomes free. The engineers call this wayting time- *interference time.* The purpose of the manager of such a system is to ballance costs of wayting times (of the machines) with the idle times (of the repairmen). Therefore such a system is a queueing system of the form

$$\bullet/\bullet/N : (N - M; FIFO) \tag{6.18}$$

with N parallel stations and discipline $FIFO$.

The simulation model uses the following variables and parameters;

N-the number of machines;

M- The number of repairmen;

I- The queue length;

K- The current number of free repairmen, $1 \leq K \leq M$;

NS- The number of repairs (services) to be performed (i.e. this is an input parameter);

RT- The running time (i.e. the arival time);

ST- the service time (duration) of a repair;

WT- The current waiting time;

TID- The idle time of the current busy repairman;

J- An index for the machine, $1 \leq J \leq N$;

$T(J)$- The total time (clock) of the machine $J, 1 \leq J \leq N$;

$SS(J)$- The total repair time of the machine J;

$SR(J)$- The total running time of the machine J;

$SW(J)$- The total interference (waiting) time of the machine J;

$TAT(L)$- The total time of the repairmam L (i.e. the clock tome of the current repairman L,$1 \leq L \leq M$; L denotes the current repairman);

$SID(L)$- The total idle time of the repairman $L, 1 \leq L \leq M$;

$TBS(L)$- The total busy time of the repairman L;

$IX(J)$-an index (code) associated to machine J as follows:

$$IX(j) = \begin{cases} 0, \ if \ the \ machine \ \text{J} \ is \ running; \\ 1, \ if \ the \ machine \ \text{J} \ is \ under \ repair; \\ -1, \ if \ the \ machine \ \text{J} \ is \ wayting; \end{cases}$$

JM- The current machine processed by the simulation program;

KM- The machine processed, having the largest wayting time;

LM- The current repairman dooing a repair;

LA -The first repairman to become free (when all repairmen are busy);

TMI- The clock time opf the simulation;

$TATM$- The total time of a repairman doing a repair (i.e. $TATM = TAT(LM)$);

TMK- The total time of the machine with the largest wayting time (i.e. $TMK = T(KM)$);

$ICOUNT$- a counter (i.e integer variable) which records the repairs (i.e. when $ICOUNT$ becomes NB then the simulation is finished).

Input variables are RT and ST (i.e. their probability distributions are known) and input parameters are N, M, NB; parameters of RT and ST are also input.

The variables $SS(J), SR(J), SW(j), 1 \leq J \leq N$ and $SID(L), TBS(L), 1 \leq L \leq M$ are output variables. All variables and parameters constizute the *agenda*and $IX(j)$ are system states. Initial conditions of the simulation are

$$\begin{gathered} I = ICOUNT0; K = M; \\ SS(J) = SW(J) = o, 1 \leq J \leq N; \\ TAT(L) = TBS(L) = SIDF(L) = 0, 1 \leq L \leq M; \\ IX(j) = 0, \ 1 \leq J \leq N. \end{gathered} \tag{6.19}$$

The flowchart of the simulation model $\bullet/\bullet/M; (N - M : FIFO)$ (of the machine interference problem) is given in Fig.3. The block (1) sets initial conditions and reads input parameters. Then, blocks (2) and (3) (the cycle)

start the simulation by generating running times for all the machines and give initial values for clock times $T(J)$. Block (4) calculates the clock time TMI and determines the machine JM having this clock time. Note that according to $FIFO$, the clock tie is the $MIN[T(J)]$ for the machines running or under repair (i. e. the machines for which $IX(J) \geq 0$).

The block (5) tests the state of the machine JM. If this is running (i.e. $IX(JM) = 0$) then follows block (8) which selects (according to $FIFO$) the repairman LM with the smallest clock. Then, the block (7) tests if this repairman is not free (i.e. $K = 0$) and block (8) makes the machine JM to wait (i.e. $IX(JM) = -1$), increases the queue length I with one unit and the repair man LM (which is not yet free) becomes LA, i.e. it will be the first to do a repair. Otherwise, if in block (7) the repairman LM is availlable, then block (9) puts him to work (generates ST, updates the $TBS(LM)$ and $SID(LM)$), puts the machine JM under repair (i.e. $IX(JM) - 1$), updates the nummber of free repairmen K (i.e. K:=K+1), updates the LM and $TAT(LM)$ and records the repair performed (i.e. puts $ICOUNT := ICOUNT + 1$).
If in the test block (5) one founds the alternative situation (i.e. JM is under repair, $IX(JM) = 1$) this means that the machine was just finished to be repaired and then, block (10) puts it to run (i.e. $IX(JM) = 0$). Then, if there are no machines waiting (i.e in (11) $I = 0$), then the repairman which finished the repais becomes free (i.e. in (12) $K := K+1$). Otherwise, if in the test block (11) it results that there are machines wayting (i.e. $I > 0$), them the block (13) performs the following operations: the machine KM from the machines waiting is selected (it has the clock $TMK = \min_{IX(J)=-1} T(J)$), the wayting time WT of the machine KM is calculated and sum of waiting times of the machine KM is updated (by $SW(KM) := SW(KM) + WT$).

Then, the service time ST for the machine KM is simulated, the state of KM is put "under repair" (i.e. $IX(KM) = 1$), the queue is diminished by one unit (i.i. $I := I-1$) and $SS(KM)$ is upgated. This service is performed by the repairman LA (determined in block (8)) and therefore $TBS(LA)$ is updated by adding ST and clock times of the machine KM and of the repairman LA are also updated becoming $TMI + ST$.

Finaly a new service is recorded in $ICOUNT$. For the machine JM selected in block (4) and just repaired, the block (14) generates its running time RT, updates its sum of running times $SR(JM)$ and also updates its clock time $T(JM)$. Block (15) tests the condition of termination. If there are not NB repairs performed, then the simulation cycle is continued from blocl (4), otherwise the simulation is finished and block (16) displays the

simulation results.

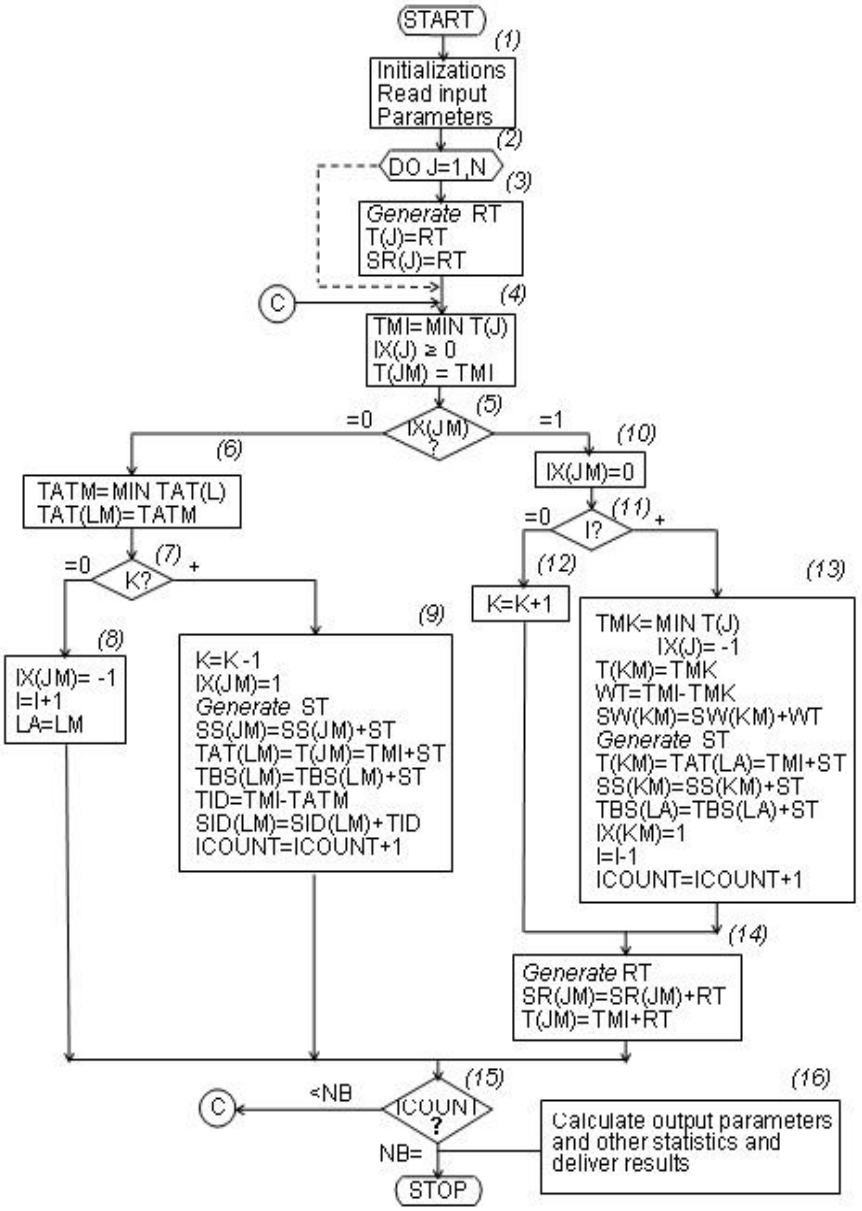

Fig. 6.3 A flowchart for simulation of the machine interference problem.

Note. If the running times of the system have different probabiliti distributions, then this can be modeled by assuming that RT has a mixture distribution with k terms (say). In this case could be also used k priorities

and instead of one queue we can use many queues $I(p), 1 \leq p \leq k$. Then, the machine to be repaired, (in blocks (5), (9),(13)) is selected from the machines with tthe highest priority. The model must be also extended in the case when the repairmen have different probability distributions of repair time, $RT(L), 1 \leq L \leq M$.

Validation of the model. In order to valigate the simulation model (6.18) we use a similar way as in Example 6.1, i. e. we solve first tzhe mathematical queueing model

$$Exp(\lambda)/Exp(\mu)/N - M; (\infty FIFO). \tag{6.20}$$

In this model, the number of customers in the system $N(t)$ is $n, 1 \leq n \leq N$, Because the intensity of RT (exponential case) is λ, then we have

$$\lambda_n = \begin{cases} (N-n)\lambda, \ 0 \leq n \leq N \\ 0, \ otherwise. \end{cases} \tag{6.21}$$

(This is explaned by the fact that when there are n machines down, the intensiry of a new gailure in the systen is $(N-n)\lambda$). In a similar manner, the intensity of a repair pf a machine is μ, therefore wnen are n machines down the intensity of the repair in the system is

$$\mu_n = \begin{cases} n\mu, \ if \ 0 \leq n \leq M-1 \\ M\mu, \ if \ M \leq n \leq N \\ 0 \, otherwise. \end{cases} \tag{6.21'}$$

Using notation $\rho = \frac{\lambda}{\mu}$ the probabilities $p_n = P(N(t) = n)$ are

$$p_n = \begin{cases} \frac{N!}{n!(N-n)!}\rho^n p_o, \ if \ 1 \leq n \leq M \\ \frac{N!}{M!m^{N-n}(N-n)!}\rho^n p_0, \ if \ M+1 \leq n \leq N, \\ 0, \ otherwise. \end{cases} \tag{6.22}$$

As usual, p_0 is determined from the condition $\sum_{n=0}^{N} p_n = 1$ which gives

$$p_0 = \left[1 + \sum_{n=1}^{M} \frac{N!}{n!(N-n)!}\rho^n + \sum_{n=M+1}^{N} \frac{N!}{M!M^{n-M}(N-n)!}\rho^n \right]^{-1}. \tag{6.22'}$$

Formulae (6.22),(6.22') could be simplified, but from computational point of view they coul be simply calculated by the computer just as simple finite sums. Assuming calculated $p_n. \, 0 \leq n \leq N$ one can easily calculate:

- Average wayting (interference) time as

$$E[I] = \sum_{n=M}^{N} (n - M)p_n, \quad E[WT] = E[I]E[ST] = E[I]\frac{1}{\mu}, \qquad (6.23))$$

-Average number of customers in the system

$$E[N(t)] = \sum_{n=0}^{N} np_n,$$

- Average number of busy stations $E[NO]$ is

$$E[NO] = \sum_{n=1}^{M-1} np_n + M \sum_{n=M}^{N}, \qquad (6.23')$$

- Average number of idle stations and average idle time

$$E[NID] = N - E[NO],\ E[TID] = E[RT]E[NID] = \frac{1}{\lambda}E[NID], \quad (6.23")$$

-The average busy time

$$E[BS] = E[ST]E[NO] = \frac{1}{\mu}E[NO]. \qquad (6.23''')$$

We can calculate also

- The interference factor

$$IF = \frac{E[WT]}{E[ST]}, \qquad (6.24)$$

- The efficiency factor of the nachines

$$EF = \frac{E[RT]}{E[RT + WT + ST}, \qquad (6.24')$$

-The efficiency factor of repairmen

$$EM = \frac{E[BS]}{E[BS + TID]}. \qquad (6.24")$$

$E[WT], E[TID]$ are theoretical values. From a simulation run we can calculate the simulated (estimated) values of the described factors as

$$\overline{IF} = \frac{TWT}{TST},\ TWT = \sum_{J=1}^{NB} SW(J),\ TST = \sum_{J=1}^{NB} SS(J),$$

$$\overline{EF} = \frac{TRT}{TRT + TWT + TST},\ TRT = \sum_{J=1}^{NB} SR(J),$$

$$\overline{EM} = \frac{TTBS}{TTBS + TTID},\ TTBS = \sum_{L=1}^{M} TBS(L),\ TTID = \sum_{L=1}^{M} TID(L).$$

Validation of the simulation model can be done by comparison of theoretical efficiency factors with the estimated ones, given by previous formulae. An alternative of validation is to compare theoretical values of $E[WT], E[TID]$ with corresponding estimated values calculated as

$$AWT = \frac{TWT}{NB},\ ATID = \frac{TTID}{NB}.$$

The queueing model (6.20) was studied in the particular case $M = 1$. Thus, in this case, the interference factor IF is in the form

$$IF = \frac{1}{y\rho} + N - 1 - \frac{1}{\rho},\ \rho = \frac{\lambda}{\mu},\ \lambda = \frac{1}{E[RT]},\ \mu = \frac{1}{E[ST]}, \tag{6.25}$$

when RT is exponential and ST is either constant, or exponential. (The y is different in these cases). If $ST = const.$ and AT is $Exp(\lambda)$, then

$$Y = y_A = 1 + \sum_{k=1}^{N-1} C_{N-1}^{k}(e^{\rho} - 1)(e^{2\rho} - 1)...(e^{k\rho} - 1). \tag{6.26}$$

This formula was found by Ashcroft. If ST is $Exp(\mu)$ and AT is $Exp(\lambda)$ then one obtain formula of Palm

$$y = y_P = 1 + \sum_{k=1}^{N-1} (N-1)(N-2)...(N-k)\rho^{k}. \tag{6.27}$$

Pollaczek has noticed that formulae (6.26),(6.27) can be written in the compressed form

$$IF = (1 - C_S^2)IF_A + C_S^2 IF_P, \tag{6.28}$$

where C_S^2 is the coefficient of variance of ST and IF_A,IF_P are interference factors corresponding to Ashcroft and Palm.

In the book (Vaduva-1977) the simulation model (6.20) was validated for $M = 1$ using the formulae (6.25),(6.26),(6.27), $\lambda = 1.0$ and the results are presented in the following table.

A table with validation results of model (6.20)

Case	Serv. Param.	Simul.servs NB	IF-Theoret	IF-Simulated
Ashcroft	$\mu = 5.0$	8,000	1.035578	1.034116
	$\mu = 2.0$	8,000	3.004391	3.00312
Palm	$\mu = 5.0$	5,000	1.424339	1.404621
	$\mu = 2.0$	5,000	3.073394	3.132766

The table shows that there is a small difference between theoretical interference factor IF and simulated interference factor $\overline{IF}$.

Pollaczek formulated the conjecture that in the case $M = 1$ with RT-exponential is valid for any distribution of ST (i.e. IF depends only on $C_S = C_V(ST)$). In the book (Vaduva-1977) this conjecture was tested via simulatioon in the case when ST has an Erlang distribution $ERLANG(\mu, \nu)$. In this case $E[ST] = \frac{\nu}{\mu}$, $C_S^2 = \frac{1}{\nu}$. Test results for $\lambda = 1.0$ and different values of μ and ν are given in the following table

A table with results for testing Pollaczek's conjecture

No.of run	μ	ν	$E[ST]$	C_S^2	NB	IF-Theoret.	IF-Simul.
1	16.0	8	o.5	0.125	6,660	3.013017	3.062517
2	80.0	8	0.10	0.125	6,600	0.402161	0.420361
3	50.0	7	0.14	0.1428	6,900	0.649463	0.678609
4	25.0	8	0.32	0.125	7,000	1.113330	1.160812
5	25.0	5	0.2	0.2	6,600	1.113330	1.161201
6	40.0	4	0.1	0.25	7,000	0.436070	0.407381
7	8.0	4	0.5	0.25	7,000	3.021642	2.994786
8	20.0	4	0.2	0.25	7,000	1.132768	1.093517
9	12.0	4	0.33	0.25	7,000	2.157796	2.156249
10	12,0	3	0.25	0.3333	7,000	1.562721	1.618264
11	30.0	3	0.1	0.3333	8,400	0.458676	0.467350
12	15.0	3	0.2	0.3333	8,400	1.165165	1.150346
13	10.0	2	0.5	0.5	7,000	1.229958	1.161601
14	20.0	2	0.1	0.5	7,000	0.503887	0.520840
15	8.0	2	0.25	0.5	7,000	3.038893	3.090300

From the table it results that the theoretical IF and the simulated $\overline{IF}$ are close each other, therefore the Pollaczek's conjecture is tested. For larger numbe of simulated services NB probably the figures will be almost equal.

This is an example on how we can use a *mathematical experiment* (i.e. stochastic simulation) to test formulae which were not yet proved, mathematically.

7. Introduction to inventory models
7.1 Preliminaries

An *inventory (or a stock)* is any resource which has an economic value, characterized by *inputs* and *outputs.* The output from the stock is determined frequently by a *demand.* (Sometimes the output may contain also expired or damaged elements (goods)). An inventory may consist of various goods (in a shop), food, materials or even water (in a dam or reservoir). A factory, for instance can have an inventory of materials or items used for beeing processed in produstion, or, can have an inventory of processed objects (machines, devices). Output from an inventory of a factory consists in delivery of smaler quantities to satisfy a demand (either for beeing processed or to be sent so final users). In this section we will deal with mathematical models which are intended to do an efficient management (or to administrate) an inventory. An inventory model uses the following elements:

- The time t used to describe dynamic behaviour of the inventory;
- The stoc level $I(t)$ at time t;
- The input rate $a(t)$ in the stock at time t;
- The rate of demand $r(t)$ at time t;
- The output $b(t)$ at time t (when this must be distinguished by $r(t)$);

The demand rate and/or output rate could be random variables. Sometimes (e.g. in the case of a dam), the input rate $a(t)$ (i.e the rate of flow of the water entering the dam) can be also a random variable. These random variables have known probability distributions. In the folowing we deal only with inventories of the type used in a factory or in a shop. The input in a such a stock is called an *order* (or an *order quantity*) and is denoted q; usualy the order q enters the stock at discrete moments of time.

In the management of an inventory are important some costs (expressed in money) as:

- The *holding* cost h (i. e the cost to stock one unit of good per unit of time);

The *penalty* cost or the *shortage* cost d (i. e. cost of a shortage of a unit in the stock, per unit of time);

The *set-up* cost s, of an order quantity. Note the difference between holding cost and shortage cost on one side and the set-up cost. The last one is associated to the whole ordered quantity q.

From the above notations it results the following relation

$$I(t) = I_0 + \int_0^t [a(u) - b(u)]du, \tag{7.1}$$

which describes the evolution of the stock level $I(t)$ in therms of input and output.

Using the introduced costs, we can define an *objective* $E[a(t), b(t), r(t)]$ (e.g. a *cost function* or a *profit*) to be optimised. When some of the functions $a(t), b(t), r(t)$ are random then the objective E is random and, is required that expectation of E to be optimized. From the optimization of E we are mainly interested in defining the optimum order quantity q.

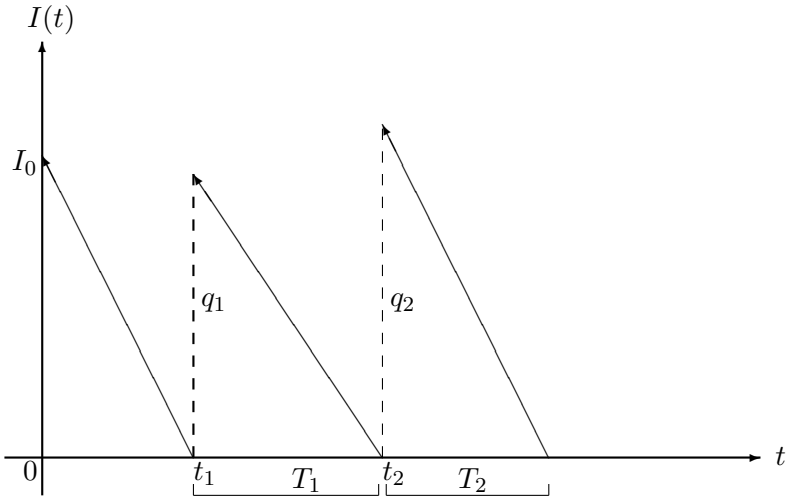

Fig. 7.1. Variation of inventory level $I(t)$

The problems we deal with could be classified in two categories: *stock of manufactured products* (i. e. the stock consists in materials for processing in a factory or in goods to be sold in a shop) or *stock supply* (when the stock contains manufactured goods as machines, devices and so on). Mathematical models for these problems are identical.

Before presenting some mathematical models of inventory theory, we describe first the dynamic of the stock level. This is done in Fig. 7.1. Note that initialy (at time $t = 0$), The stock level is I_0. Than, on the interval $[0, t_1]$ the stoch level is decxreasing becausae it statisfies a demand.

(From the figure, the rate of demand is constant and therefore the stock level decreases linearly). Then, at time t_1 the order quantity q_1 enters the stock. Then, on $[t_1, t_2]$ the stock decreases again and another quantity q_2 enters the stock at time t_2 and so on. The time lengtths $T_i = t_{i+1} - t_i$ are the time intervals when the order quantities enter the stock; they are called *reorder cycles.* (Sometimes the reeorder cycle is constant or random). The dynamics of supply of the inventory (presented in Fig. 7.2) needs some

more elements used by a mathematical model. The order quantity q is set-up when the (decreasing) stock level is P. This is called *reorder level or reorder point.* It is assumed that quantity P existing in the stock, can satisfy the demand until the ordered quantity q enters the stock.

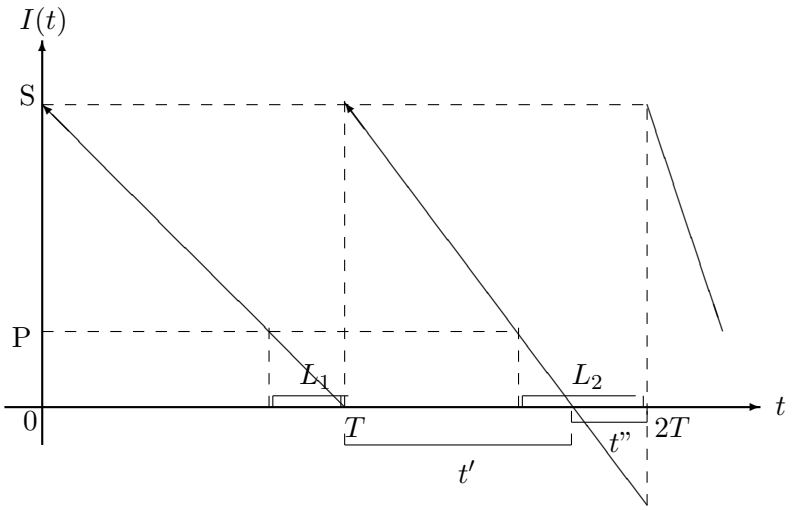

Fig. 7.2. Re-ordering mechanism

This order enters the stock at time T (reorder cycle); the time L elapsed from the moment when q is set-up until it enters the stock is the *lead time.* Some time (on the interval $[T, 2T]$) the stock level $I(t)$ becomes negative (at time $2T$). This means that from the moment of time T there is an interval of time of the length t' when the stock is positive (i.e. on this interval of time the demand can be satisfied). On the other interval of time of the length t", there is a shortage and therefore the demand cannot be satisfied. The figure shows that when the new order q arrives in stock (at time $2T$) the previous unsatisfied demand (on t") must be retrived (i.e. this demand must be satisfied when the order enters the stock). (Usualy, this is the case of stock supply in a factory; in the case of a merchant shop, the unsatisfied demand is lost, in the sense that it is not satisfied when the order enters the stock). In mathematical models the costs h, d, s are known, the demand r and the lead time L are known, while the order quantity q and reorder point P must be determined subject an optimum requirement (e.g. mimimum cost or maximum profit). All mentioned elements defeining optimum supply of a stock are the *optimum policy of the inventory management.* When the demand r and/or lead time L are random the model is *stochastic*, otherwise

it is deterministic. If the time is not explicitedly involved in the model this is a *static* model, while contrary, the model is *dynamic.* The simplest models are those which are deterministic and static. Tho most complex are stochastic and dynamic models.

7.2 Simple one product models

In the following we will present the simplest static, deterministic one product models.

7.2.1 The economic lot size model.

This models assumes that r is constant, the costs h, s are given, there is no lead time (i.e. $L = 0$), there is no shortage (i.e. $I(t) > 0, \forall t$ and $d = 0$). The stock variation is given in Fig. 7.3. The model will determine the order quantity q and the reorder cycle T which minimize a cost function to be defined. Note that because

$$T = \frac{q}{r} \tag{7.2}$$

the only objective is to determine the optimum $\hat{q}$. In order to specify the cost function we note first that the cost per reorder cycle C_T is the sum

$$C_T = C_{h,T} + s, \tag{7.3}$$

where $C_{h,T}$ is the holding cost at the time T. We have

$$C_{h,T} = h \int_0^T I(t)dt = h\frac{qT}{2} = \frac{hq^2}{2}. \tag{7.3'}$$

The cost function cannot be C_T because thies is minimum when $T = 0$, whch does not have any sense. Therefore, we define the cost function as

$$C(q) = \frac{C_t}{T} = \frac{hq^2}{2}\frac{1}{T} + \frac{s}{T} = \frac{hq}{2} + \frac{sr}{q}. \tag{7.3"}$$

From the optimum condition (i.e minimization of $C(q)$) one obtain

$$\hat{q_0} = \sqrt{\frac{2rs}{h}},\ \hat{T_0} = \frac{\hat{q_0}}{r} = \sqrt{\frac{2s}{rh}}, \tag{7.4}$$

and the minimum cost is

$$\hat{C_0} = \sqrt{2rsh}. \tag{7.4'}$$

Note that (from (7.3')) the holding cost per interval $[0, t]$ is

$$C_{h,t} = h\frac{q}{2}t. \tag{7.4"}$$

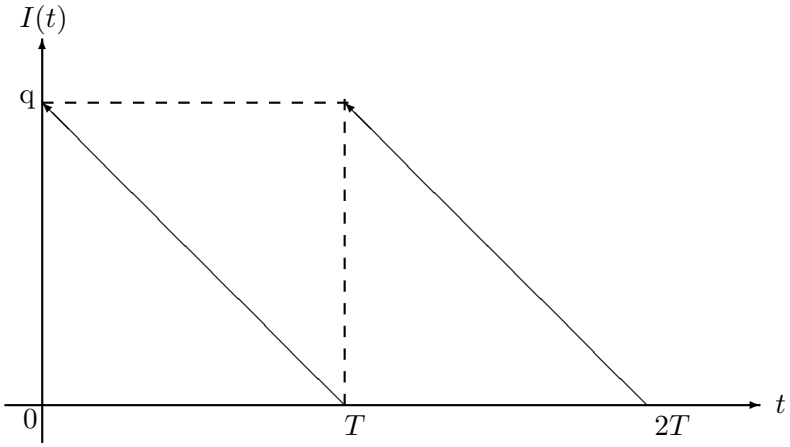

Fig. 7.3. Inventory variation of the economic lot model

7.2.2 The shortage model.

Assume satisfied all hypotheses of the previous model and furthermore there is shortage (cost $d > 0$, is given) and the demand is retrieved (i.e. the unsatisfied demand is satisfied when q enters the stock). The stock variation in this case is shown in Fig.7.4. Note that apart from q, the maximum (optimum) stock level S must be also calculated. From some triangles in the figure one finds

$$t' = \frac{Sr}{,}\ t" = T - t' = T - \frac{S}{r} = \frac{rT - S}{r}.$$

Similar to the previous model one finds that the cost per unit time has three components, namely

$$C(S,T) = \frac{S}{T} + \frac{hS^2}{2rT} + \frac{d(rT-S)^2}{2rT}. \tag{7.5}$$

From the minimum conditions of the cost function

$$\frac{\partial C}{\partial S} = 0,\ \frac{\partial C}{\partial T} = 0,$$

we derive

$$\hat{T_1} = \sqrt{\frac{2s}{rh}}\sqrt{\frac{1}{\rho}},\ \rho = \frac{d}{h+d},$$

$$\hat{S_1} = \sqrt{\frac{2rs}{h}}\sqrt{\rho},\ \hat{q_1} = \hat{T_1}r = \sqrt{\frac{2rs}{h}}\sqrt{\frac{1}{\rho}}, \hat{C_1} = \sqrt{2rsh}\sqrt{\rho}. \tag{7.6}$$

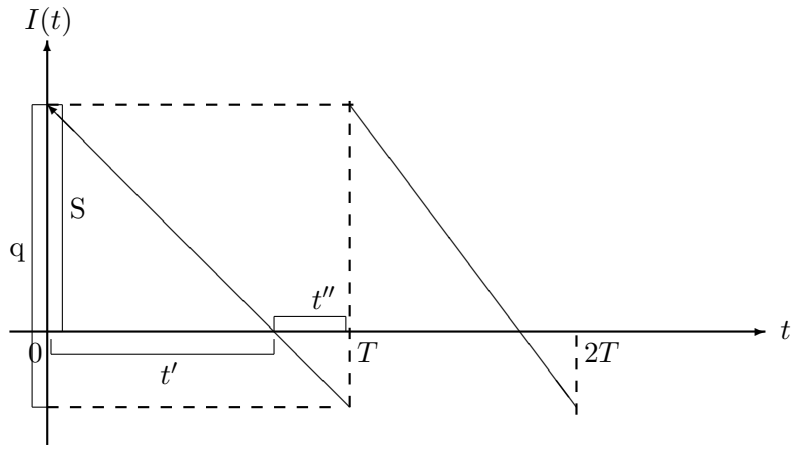

Fig. 7.4. Stock variation for the shortage model

Note that $0 < \rho < 1$ (ρ is called *shortage index*) which gives

$$\hat{C}_1 = \sqrt{\rho}\hat{C}_0 < \hat{C}_0$$

which means that the sortage model is *better* than the economic lot model. But this is not quite true because $\rho = \frac{\hat{S}_1}{\hat{q}_1}$, and if we assume that $\alpha = 1 - \rho$ is given then would result in $gd = \frac{1-\alpha}{\alpha}h$, a relation whch means a dependence between h and d, a reation which does not seem to be valid in practice.

This model will be used for building up a simulation model in section 7.5.

7.3 Multiproduct models.

These models derive from the economic lot model. Assume that in a factory are produced p different types of products. Let us denote for the product i the unit price of row materials c_i, the demand rate is r_i, the setup cost is s_i and assume that the holding cost h_i is a fraction k_i, of c_i (i.e $h_i = k_i c_i$),$0 < k_i < 1, 1 \leq i \leq p$. Therefore the production cost per unit of product is $c_i + \frac{s_i}{q_i}$ and the cost of the delivered production is $r_i(c_i + \frac{s_i}{q_i})$. Hence, the cost function to be minimized is

$$C(q_1, q : 2, ..., q_p) = \sum_{i=1}^{p} [r_i(c_i + \frac{s_i}{q_i}) + k_i \frac{q_i}{2}(c_i + \frac{s_i}{q_i})]. \qquad (7.7)$$

The last formula is in the form

$$C(q_1, q_2, ..., q_p) = K + \sum_{i=1}^{p}(\frac{A_i}{q_i} + \frac{B_i}{2}q_i), \tag{7.8}$$

where

$$K = \sum_{i=1}^{p}(r_i c_i + \frac{k_i}{2}s_i),\ A_i = r_i s_i,\ B_i = k_i c_i.$$

If the quantities are independent then the optimum $q_i's$ are derived as in the economic lot model. In practice the $q_i's$ are connected by some *restrictions* (constraints).

One restriction can be defined by an upper limit of the whole stock V. From the economic lot model it resulta that the average stocked products is $\frac{1}{2}\sum_{i=1}^{p} v_i q_i$ where v_i is the volume *ocupied* by an unit of product i, i.e. the *volume restriction* is

$$\frac{1}{2}\sum_{i=1}^{p} v_i q_i \leq V. \tag{7.8'}$$

(Note that v_i, V could be expressed in currency!).

Another restriction can be defined in terms of *production capacity* (production time is limited!). If total production time T is given and if t_i is the time necessary to produce an unit of type i, then the restriction of production capacity is

$$\sum_{i=1}^{p} \frac{r_i t_i}{q_i} \leq T. \tag{7.8"}$$

If we denote $Q = (q_1, ..., q_n)$ and $G_i(Q) \leq 0$ with

$$G_1(Q) = \frac{1}{2}\sum_{i=1}^{n} q_i v_i - V,\ G_2(Q) = \sum_{i=1}^{n} \frac{t_i r_i}{q_i} - T$$

and $D = \{Q; G_1(q) \leq 0;\ G_2(q) \leq 0\}$; then, the optimization problems could be formulated as follows

$$Min_{Q \in D} C(q_1, q_2, ..., q_n) \tag{7.9}$$

with C given by (7.8). These problems are solved by using the method of Lagrange multipliers. The multipliers λ_1, λ_2 are selected depending of the nature of constraints. If the constraints are equalities then $\lambda_i \in R$, and if the constraints are inequalities, then

$$\lambda_i = \begin{cases} < 0,\ if\ Q \in Fr(D) \\ 0,\ if\ Q \in Int(D) \end{cases}$$

where

$Fr(D) = \{Q; G_1(Q) = 0,\ G_2(q) = 0, \}$, $Int(D) = \{Q; G_1(Q) < 0,\ or,\ G_2(Q) < 0\}$.

The minimization problem (7.9) is equivalent to

$$Min_{Q \in D} L(\Lambda, Q), \Lambda = (\lambda_1, \lambda_2), L(\Lambda, Q) = C(Q) + \lambda_1 G_1(Q) + \lambda_2 G_2(Q). \tag{7.9'}$$

The minimum condition of L in (7.9') i.ie $\frac{\partial L}{\partial q_i} = 0$ gives finally in our case

$$q_i(\lambda_1, \lambda_2) = \sqrt{\frac{2(A_i - \lambda_2 t_i r_i)}{B_i - \lambda_1 u_i}}. \tag{7.9"}$$

Note that $\frac{\partial L}{\partial \lambda_i} = 0$ give $G_i(Q) \leq 0$. Now , by applying a suitable numerical algorithm, we must find $(\lambda_1, \lambda_2) \in (-\infty, 0) \times (-\infty, 0)$ such as $G_1(\Lambda, Q) < 0$, $G_2(\Lambda, Q) < 0$. This can be done also by a *random search* algorithm (see section 5.5 or 8 of this text).

7.4 Stochastic models.

In such models, the demand rate r, the lead time L and the reorder cycle $T = \tau$ (maybe) are positive random variables. The number of demands per reorder cycle $n(\tau)$ is a random variable and the random variable $Y = r_1 + r_2 + ... + r_{n(\tau)}$ is *total demand.* The following proposition is true

Proposition 7. 1 *If τ has the cummulative distribution function (cdf) $G(\tau)$, the demand r has the cdf $F(x)$ and the number of demands on $[0, t]$ is $Poisson(\lambda t)$, then the total demand Y on τ has the cdf*

$$H(x) = \int_0^\infty \sum_{n=0}^{\infty} \frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!} F_n(x) dG(\tau) \tag{7.10}$$

where $F_n(x) = (F * F * ... * F)(x)$ (the $n-$th convolution product) is the cdf of n independent demands $r_i, 1 \leq i \leq n$.

(The proof can be found in Vaduva -1964).

Note that the probability distribution of Y can be also derived by a *direct simulation algorithm,* namely

for $i = 1$ **to** N **do begin** (N is very large!);

1. Simulate a random variate τ with cdf $G(\tau)$;
2. Simulate a Poisson random variate n with distribution $Poisson(\lambda\tau)$;
3. Simulate n independent random variates r_i with cdf $F(x)$;

4. Calculate $Y = \sum_{i=1}^{n} r_i$;

end;

5. Using $Y_j, 1 \leq j \leq N$ generated in the previous cycle, built-up a *histogram.* This is the empirical distribution of total demand Y.

¡smallskip

The distribution of total demand Y can be expressed in terms of characteristic functions. If V is a random variable with cdf $U(x)$ then the corresponding *characteristic function* is

$$\varphi(t) = E[e^{itV}] = \int_{-\infty}^{\infty} e^{itx} dU(x),\ i = \sqrt{-1}, \tag{7.11}$$

where the integral is in Stieltjes sense. (If the cdf U is zero on $(-\infty, 0)$ then the integral is extended only on $(0, \infty)$). If $\Phi(t), \Psi(t), W(t)$ are characteristic functions of $G(x), F(x), H(x)$ respectively, then from (7.10) the following property is derived

$$W(t) = \Phi[i\lambda(1 - \Psi(t)]. \tag{7.10'}$$

The inverse of the characteristic function $W(t)$ is the cdf, i.e.

$$H(x-0) = \lim_{h \to 0} \frac{1}{\pi} \int_0^{\infty} \frac{\sin th}{t} [\Phi(i\lambda(1 - \Psi(t)) + \Phi(i\lambda(1 - \Psi(-t))]dt. \tag{7.10"}$$

If the function $H(t)$ is known, then *one can calculate the order quantity* q of the stock for a given risk $\alpha, 0 < \alpha << 1$. Thus, if the "stock on hand" is I_0 then we must have

$$H(I_0 + q) = P(I_0 + q \geq Y) = 1 - \alpha. \tag{7.12}$$

where $1 - \alpha$ is a large probability.

If y_α is the α-quantile of Y, such as $P(Y > y_\alpha) = \alpha$, then the order quantity is

$$q = y_\alpha - I_0. \tag{7.12'}$$

As the cdf $H(x)$ is difficult to be handled according to previous formulae, the simplest way to determine y_α is to use the histogram of Y determined by the described simulation algorithm.

Exercice. The reorder cycle has an exponential $Exp(\mu)$ distribution and the distribution of a demand X is $Exp(\theta)$. If the number of demands per reorder cycle $n(\tau)$ has a $Poisson(\lambda)$ distribution, give the expression of the c.d.f. of total demand $Y_{n(\tau)} = X_1 + X_2 + ... + X_{n(\tau)}$. Give also the formula of the characteristic function $W(t), t \in R$, of the total demand Y.

Hint. We have

$$G(u)=P(\tau<u)=\begin{cases}0,\ u<0,\\ 1-e^{-\mu u},\ u\geq 0\end{cases};F(x)=P(X<x)=\begin{cases}0,\ x<0\\ 1-e^{-\theta x},\ x\geq 0,\end{cases}$$

and $F^{(n)}(x)=P(Y_n<x)$ is an $Erlang(\theta,n)$ c.d.f., i.e

$$F^{(n)}(x)=\frac{\theta^n}{(n-1)!}\int_0^x u^{n-1}e^{-\theta u}du.$$

Hence

$$H(x)=P(Y_{n(\tau)}<x)=\int_0^\infty\left[\sum_{n=0}^\infty\frac{e^{-\lambda\tau}(\lambda\tau)^n}{n!}\frac{\theta^n}{(n-1)!}\int_0^x u^{n-1}e^{-\theta u}du\right]\mu e^{-\mu\tau}d\tau.$$

In terms of characteristic functions (according to (7.10')) we have

$$\varphi_\tau(t)=\phi(t)=\frac{\mu}{\mu-it};\ \varphi_X(t)=\Psi(t)=\frac{\theta}{\theta-it},i=\sqrt{-1},$$

giving finally

$$W(t)=\varphi_{Y_{n(\tau)}}(t)=\phi[i\lambda(1-\Psi(t))]=\frac{\mu(\theta-it)}{\mu\theta-it(\mu+\lambda\theta t)}.$$

7.4.1 A dynamic stochastic model

A model which describes the most common situations is the folowing. Assume that orders are set-up at regular (equal) intervals (periods) of time $i=,1,2,...$ (i.e days, weeks, months,etc). Assume that the lead time L is a given constant integer ($L>1$).We use the notations

I_i- the stock level at the end of period i;

q_i- the order quantity which enters the stock during the period i;

r_i- the demand per period i; r_i are random variables, identically distributed and independent, with a known probability distribution;

S_i- is the *generalized stock* at the beginning of the period i; it consists of the stock on hand I_{i-1} at the beginning of the period i plus, all quantities q set up until then, which did not arrived yet in the stock, hence

$$S_i=I_{i-1}+q_i+...+q_{i+L-1},\tag{7.13}$$

(Note that q_{i+L} is the ordered quantity at time i, which will arrive in stock at time $i+L$).

From the above it results that the demand per $L+1$ periods of time is

$$R_i = r_i + r_{i+1} + ... + r_{i+L},$$

it has a known distribution and R_i, R_{i+1} are independent. Denote $F(r)$ the cdf of R anf f(r) the pdf of R. If the cost h, d are known and set-up cost is omitted (i.e. $s = 0$), the problem is to determine the optimum order $\hat{q}_{i+L}$ which is set-up at the end of period i,and, which will enter the stock in the period $i + L$. Let us consider the quantity

$$Q = S_i + q_{i+L}.$$

From the relation (7.13) and

$$I_i = I_{i-1} + q_i - r_i$$

$$S_i = I_i + r_i + q_{i+1} + ... + q_{i+L-1} =$$
$$= I_{i+1} + r_i + r_{i+1} + ... + q_{i+L-1} =$$
$$.....$$
$$= I_{i+L} + r_i + ... + r_{i+L-1}$$

we obtain

$$Q = S_i + q_{i+L} = I_{i+L} + R.$$
$$I_{i+L} = Q - R. \tag{7.14}$$

Denoting

$$I^+_{i+L} = \begin{cases} I_{i+L},\ if\ I_{i+L} > 0, \\ o, otherwise \end{cases};\ I^-_{i+L} = \begin{cases} o,\ if\ I_{i+L} < 0 \\ I_{i+L},\ otherwise \end{cases}$$

it results from (7.14) that we have to minimize the average cost function

$$E[C(Q)] = hE[(Q-R)^+] - dE[(Q-R)^-],$$

$$E[C] = h\int_0^Q (Q-R)f(R)dR + d\int_Q^\infty (R-Q)f(R)dR. \tag{7.13'}$$

(For a variable I, I^+ is called *surplus* and I^- is called *deficit*). If we find the optimum $\hat{Q}$ then we determine easily the optimum $\hat{q}_{i+L} = \hat{Q} - S_i$. (Note that S_i is known according to (7.13)).

Our problem is now to solve the *min* problem (7.13') which gives $\frac{dE(c)}{dQ} = 0$.. Let us denote $A(Q,R) = (Q-R)^+ > 0$, $B(Q,R) = (Q-R)^- < 0$. Then (7.13') becomes

$$C = hA(Q,R) - dB(Q,R),\ E[C] = min,\ \frac{d}{dQ}E[A(Q,R) + B(Q,R)] = \alpha, \tag{7.15}$$

which is a *problem of linear control* having the objective (7.13') and a (*differential*) constraint specified by the second formula (7.15). Because in our case we have $E[A+B] = Q - E[R]$ (from (7.14)), we have $\alpha = 1$. From the condition of minimum and from (7.15) we have

$$h\frac{dE[A]}{dQ} - d(-\frac{dE[A]}{dQ} + \alpha) = 0,$$

which gives

$$\frac{dE[A]}{dQ} = \frac{\alpha d}{h+d},\ or\ F(Q) = \frac{d}{h+d}. \tag{7.16}$$

The solution in Q of the second relation (7.16) is

$$\hat{Q} = F^{-1}(\frac{d}{h+d}$$

and the model is solved, i.e $\hat{q}_{i+L} = \hat{Q} - S_i$.

Ecercice. The demand r per (unit) period of time has an exponential $Exp(\lambda)$ distribution. In the hypotheses of the model 7.4.1, determine the optimum order $\hat{q}_{i+L}$ when the initial stock level I_0 is known and the previous ordered quantities $q_i, q_{i+1}, ..., q_{i+L-1}$ are known. The lead time L is a given positive integer and the costs h, d are also known.

Hint. The demand per $L+1$ control periods of time is $Erlang(\lambda, L+1)$ distributed, i.e.its p.d.f. is

$$f(R) = \frac{\lambda^{L+1}}{L!}R^L e^{-\lambda R},\ r > 0.$$

Hence

$$F(Q) = \frac{\lambda^{L+1}}{L!}\int_0^Q R^L e^{-\lambda R}dR.$$

The optimum $\hat{Q}$ is the solution of equation

$$F(\hat{Q}) = \frac{\lambda^{L+1}}{L!}\int_0^{\hat{Q}} R^L e^{-\lambda R}dR = \frac{d}{h+d}$$

and $\hat{q}_{i+L} = \hat{Q} - S_i$, $S_i = I_0 + r_i + r_{i+1} + ... + r_{i+l-1}$. If we apply Proposition 2.1 bellow, then $\hat{q}_{i+L} = r_{i-1}$, where r_{i-1} is *simulated* from $Exp(\lambda)$ distribution.

For a cdf $F(x)$ which cannot be inversed analytically, (as in the previous Ecercice), its inverse $F^{-1}(t)$ (which exists for every $t > 0$) can be determined by an obvious numerical procedure. One can prove the following proposition

Proposition 7. 2 *In the hypotheses of this model we have*

$$q_{i+L} = r_{i-1}. \tag{7.17}$$

The last formula is called *the feed back rule* and it is proved in (Vaduva-1974).It states that the best order to be *set-up now* at period i is equal with the demand r_{i-1} from the previous period $i - 1$.

7.4.2 A model for reserve stock

Assume that in a manufacturing company there are two automatic machines M_1, M_2 (See Fig. 7.5)

$$\rightarrow \quad M_1 \quad \stackrel{S}{\rightarrow} \quad M_2 \quad \rightarrow$$

Fig.7.5. Two machines M_1, M_2 in series

which are running in series, i.e. in M_1 enters material which is partially processed by this machine; then the output of M_1 enters M_2 to get further processing. The machine M_1 has a random failure time τ with pdf $g(\tau)$ and the machine M_1 has the average running time $E[\theta] = \mu$. When the machine M_1 breaks-down, then the machine M_2 is *suffering* due to shortage output of M_1. The problem is to determine an *optimum reserve* S (a safety inventory) with material partially processed (by M_1 or by other parallel like machine), to ensure a continuous running of the system of serial machines M_1, M_2. Apart from the known elements μ and $g(\tau)$ we assume that the holding cost h of the stock S and the shortage cost d (related to the machine M_2) are known, the demand rate r (or the processing rate of material) of the machine M_2 is also known and we must determine the optimum reserve $\hat{S}$. We assume also that S remains constant such as during a failure of M_1 the machine M_2 gets supply from S and the reserve S is quickly filled in by another parallel machine. Another assumption is that the failure duration of M_1 plus the time to refill S is smaller than $E[\tau]$. From this, results that the holding cost per unit time for the whole stock S is $C_h = hS$ (and not $h\frac{S}{2}$ as usual, even if this does not imply any difficulties!). Note that for the machine M_2 there

is an *idle time* t caused by the failure time τ of M_1. (During this failure time the machine M_2 is supplied by S). Hence we nhave

$$t = \begin{cases} 0, \ if \ S \geq r\tau \\ \tau - \frac{S}{r}, \ if \ S < r\tau \end{cases}$$

and the average shortage cost per failure is

$$D = d \int_{\frac{S}{r}}^{\infty} (\tau - \frac{S}{r}) g(\tau) d\tau.$$

Because there are in the average $\frac{1}{\mu}$ failures per unit time, this means that the shortage cost per unit time is

$$C_d = \frac{d}{\mu} \int_{\frac{S}{r}}^{\infty} (\tau - \frac{S}{r}) g(\tau) d\tau.$$

Therefore the cost function of the model (i.e. the average cost) to be minimized is

$$C(s) = hS + \frac{d}{\mu} \int_{\frac{S}{r}}^{\infty} (\tau - \frac{S}{r}) g(\tau) d\tau.$$

From the minimum condition we have

$$\frac{dC}{dS} = h - \frac{d}{r\mu}[1 - G(\frac{S}{r})),$$

where $G(x)$ is cdf of τ, and hence optimum $\hat{S}$ is the solution of ecuation

$$G(\frac{S}{r}) = 1 - \frac{r\mu h}{d}.$$

Note that

$$\hat{S} = \begin{cases} 0, \ if \ r\mu h \leq d, \\ rG^{-1}(1 - \frac{r\mu h}{d}), \ if \ r\mu h > d. \end{cases} \tag{7.18}$$

Note. The last formula says that when $r\mu h \leq d$ then is not necessary a reserve $\hat{S}$. This model gives a rule of managing a production line with machines running in series.

Example. If τ has an exponential distribution $Exp(\lambda)$ (as usual!), then $G(x) = 1 - e^{-\lambda x}, x > 0$, therefore

$$\hat{S} = -\frac{r}{\lambda} \log \frac{r\mu h}{d}, , \ r\mu h > d. \tag{7.18'}$$

If the equation (7.18) cannot be solved analytically (as in (7.18')) then one can built up a numerical solution.

7.5 A simulation model.

Here we will present a simulation model using the fixed increment clock time (increment $c = 1$),which is based on the shortage model from section 7.1. The identifiers used in the model are:

H= the holding cost;D= the shortage cost; S= the set-up cost;

CH= total holding cost over the period of time of simulation;

CD= total shortage cost for the period of simulation;

CS= totsl set-up cost for the peeriod of simulation;

TC= total cost (i.e. $TC = TH + TD + TS$);

T= the moment of time when an order enters the stock;

R= demand rate (demand per unit of time); a random variable of a known distribution;

VI= the (current) stock level;

Q= the optimum order quantity;

P= the reorder stock level (reorder point);

L= the lead time; a discrete random variable ($L = 1, 2, ...$) which has a known probability distribution;

$CLOCK$= the clock time (an integer);

BI= the initial stock level;

TT= the period of time for simulation; (a large integer!).

Input random variables are R, L and input parameters are H, D, S, P, BI, TT. The reorder point P can be determined by the following formula

$$Prob(R_{(L)} > P) = 1 - \alpha, \tag{7.19}$$

whewre α is the risk of not satisfying the demand $R_{(L)} = R_1 + ... + R_L$ during the lead time L. If R is normal $N(m, \sigma)$ distributed and denote $l = E[L]$, then $R(L)$ is normal $N(lm, \sqrt{l}\sigma$. If we consider the quantile z_α defined as

$$\int_{-z_\alpha}^{z_\alpha} e^{-\frac{t^2}{2}} dt = 1 - \alpha$$

then the reorder pouint is

$$P = lm + z_\alpha \sqrt{l}\sigma. \tag{7.20}$$

If R is distributed as $Exp(\lambda)$, then $R_{(L)}$ is distributed as $Gamma(0, \lambda, l)$ (i.e. Erlang) and therefore thew reorder point P satisfies relation

$$Prob(R_{(L)} > P) = \int_P^\infty \frac{1}{\lambda\Gamma(l)} t^{l-1} e^{-\frac{t}{\lambda}} dt = 1 - \alpha \tag{7.20'}$$

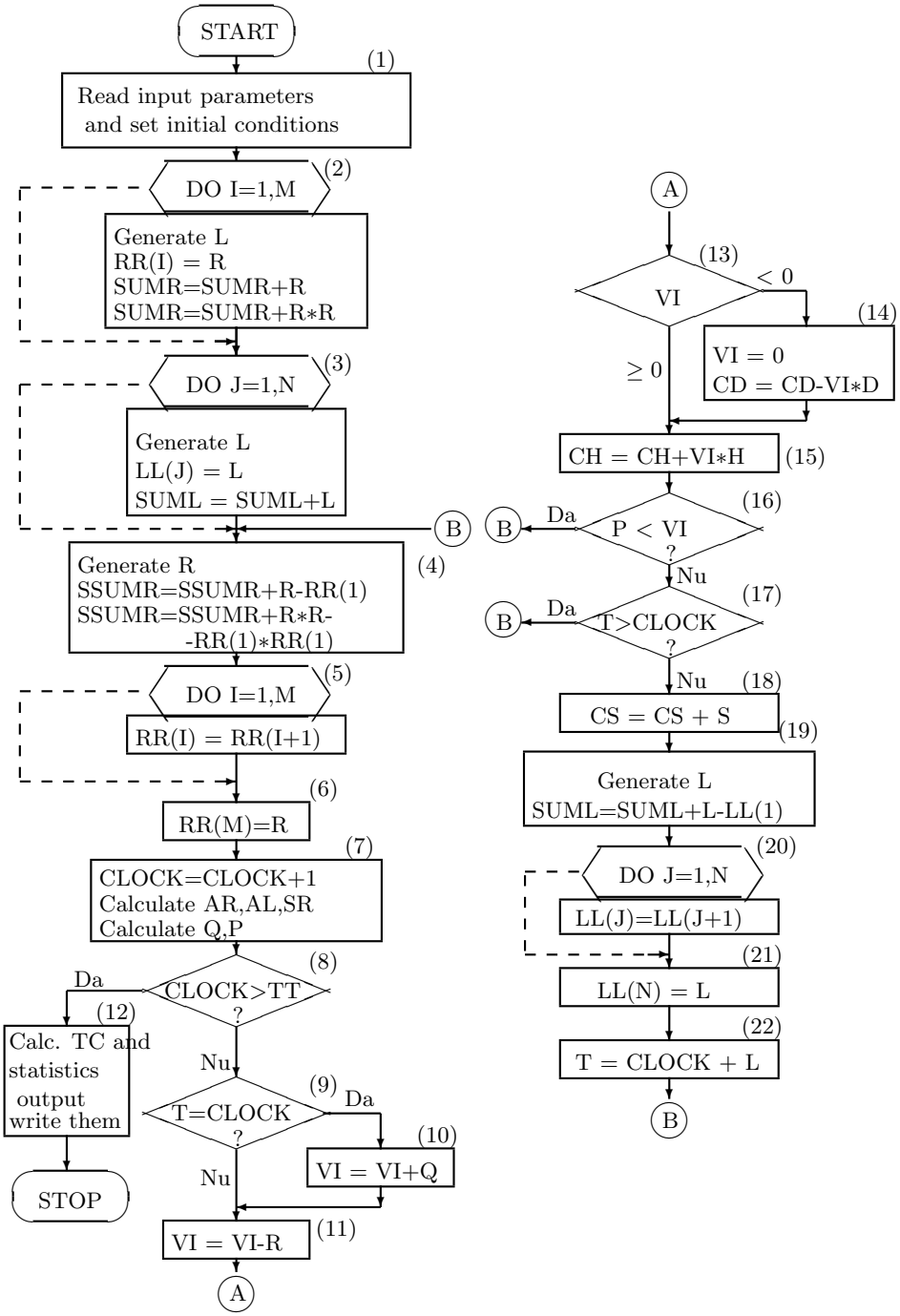

Fig. 7.6. Flowchart for an inventory simulation model

i.e $P = p_\alpha =$ the upper α quanztile of $R_{(L)}$. The simulation model calculates the input parameter P the quantiles z_α or p_α and formulae (7.20),(7.20') (depending on distribution of R). In order to calculate the order Q one uses the formulare from the shortage model, with some further improvement. One uses instead of R its estimate using the moving average based on M terms and instead of $l = E[L]$ its estimate as moving average based on N terms namely

$$E[R_{(L)}] \approx AR = \frac{\sum_{i=1}^{M} R_i}{M},\ l = E[L] \approx AL = \frac{\sum_{j=1}^{N} L_j}{N}.$$

(It is assumed that R is $N(m(t), \sigma(t))$ or $Exp(\lambda(t)$ and L has the mean $E[L] = l(t))$. The order Q is therefore

$$Q = \sqrt{\frac{2.AR.S}{H}}\sqrt{\frac{H+D}{D}}. \tag{7.21}$$

The flowchart is given in Fig.7.5. The initial conditions are

$$TC = CS = CD = CH = 0,\ VI = BI,\ T = CLOCK = 0.$$

All blocks in the flowchart are explanatory. Note thar tme model refers to only one type of material. Using the ideas from this section, it can be improved for different distributions of R and L or it can be modified for multiproduct inventories.

8 Optzimization using random search

8.1 Introduction

In this section we present methods of optimization based on *random search.* The problem is the following,

$$\max_{x \in D} f(x), \tag{8.1}$$

where $D \subset R^k$ is a k-dimensional set. For solving a *min* problem, the function $f(x)$ is changed in $-f(x)$ or in another problem (e.g. $f(x) \mapsto \frac{1}{f(x)}$ if $f(x) \neq 0$). There is a large "classical" literature based on properties of $f(x)$ or of D. Here we are interested in determining the point $x* \in D$ such as $\max_{x \in D} f(x) = f(x^*) = f^*$, i.e. x^* is a *global* maximum point. The global maximum point is selected from several local maximums. Such a local maximum belongs to a *capture set* i.e. a set which contains only the maximum

point x^*. (In fact the term *capture* comes from the recurent methods for determining x^* and f^*).

The idea of *random search* is the following. Generate a large number N of random points $X_1, X_2, ..., X_N$ in D. (When D is bounded, i.e. $\sup_{x,y\in D} ||x-y|| < \infty$, these points could be uniformly distribited on D). Then calculate $f(X_i), 1 \leq i \leq n.$ and take $f^*_{(N)} = f(X^*_{(N)}) = \max_{1\leq i\leq N} f(X_i)$. A theorem of Gnedenko (1943) says that in some conditions (i.e. f is a continuous function), we have $\lim_{N\to\infty} f^*_{(N)} = f^*$, $\lim_{N\to\infty} X^*_{(N)} = x^*$. When the points $X_i, 1 \leq i \leq N$ are uniform on D, then we call this *Crude Monte Carlo* (CMC). If the optimum solution $x^* \in T^* \subset D$, (T^* is a capture set containing the solution), then, it is known that for a given risk $\epsilon, 0 < \epsilon < 1$ there is a p such as

$$P(X_N \in T^*; ||X_N - x^*|| \geq \epsilon) = p, \tag{8.2}$$

it is necessary to use

$$N > [\frac{\log \epsilon}{\log(1-p)} + 1] = N^*, \tag{8.2'}$$

previous probability beeing calculated for the assumed distribution of X. As p is not known, one can use $p > 1 - \epsilon$. The CMC algorithm, given by Anderssen and Bloomfeld (1975), for approximating x^* is

1. Input ϵ, p and calculate N^*; Take $Z_0 = -A$ (where $A = MAX$ is a large number) and take Y_0 a point in D;

for $i = 1$ **to** N^* **do begin**

Generate X uniform distributed on D, and calculate $f = f(X)$;

Calculate $Z_1 = max(Z_0, f)$ and

$$Y_1 = \begin{cases} Y_0, \, if \; f < Z_0 \\ X \; if \; f \geq Z_0; \end{cases}$$

Take $Z_0 = Z_1$, $Y_0 = Y_1$,

end;

3 Take $f^* = f$, $x^* = Y_1$.

Note that the algorithm works in the only assumption $|f(x)| \leq M < \infty, \forall x \in d.$

8.2 Importance Monte Carlo for Optimization

In the CMC algorithm, all poionts $X \in D$ are equaly likely to be taken at random. It should be better if we use a sampling method for selecting with higher probability those points in D which are close to x^*. (See Vaduva and Spircu-1981). This can de done if we use a pdf on D which can perform like this. If the function $f(x)$ has a maximum in x^*, then this must be a *mode* of this distribution.

Therefore, we assume that $f(x)$ has the following properties:

a. $f(x) > 0$, $\forall x \in D$;

b. $f(x)$ is a continuous function (i.e. is an integrable function);

c. $f(x)$ is bounded on D, i.e $|f(x)| \leq M$ (this is true if D is a compact set). If a. is not true, then we can select $P, 0 < P < \infty$ and change $f(x)$ with $f(x) + P$. which has the same maximum point.

Now, we can choose a pdf in the form

$$g(x) = \frac{f(x)}{\int_D f(u)du},\ x \in D,\ I = \int_D f(u)du.$$

A random vector X having pdf $g(x)$ can be now simulated using the *rejection enveloping procedure,* with the enveloping pdf density $h(x)$ uniform on D,

$$h(x) = \begin{cases} \frac{1}{H},\ if\ x \in D, \\ 0,\ otherrwise, \end{cases} \qquad H = mesD.$$

If denote $K = \sup_{x \in D} g(x)$, then we find $\frac{g(x)}{h(x)} \leq \alpha = KH = \frac{MH}{I}$, and therefore the algorithm for simulating X is the following

1. Input M, H, I;
2. repeat
Generate Y uniform distributed on D;
Generate U uniform on $(0, 1)$;
until $U \leq \frac{If(Y)}{M}$;
3. Take $X = Y$.

The method based on simulating X with cdf $g(x)$ is called *Importance Monte Carlo* method (IMC). The IMC algorithm is similar to CMC algorithm where X is simulated with pdf $g(x)$.

Convergence of IMC. Now we talk about the convergence of the IMC method of optimization. For a sample of size N, $X_1, X_2, ..., X_N$, $X_i \in D$, from the pdf $g(x)$ (whose cdf is $G(x)$), let us denote $f_i = f(X_i)$, where f is the function to be maximized. Then, the estimate of $f^* = \max_{x \in D} f(x) = f(x^*)$ is $f_N^* = \max_{1 \leq i \leq N} f_i$. Two questions arise now:

1. Does the convergence of f_N^* to f^* depend on the cdf $G(x)$?

2. Assumming that $f_N^* \to f^*$, which is the influence of $G(x)$ on the *speed* of convergence.

The answer to question 1. is given by the following proposition of Gnedenko (1943).

Proposition 8. 1 *If for cdf $G(x)$, any neighbourhood $V(x^*)$ of x^* has a positive probability (i.e $P(X \in V(x^*) > 0$) then $f_N^* \to f^*$ when $N \to \infty$, almost sure.*

From this proposition it results that the cdf $G(x)$ does not influence qualitatively the convergence of f_N^* to f^*.

To the question 2.,the answer is that $G(x)$ does have a big influence to the convergence. This fact was proved by Rubinstein and Weissman (1974) when $G(x)$ is a uniform cdf. In the following we will show that for a given sample size N the IMC algorithm is faster than CMC algorithm, when the function $f(x)$ to be optimized, is continuous. To formulate the proposition which specifies this, we introduce first notations:

Y is the uniform distributed vector on D, and cdf of Y is $G_1(x)$;

X has the cdf $G_2(x)$;

If for the samples of the same size N, $Y_1, Y_2, ..., Y_N$ from $G_1(x)$ and $X_1, X_2, ..., X_n$ from $G_2(x)$ we denote $f_i^1 = f(Y_i)$, $f_i^2 = f(X_i)$ and $f_N^{1*} = \max\limits_{1 \le i \le N} f_i^1$, $f_N^{2*} = \max\limits_{1 \le i \le N} f_i^2$ and note that cdf of f_N^{1*} is $H_1(x) = (G_1(x))^N$ and cdf of f_N^{2*} is $H_2(x) = (G_2(x))^N$, then the following proposition holds

Proposition 8. 2 *Let $\delta > 0$ be a positive number and consider the interval $(f^* - \delta, f^*)$ in the neighbourhood of f^*. Then, $\forall y \in (f^* - \delta, f^*)$ we have*

$$P(f_N^{1*} \in (y, f^*)) < P(f_N^{2*} \in (y, f^*)).$$

Therefore, f_N^{2*} enters faster in the interval (y, f^*) than f_N^{1*}, i.e. IMC procedure converges faster to x^* than CMC procedure. Note that the presented random search methods assume that the constraint domain D is bounded. However, the IMC method can be used also if the domain D is not bounded. In this case, the cdf $G(x), x \in D$ must be properly selected as in the following example.

Example. Consider the following problem

$$\max_{x \in d} f(x),\ x \in R^2, D = (0, \infty) \times (0, \infty),$$

where $f(x)$ is a continuous function. In this case we select the *importance* pdf $g(x,y)$ in the form

$$g(x,y) = \lambda\mu \begin{cases} e^{-\lambda x-\mu y}, \ if \ x>0, y>0, \\ o, \ otherwise, \end{cases} \lambda, \mu > 0.$$

If it is apriory known that the maximum point (x^*, y^*) of $f(x,y)$ is far from the origin $(0,0)$, then we must choose $\lambda < 1, \mu < 1$. A good choice would be also $\lambda = \mu = 1$.

9. Simulation of some discrete stochastic processes

In this section we deal with simulation of some discrete stochastic processes, suvh as Bivariate Uniform Poisson Process (BUPP) and Bivariate Uniform Binomial Process (BUBP) and villustrate their application to a healthcare problem.

9.1. Simulation of Poisson Processes and their use to analyze scan statistics[2]

9.1.1 Introduction

Scan statistics studies clusters of random points taken in an interval I. If $I = [a,b]$, $-\infty < a < b < \infty$, we deal with the *one-dimensional scan statistics* and, if $I = [a_1,b_1] \times ... \times [a_n,b_n]$, $-\infty < a_i < b_i < \infty$, $1 \leq i \leq n$, we have *multivariate scan statistics.* In scan statistics the main problem is to study clusters of points which describe *unusual* situations, namely to see if *it is natural* to find *large* or *small* clusters with a large or a small probability.

The aim of this subsection is to study a *bivariate scan statistic* and to approximate (by simulation of a BUPP) the *critical value* of a statistical test rtelated to the *continuous* scan.

Definition 9. 1 *Let $X_1, X_2, ..., X_N$ be random points in the interval $[0,T]$. Denote S_w the maximum number of points which are found in an interval of length w, $w < T$, when such an interval scans $[0,T]$. The small interval of length w is called scan window and the random variable S_w is called one-dimensional scan statistic.*

When $X_1, X_2, ..., X_N$ are integer valued random variables, then S_w is the discrete one-dimensional scan statistic and when $X_1, X_2, ..., X_N$ are a trajectory of N points of a Poisson process $\{X_t, t \geq 0\}$, then S_w is the one-dimensional continuous scan statistic (sere Alm-1997,Glaz,Naus, Wallenstein-2001).

An alternative to S_w is the random variable W_k defined as the minimum length of an interval in $[0,t]$ which contains k points. Note that we have

$$P(W_k \geq w) = P(S_w \leq k). \tag{9.1}$$

Let us denote this probability with:

$$P(S_w \leq k) = P(k, N, w, T).$$

We will focus in this subsection on a two-dimensional continuous scan statistic:

Definition 9. 2 *Let $I = [0,L] \times [0,T]$ be a two-dimensional interval and $u, v > 0$ two positive numbers such as $0 < u < L < \infty$, $0 < v < T < \infty$. (The numbers u, v define a two-dimensional scan window with dimensions u and v). Assume that in the interval I there are $\{X_1, X_2, ..., X_N\}$ which are a trajectory of N points of a bivariate Poisson process $\{X_t, t \geq 0\}$ with intensity λ. Denote $\nu_{t,s} = \nu_{t,s}(u,v) =$ the number of points which fall in the scanning window $[t, t+u] \times [s, s+v]$. Then the bivariate scan statistic is*

$$S = S((u,v), N, L, T) = \max_{0 \leq t \leq L-u,\ 0 \leq s \leq T-v} \nu_{t,s}. \tag{9.2}$$

The probability of interest is now:

$$P(S((u,v), N, L, T) \geq k) = P((u,v), N, L, T, k). \tag{9.3}$$

The probability distribution (9.3) is hard to calculate. Therefore a simulation procedure is the simplest way to estimate it. In Haiman and Preda-2002 is introduced a method for estimating the probability distribution (9.3) using the simulation of conditional scan statistic and the relationship between scan statistic and conditional scan statistics.

We estimate the probability distribution (9.3) using the simulation of scan statistic and the main steps of the algorithm that we use are the following:

Algorithm SIMSCAN

Input T, N, w, m, λ;

1. For $j = 1$ to m do

begin

generate $X_1, ..., X_N$;

Determine S_w, take $K_j = S_w$;

end;

(*In the Section 3 we describe the implementation of the algorithm SIMSCAN which determines S_w, denoted there by n_w*).

2. Determine the empirical distribution of the sample $K_1, ..., K_m$ as follows:

   2.1 Determine the order statistics $K_{(1)} < K_{(2)} < ... < K_{(r)}$, $r < m$;

   2.2 Determine the frequencies $f_i, 1 \leq i \leq r$, f_i = number of sampling values $K's$ equal to $K_{(i)}$, $1 \leq i \leq r$, $\sum_{i=1}^{r} f_i = m$;

   2.3 Determine the relative frequencies (i.e. sampling probabilities) $\pi_i = \frac{f_i}{m}$.
   (*In fact, step 2 builds-up a frequency distribution, i.e. a histogram of the scan statistics*).

If m is large enough, then the sampling distribution converges to (9.3) (according to the consistency property of the estimates π_i).

9.1.2 Algorithms for the simulation of a multivariate Poisson process

For the estimation of the probability distribution of the two-dimensional continuous scan statistic we need a simulation method for the bivariate Poisson process.

Definition 9. 3 *(see Devroye-1986) A process consisting of randomly occuring points in the plane is said to constitute a two-dimensional Poisson process having rate λ, $\lambda > 0$, if*

1. *The number of points occuring in any given region of area A is Poisson distributed with mean λA.*

2. *The number of points occuring in disjoint regions are independent.*

For $m = 1$ (i.e. $X's$ are real numbers), it is well known that the *spacings* between random points of the Poisson process $Poisson(\lambda t)$, $t \in [0, \infty)$, have an exponential distribution of the parameter λ. This gives the following algorithm for simulating a *trajectory* of k points of the one dimensional homogenuous Poisson process of the intensity λ:

Algorithm SIMPO1

Input λ, k (preparatory step);

1. $i = 0$, $T = 0$;

2. repeat

   Generate $E \mapsto Exp(1)$;

   $i := i + 1$, $T_i := T_i + \frac{E}{\lambda}$;

   until $i = k$.

The algorithm produces the trajectory $T_1, T_2, ..., T_k$ of the univariate $Poisson(\lambda)$ process on $[0, \infty)$. Simulation of $E \mapsto Exp(1)$ can be done by *the inverse* method or by the *rejection* method (see Devroye-1986, Fishman-1996, Vaduva-1977). From this algorithm it results immediately an algorithm for simulating a *bivariate* Poisson process on $A = [0, t] \times [0, 1]$, with the rate λ:

Algorithm SIMPOT01

1. Generate $T_1, T_2, ..., T_k$ a Poisson trajectory on $[0, t]$;

2. Generate $U_1, U_2, ..., U_k$ uniform and independent random variates on $[0, 1]$.

The points $(U_1, T_1), (U_2, T_2), ..., (U_k, T_k)$ determine an uniform Poisson process on A.

Note that k is an integer sampling value of the Poisson random variable with parameter λt.

Similar ideas could be used to built up an algorithm for simulating an uniform Poisson process of the intensity λ on the $n-$dimensional interval $\mathbf{I} = [0, T_1] \times ... \times [0, T_n]$. If we denote $V_0 = \prod_{i=2}^{n} T_i$, the *volume* of the $(n-1)-$ dimensional interval $\mathbf{I_1} = [0, T_2] \times ... \times [0, T_n]$, then the following algorithm simulates the required Poisson process.

Algorithm SIMPOMD

1. Simulate $0 < X_{11} < X_{12} < ... < X_{1k}$ an one-dimensional Poisson process with parameter λV_0 on $[0, T_1]$.; (i.e. k is random). This step is performed as follows:

   1.1 Initialize $t = 0, k = 0$;

1.2 repeat

i. Generate $E \mapsto Exp(1)$; (E is an exponential random variable of parameter 1);

ii. $k := k + 1$; $t := t + \frac{E}{\lambda}$; $X_{1k} := \frac{t}{V_0}$;

until $X_{1k} \geq T_1$;

2. Generate $\mathbf{P_1}, ..., \mathbf{P_k}$ independent and uniform distributed points on $\mathbf{I_1}$;

3. Output $(X_{11}, \mathbf{P_1}), ..., (X_{1k}, \mathbf{P_k})$. (This is a realization of the Poisson process of intensity λ on the $k-$dimensional interval $\mathbf{I}$).

The following theorem justifies the algorithm.

Theorem 9. 1 *The points* $\mathbf{Q_i} = (X_{1i}, \mathbf{P_i})$, $1 \leq i \leq k$ *determine an uniform Poisson process of parameter* λ *on* $\mathbf{I}$.

Proof. Denote N the random number of points X_i (in $[0, T_1]$) and denote N_Q the number of points $\mathbf{Q_i}$ generated by the algorithm. Since the variables E in the algorithm are exponential then N is distributed as in formula (9.5) bellow and

$$P(N_Q = k) = P(N = k)V_0. \tag{9.4}$$

If we denote $\Lambda_0 = \lambda V_0$, then

$$P(N = k) = F^{(k-1)}(T_1) - F^{(k)}(T_1) \tag{9.5}$$

with

$$F^{(k)}(t) = \int_0^{T_1} \frac{(\Lambda_0 u)^k}{k!} e^{-\Lambda_0 u} du$$

because $F^{(k)}$ is the Erlang distribution (i.e. the convolution product of exponentials). Hence

$$P(N = k) = \frac{\Lambda_0^{k-1}}{(k-1)!} \int_0^{T_1} u^{k-1} e^{-\Lambda_0 u} \left(1 - \frac{\Lambda_0 u}{k}\right) du =$$

$$= \frac{\Lambda_0^{k-1}}{(k-1)!} \left[\int_0^{T_1} u^{k-1} e^{-\Lambda_0 u} du - \int_0^{T_1} \frac{\Lambda_0 u^k}{k} e^{-\Lambda_0 u} du\right] =$$

$$= \frac{\Lambda_0^{k-1}}{(k-1)1} \left[\int_0^{T_1} u^{k-1} e^{-\Lambda_0 u} du + \frac{u^k}{k} e^{-\Lambda_0 u}|_0^{T_1} - \int_0^{T_1} u^{k-1} e^{-\Lambda_0 u} du\right] = \frac{\Lambda_0^{k-1}}{(k-1)!} \frac{T_1^k}{k} e^{-\Lambda_0 T_1}.$$

Finally we get

$$P(N_Q = k) = P(N = k)V_0 = \frac{\Lambda^k}{k!} e^{-\Lambda},\ \Lambda = \lambda V,\ V = T_1 V_0$$

and the theorem is proved.

Particularly, an algorithm for simulating an uniform Poisson process with intensity λ, on the bivariate interval $[0,T]\times[0,L]$, is the following (see also Devroye-1986):

Algorithm SIMPOTL

1. Simulate $0<T_1<T_2<...<T_k$ a uniform Poisson Process with the rate λ, $T_k\leq T$ (i.e. k is random). This is done as follows:

   1.1 Initialize $t=0, k=0$;

   1.2 repeat

   i. Generate $E\mapsto Exp(1)$;

   ii. Take: $k:=k+1$; $t:=t+\frac{E}{\lambda}; T_k:=\frac{t}{L}$;

   until $t\geq T$

2. Generate $L_1, L_2, ..., L_k$ uniform on $[0,L]$;

3. Output $(T_1,L_1),(T_2,L_2),...,(T_k,L_k)$.

The sequence in step 3 defines a bivariate Poisson process with intensity λ, which is uniform on $[0,T]\times[0,L]$.

Now, using the algorithm SIMSCAN for the bivariate case, we can determine an empirical distribution of the scan statistics (i.e. a histogram) and then, estimate the critical value S_α, corresponding to a risk α, such as $P(S\geq S_\alpha)=\alpha$.

Notes.

a). All the algorithms are easily written to produce uniform integer values for the one-dimensional Poisson process, or for the two-dimensional Poisson process with integer coordinates of the sampling points.

b). In the one dimensional case, algorithms could be written for non homogenous Poisson process with intensity $\lambda(t)$; these algorithms use the *cumulative* intensity

$$\Lambda(t)=\int_0^t \lambda(u)du. \tag{9.6}$$

c). Algorithm SIMSCAN can be then applied to estimate the $\alpha-$ quantile of the scan statistics $S((u,v),k,T,L)$.

d) If the number N of points in $A=[0,T]\times[0,L]$ is binomially distributed (see Subsection 9.2 bellow for details) with the parameters $p,n, 0<$

$p < 1, n \in N^+$, then the simulation of the one scan statistics sampling value is given by the following algorithm:

Algorithm SCANBIN

1. Input parameters n and p;
2. Simulate a sampling value of N as binomial of parameters p, n;
3. Simulate N points $(T_i, L_i), 1 \leq i \leq N$, uniformly distributed in $A = [0, T] \times [0, L]$. These are simulated as follows:
   - 3.1 Take $i = 0$;
   - 3.2 repeat
     - i. Generate U_1, U_2 uniform and independent (0,1) random numbers;
     - ii. Take $T_1 = T \cdot U_1,\ L_1 = L \cdot U_2,\ i := i + 1$;

     until $i = N$;

Then an algorithm similar to SIMSCAN can be applied to produce a sampling value of the bivariate scan statistics.

In the following section we give some results on the implementation of the algorithm SIMPOTL for producing the empirical distribution of the scan statistics when the points $(T_i, L_i), 1 \leq i \leq N$ are realizations of an uniform bivariate Poisson process on $[0, T] \times [0, L]$ with intensity λ. Comparisons with the results of Alm-1997 and with the results of Haiman and Preda-2002 are presented.

9.1.3 Implementation and test results

Two programs using the algorithm SIMPOTL and a scan algorithm were written. One of them is written in C language and runs in Linux operating system, and one is written in C++ language and runs in Windows 2000 operating system. The two programs contain small differences concerning the scan module. We will present here the scan algorithm and the results obtained from the C program. We call the following algorithm SCAN2.

In order to understand this implementation we refer to Fig.9.1) in subsection 9.2 bellow.

We suppose that the scan surface and the scanning window are rectangles with the sides parallel with the horizontal and respectively vertical axes. The

width of the scan surface is denoted with T and its height with W. The width of the scanning window is denoted by u and its height by v.

Furthermore we suppose that both the scan surface and the scanning window are defined by two of their corners: the upper right corner and the lower left corner. We denote these corners with S_{right} and S_{left} for the surface, and with W_{right} and W_{left} for the window. Initially $S_{right} = (T, W)$ and $S_{left} = (0, 0)$. After generating the points $(T_i, W_i), 1 \leq i \leq N$, realizations of an uniform bivariate Poisson process on $[0, T] \times [0, L]$ with intensity λ, we begin the scanning process. We assume that the first position of the scanning window is characterized by the coordinates: $W_{right} = (T_{max}, W_{max})$, $W_{left} = (T_{max} - u, W_{max} - v)$, where W_{max} and T_{max} are the maximum values of W_i and respectively T_i.

The scanning widow moves over the scan surface on vertical bands, parallel with the y-axis. If we assume that the window is characterized by the coordinates $W_{right} = (x_r, y_r)$, $W_{left} = (x_l, y_l)$, then the following position of the scan window will be: $W_{right} = (x_r, y_r - d)$, $W_{left} = (x_l, y_l - d)$ where $d = \min\{y_r - y_{max}, y_l\}$ and y_{max} is the biggest coordinate y from the band which is smaller than y_r.

After a band was entirely scanned, the scanning window is repositioned on the next band in the following way: if the last position on the previous band was characterized by $W_{right} = (x_r, y_r)$, $W_{left} = (x_l, y_l)$, then the present position is characterized by: $W_{right} = (x_r - h, y_{max})$, $W_{left} = (x_r - h - u, y_{max} - v)$ where $h = \min\{x_r - x_{max}, x_l\}$, x_{max} is the biggest coordinate x smaller than x_r, and y_{max} is the maximum value of W_i for the points which have $x_r - h - u \leq T_i \leq x_r - h$. We use this method of scan because the points of the bivariate Poisson process are generated with T_i in increasing order.

For each position of the window there are counted the number of points n_W that are in the window. After the scanning of all the surface we determine the maximum value of the n_W. This maximum is a simulation value of the bivariate scan statistics, (i.e. $n_w = S_w$ in the notation of Section 1).

By repeating the algorithm SCAN2 for N runs ($N-$large), one determines the empirical distribution of the scan statistics.

The following tables contain test results. In the table there are also mentioned for comparison, simulated results produced by Alm [**?**] and approximations produced by a special method due to Haiman and Preda [**?**].

On the top of each table are mentioned particular values of the input data used:

- λ intensity of the bivariate Poisson process;

- W, T dimensions of the rectangle;
- u, v dimensions of the scanning window;
- N number of simulation runs;
- t time in seconds necessary for N simulation runs on a PC with an Athlon processor at 997 MHz and with 256 MB of RAM.

$\lambda = 0.01$, $W = T = 10$, $u = v = 1$, $N = 10000$
$p_1 = 0.01, \lambda_1' = 1$, $p_2 = 0.1, \lambda_2' = 0.1$

k	Poisson	H&P	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
1	0.9818	0.9826	0.9959	0.9524	0.9545
2	1.0000	0.9998	0.9999	0.9981	0.9988

$\lambda = 0.05$, $W = T = 10$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 5$, $p_2 = 0.1$,$\lambda_2' = 0.5$

k	Poisson	H&P	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
2	0.9859	0.9854	0.9905	0.8524	0.8547
3	0.9998	0.9996	0.9997	0.9825	0.9798
4	1.0000	0.9999	0.9999	0.9982	0.9982

$\lambda = 0.05$, $W = T = 200$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 5$, $p_2 = 0.1$,$\lambda_2' = 0.5$

k	Poisson	H&P	Alm	$bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
3	0.8620	0.8621	0.8935	0.8610	0.8580
4	0.9974	0.9976	0.9981	0.9972	0.9976
5	1.0000	0.9999	0.9999	1.00	1.00

$\lambda = 0.1$, $W = T = 50$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 10$,$p_2 = 0.1$,$\lambda_2' = 1$

k	Poisson	H&P	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
3	0.8762	0.8761	0.9052	0.8719	0.8744
4	0.9957	0.9957	0.9966	0.9944	0.9957
5	1.0000	0.9998	0.9999	0.9998	0.9999

$\lambda = 0.5$, $W = T = 10$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 0.50$,$p_2 = 0.1$,$\lambda_2' = 5$

k	Poisson	H&P	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
4	0.7865	0.7938	0.8343	0.7911	0.7932
5	0.9692	0.9707	0.9759	0.9731	0.9680
6	0.9968	0.9970	0.9974	0.9976	0.9971
7	0.9999	0.9997	0.9997	0.9999	0.9999

$\lambda = 1$, $W = T = 10$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 100$,$p_2 = 0.1$,$\lambda_2' = 10$

k	Poisson	H&P	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2(\lambda_2')$
6	0.8396	0.8248	0.8603	0.8436	0.8335
7	0.9695	0.9468	0.9732	0.9714	0.9690
8	0.9956	0.9691	0.9959	0.9949	0.9954

The following tables compare only our results with the results from the implementation of Alm.

$\lambda = 2$, $W = T = 20$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 200$,$p_2 = 0.1$,$\lambda_2' = 20$

k	Poisson	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
7	0.0007	0.0004	0.0002,	0.0001
9	0.5111	0.5283	0.5100	0.5119
11	0.9690	0.9640	0.9692	0.9653

$\lambda = 5$, $W = T = 20$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 500$,$p_2 = 0.1$,$\lambda_2' = 50$

k	Poisson	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
13	0.0010	0.0040	0.0002	0.0005
15	0.2390	0.2535	0.2645	0.2610
17	0.8540	0.8442	0.8509	0.8457

$\lambda = 5$, $W = T = 30$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 500$,$p_2 = 0.1$,$\lambda_2' = 50$

k	Poisson	Alm	$Bin(p_1, \lambda_1')$,	$Bin(p_2, \lambda_2')$
14	0.0003	0.0016	0.0004	0.0007
16	0.3167	0.3346	0.3202	0.3201
18	0.8851	0.8945	0.8908	0.8818
20	0.9893	0.9907	0.9899	0.9899
22	0.9994	0.9994	0.9992	0.9995

During various runs it resulted a convergence of the frequencies to the probabilities calculated by Haiman and Preda-2002 .

The number of runs $N = 10000$ considered in the tables seems to be large enough to ensure a good estimate of the probability distribution of the scan.

9.2.On Simulation of a Bivariate Uniform Binomial Process and its Use to Scan Statistics [3]

9.2.1 Basic notions

The aim of this subsection is to study a Bivariate Uniform Binomial Process (introduced by Vaduva and Alexe-2006) and to underline its use and to approximate (by simulation) the *critical value* of a bivariate scan test.

In subsection 9.1 and Suter, Vaduva, Alexe-2005 is considered a scan statistics for a Bivariate Uniform Poisson Process (BUPP) of the intensity λ. Here, in a similar manner, we will consider a discrete Bivariate Uniform Binomial process (BUBP) of the parameters p, λ, I, $p \in (0,1), \lambda \in R$, $I \subset R^2$, defined as

Definition 9. 4 *Let I be the bivariate interval $I = [0,T] \times [0,W], 0 < T, W < \infty$ and $n = [\lambda \times mes(I))]$, $mes(I) = T \times W, ([x] - integer\ part)$ a positive integer. Let p be a given probability, $0 < p < 1$. Let $X_1, X_2, ..., X_N$ be a set of random points, uniformly distributed on I where N is an integer random variable having a binomial distribution of parameters (n,p) (denoted $Bin(n,p)$). The set of points $X_1, X_2, ..., X_N$ is called a trajectory of the bivariate uniform binomial process of parameters (p, λ, I) on I (denoted $BUBP(p, \lambda, I)$) if:*

1). Points $X_1, X_2, ..., X_N$ are stochastically independent;

2). For any bivariate disjoint intervals $B_i = [\alpha_{1i}, \beta_{1i}] \times [\alpha_{2i}, \beta_{2i}], 1 \leq i \leq k, \alpha_{mi}, \beta_{mi} \in R, m = 1,2$ and every finite k, the number of points N_i which fall in B_i is binomially distributed $Bin(n_i, p), n_i = [\lambda \times (\beta_{1i} - \alpha_{1i}) \times (\beta_{2i} - \alpha_{2i})]$, and $N_1, N_2, ..., N_k$ are independent random variables. This process will be denoted for short as $\{X_t\} \mapsto BUBP(p, \lambda, I)$, $t \in \mathcal{N}$. The constant λ will be also called the intensity of the process.

[3]This research was done in the frame of the research Program "PAI-Brancusi" of Roanian-French cooperation,2005-2006.

This binomial process has a *property of stability* similar to a Poisson process, namely

Theorem 9. 2 *If $B_1, B_2, ..., B_k$ are disjoint subsets in the interval $I = [0,T] \times [0,W]$ and $\{X_t\} \mapsto BUBP(p,\lambda,I), t \in \mathcal{N}$ then the processes $\{X_i(t)\}, t \in \mathcal{N}$, are $\{X_i(t)\} \mapsto BUBP(p,\lambda,B_i)$ and X_i is independent of X_j, $i \neq j$. Particularly, if $B \subset I$ then $X_k \mapsto BUBP(p,\lambda,B)$, $k = [\lambda mes(B)]$. On the other hand if $I = B_1 \cup B_2 \cup ... \cup B_m$, $B_i \cap B_j = \emptyset, i \neq j$ and $\{X_t\} \mapsto BUBP(p,\lambda,B_i)$ then $X_{B_1} + X_{B_2} + ... + X_{B_m}$ is a $BUBP(p,\lambda,I)$.*

Proof. The proof can be easily done using the characteristic function of the binomial distribution. Thus, for the binomial distribution $X \mapsto Bin(n,p)$ the characteristic function is

$$\varphi(t) = E[e^{itX}] = (p + qe^{it})^n,\ t \in R,\ q = 1 - p. \tag{9.7}$$

Let us consider the random variables $X_{B_i}, 1 \leq i \leq m$ which are independent (points defining X_{B_i} beeing uniformly distributed on B_i), and $X_{B_i} \mapsto Bin([\lambda mes(B_i)], p)$. Then the random variable $Y = X_{B_1} + X_{B_2} + ... + X_{B_m}$ has the characteristic function of $Bin(n,p)$ distribution, i.e.

$$\varphi_{Y(t)} = \prod_{i=1}^{m} \varphi_{X_{B_i}}(t) = \prod_{i=1}^{m} (p + qe^{it})^{n_i} = (p + qe^{it})^n, n = \sum_{i=1}^{m} n_i, n_i = [\lambda mes(B_i)].$$

The last formula gives the end of the proof. %item For $j = 1$ to m do %par begin

Using algorithm SIMSCAMN from the subsection 9.1, we can estimate the prfobability distribution of the bivariate scan based on simulation of the BUBP. The empirical distribution of the bivariate scan statistic $S_{u,v)}$ helps to determine the critical value k_α of significance level α of a scan test.

Given the risk α, $0 < \alpha < 1$, the critical test value k_α of the scan statistics is defined as

$$P(S_{(u,v)} > k_\alpha) = \alpha. \tag{9.8}$$

In the next section we discuss the simulation of a trajectory of BUBP.

Algorithms for the simulation of a Bivariate Uniform Binomial Process

The Definition 9.4 leads to the following algorithm for simulating a *trajectory* of N points of the bivariate uniform binomial process of the intensity λ:

Algorithm SIMBIN2

1. (Preparatory step). $i = 0$, input $\lambda, W, T, p, 0 < p < 1$;
2. calculate $n = [\lambda \times W \times T]$;
3. Generate N a sampling value $Bin(n,p)$;

   repeat

Generate $U \mapsto uniform\,[0,T]$, $V \mapsto uniform\,[0,W]$;

(This can be done as follows:

- Generate U_1 uniform $(0,1)$; Take $U = U_1 T$;
- Generate U_2 uniform $(0,1)$; Take $V = U_2 W$;)

$i := i + 1$,; take $X_i = (U, V)$;

until $i = N$.

The algorithm produces the trajectory $X_1, X_2, ..., X_N$ of the $BUBP(p, \lambda, I)$.

Now, using the algorithm SIMSCAN for the bivariate case, we can determine an empirical distribution of the scan statistics (i.e. a histogram) and then, estimate the critical value k_α.

The simulation of the random variable X which is binomially distributed with the parameters $p, n, 0 < p < 1, n \in \mathcal{N}^+$, can be done in various ways (see Devroye-1986, Vaduva-1077). For large n,we use the fact that $X \mapsto Bin(n,p) \approx N(np, \sqrt{p(1-p)})$ i.e. X is normally distributed. Therefore, the algorithm to simulete X is the following (see Vaduva-1977)

Algorithm BINCL

1. Input n, p; Calculate $m = np$, $\sigma = \sqrt{np(1-p)}$;
2. Generate $Z \mapsto normal\,N(0,1)$;
3. Calculate $X = m + Z\sigma$; Take $N = round(X)$.

The function $round(x)$ gives the integer which is the closest to x.

Simulation of the normal deviate $Z \mapsto N(0,1)$ can be done in several ways; two methods will be presented in short in the following. The first one is based on *Central Limit Theorem* (CLT) [6,7].

Algorithm CLNORM (simulates $Z \mapsto N(0,1)$ based on CLT)

1. $Z = 0$;
2. **for** $i := 1$ **to** 12 **do begin**

   Generate U uniform $(0,1)$;

   $Z := Z + U$;

   end;

Another algorithm combines a *rejection* (*enveloping*) and a *discrete composition* method (see Vaduva-1977, Devroye-1986. It looks as follows:

Algorithm RJNORM

1. **repeat**

   Generate $U_1 \mapsto uniform(0,1)$;

   Generate $Y \mapsto Exp(0,1)$;

   (This can be done by the *inverse* method as follows:

   Generate $U \mapsto uniform(0,1)$;

while $U \leq 0.0000001$ **do** Generate $U \mapsto uniform(0,1)$;
$Y := -\log(U)$);

2. **until** $U_1 \leq e^{-\frac{Y^2}{2}+Y-0.5}$;
3. Take $Z_1 := Y$;
4. Generate $U \mapsto uniform(0,1)$;
   if $U \leq 0.5$ **then** $s := 1$ **else** $s := -1$; (s is a random sign);
5. take $Z := sZ_1$. (Z is normal $N(0,1)$).

In the following section we give some results on the implementation of the algorithm SIMSCAN for producing the empirical distribution of the scan statistics when the points $(T_i, W_i), 1 \leq i \leq N$ are realizations of a bivariate uniform binomial process on $[0,T] \times [0,W]$ with intensity λ. Comparisons with the results of Alm-1997 and with the results of Haiman and Preda-2002, in the case of BUPP process, are presented.In order to compare the results with the bivariate uniform binomial process, we use the fact that the binomial distribution $Bin(n,p)$, with n large is approximated by a Poisson distribution $Poisson(\lambda), \lambda = np$. When we refer to the $BUPP(\lambda, I)$ process and to the binomial $BUBP(p, \lambda', I)$ process, both on $I = [0,T] \times [0,W]$, we must distinguish betweem the intensities λ (for Poisson) and λ' (for binomial). In fact, for n large, we must have

$$\lambda WT = \lambda' pWT \tag{9.9}$$

which gives

$$\lambda' = \frac{\lambda}{p}. \tag{9.9'}$$

9.2.3 Implementation and test results

In this implementation we use one of the programs presented in Vaduva,Alexe-2006, namely the algorithm called SCAN2. In fact the main ideas derive from the algorithm SIMSCAN presented in the section 1. The discrete process used in this implementation is either $BUPP$ or $BUBP$ according to SIMBIN2. In the following, we underline the main steps of SCAN2 (see Suter,Vaduva,Alexe-2006)). Figure 9.1 gives some hints on the construction of SCAN2.

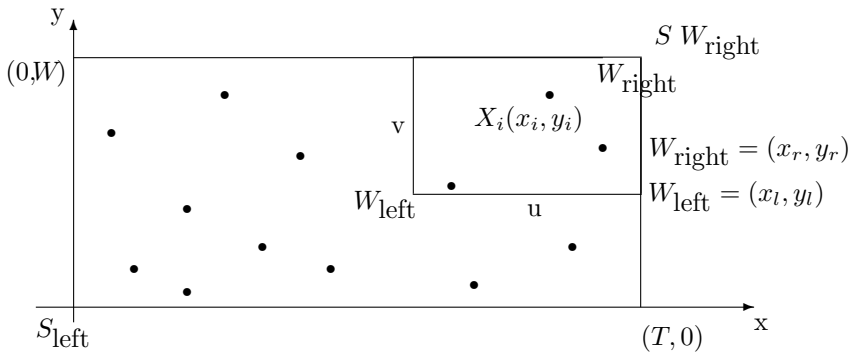

Fig. 9.1 a)

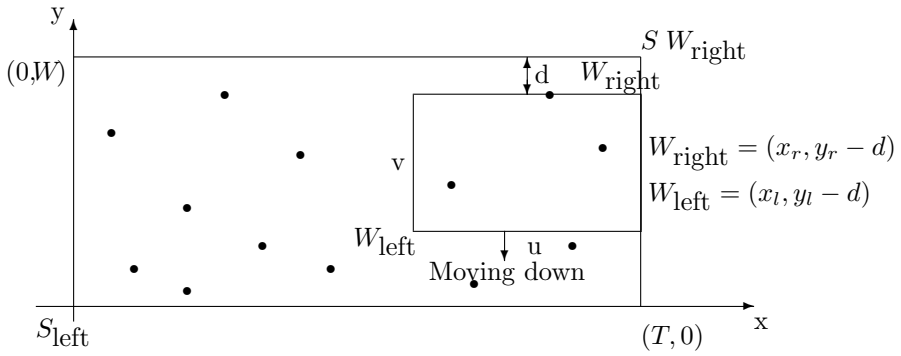

Fig. 9.1 b)

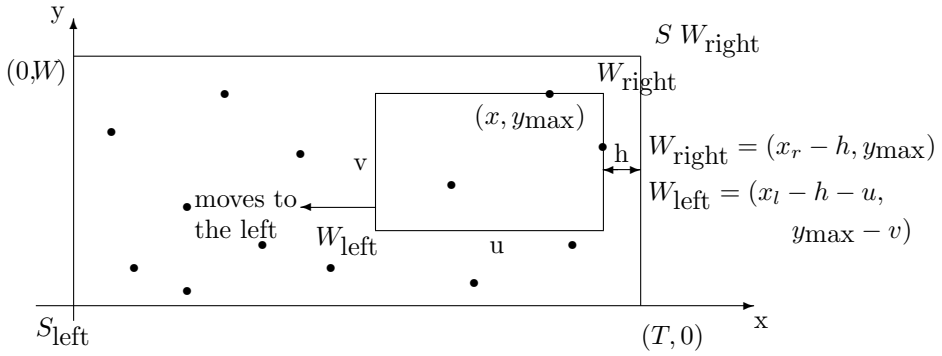

Fig.9. 1 c)

Figure 9.1. Hints for scanning algorithm:
a) initial pozition of scan window;
b) moving down the scan window;
c) moving window to the left.

We suppose that the scan surface (called also the *map*) and the scanning window are rectangles with the sides parallel with the horizontal and respectively vertical axes. The width of the map is denoted by T and its height by W. The width of the scanning window is denoted by u and its height by v.

Furthermore we suppose that both the scan surface and the scanning window

are defined by two of their corners: the upper right corner and the lower left corner. We denote these corners by S_{right} and S_{left} for the surface, and W_{right} and W_{left} for the window. Initially $S_{right} = (T, W)$ and $S_{left} = (0, 0)$. After generating the points $(T_i, W_i), 1 \leq i \leq M$, realizations of an bivariate uniform binomial process on $[0, T] \times [0, W]$ with intensity λ, we begin the scanning process. First we order the simulated points with respect to T_i. Assume that this was already done. Then, let us assume that the first position of the scanning window is characterized by the coordinates: $W_{right} = (T, W)$, $W_{left} = (T - u, W - v)$.

The scanning window moves over the scan surface on vertical bands, parallel with the y-axis. If we assume that the window is characterized by the coordinates $W_{right} = (x_r, y_r)$, $W_{left} = (x_l, y_l)$, then the following position of the scan window will be: $W_{right} = (x_r, y_r - d)$, $W_{left} = (x_l, y_l - d)$ where $d = \min\{y_r - y_{max}, y_l\}$ and y_{max} is the biggest coordinate y from the band which is smaller than y_r. (See Fig 9.1).

After a band was entirely scanned, the scanning window is repositioned on the next band in the following way: if the last position on the previous band was characterized by $W_{right} = (x_r, y_r)$, $W_{left} = (x_l, y_l)$, then the present position is characterized by: $W_{right} = (x_r - h, y_{max})$, $W_{left} = (x_r - h - u, y_{max} - v)$ where $h = \min\{x_r - x_{max}, x_l\}$, x_{max} is the biggest coordinate x smaller than x_r, and y_{max} is the maximum value of W_i for the points which have $x_r - h - u \leq T_i \leq x_r - h$. We use this method of scan because the points of the $BUBP$ or $BUPP$ have the coordinates T_i in increasing order.

For each position of the window there is counted the number of points that are in the window and is stored the largest number n_w of points found during the scan process. This maximum n_w is a simulation value of the bivariate scan statistics, (i.e. $n_w = S_w = S$ in the notation of Section 9.1).

By repeating the algorithm SCAN2 for N runs or *iterations* ($N-$large), one determines the empirical distribution of the scan statistics.

The following tables contain test results. In each table there are also mentioned for comparison, simulated results produced by Alm-1997 and approximations produced by a special method due to Haiman and Preda-2002. (Some of the tables are reproduced fron Alm-1997). On the top of each table are mentioned particular values of the input data used, namely:

- λ intensity of the bivariate Poisson process;
- W, T dimensions of the rectangle;
- u, v dimensions of the scanning window;
- N number of simulation runs;
- p_1, p_2 and λ_1', λ_2' refer to different values of parameters of binomial processes corresponding to the approximate parameter of the Poisson process (determined according to (9.9),(9.9')).
- k is the value of scan statistics for which is calculated empirical probability;
- $H\&P$ in the table refers to the results from "Haiman and Preda" (2002).
- P refers to BUPP; A refers to Alm; B refers to BUBP; The entries in the following tables represent probabilities $P(S = k)$ where $S = S((u, v), T, W)$ is the bivariate scan statistics from Definition 9.1.

$\lambda = 0.05$, $W = T = 10$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 5$, $p_2 = 0.1$,$\lambda_2' = 0.5$

k	P	H&P	A	$B(p_1, \lambda_1')$	$B(p_2, \lambda_2')$
2	0.9859	0.9854	0.9905	0.8524	0.8547
3	0.9998	0.9996	0.9997	0.9825	0.9798
4	1.0000	0.9999	0.9999	0.9982	0.9982

$\lambda = 0.1$, $W = T = 50$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 10$,$p_2 = 0.1$,$\lambda_2' = 1$

k	P	H&P	A$B(p_1, \lambda_1')$	$B(p_2, \lambda_2')$	
3	0.8762	0.8761	0.9052	0.8719	0.8744
4	0.9957	0.9957	0.9966	0.9944	0.9957
5	1.0000	0.9998	0.9999	0.9998	0.9999

$\lambda = 0.5$, $W = T = 10$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 0.50$,$p_2 = 0.1$,$\lambda_2' = 5$

k	P	H&P	A	$B(p_1, \lambda_1')$	$B(p_2, \lambda_2')$
4	0.7865	0.7938	0.8343	0.7911	0.7932
5	0.9692	0.9707	0.9759	0.9731	0.9680
6	0.9968	0.9970	0.9974	0.9976	0.9971
7	0.9999	0.9997	0.9997	0.9999	0.9999

$\lambda = 1$, $W = T = 10$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 100$,$p_2 = 0.1$,$\lambda_2' = 10$

k	P	H&P	A	$B(p_1, \lambda_1')$	$B(p_2(\lambda_2')$
6	0.8396	0.8248	0.8603	0.8436	0.8335
7	0.9695	0.9468	0.9732	0.9714	0.9690
8	0.9956	0.9691	0.9959	0.9949	0.9954

The following tables compare only our results with the results from the implementation of Alm.

$\lambda = 2$, $W = T = 20$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 200$,$p_2 = 0.1$,$\lambda_2' = 20$

k	Poisson	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
7	0.0007	0.0004	0.0002,	0.0001
9	0.5111	0.5283	0.5100	0.5119
11	0.9690	0.9640	0.9692	0.9653

$\lambda = 5$, $W = T = 20$, $u = v = 1$, $N = 10000$
$p_1 = 0.01$,$\lambda_1' = 500$,$p_2 = 0.1$,$\lambda_2' = 50$

k	Poisson	Alm	$Bin(p_1, \lambda_1')$	$Bin(p_2, \lambda_2')$
13	0.0010	0.0040	0.0002	0.0005
15	0.2390	0.2535	0.2645	0.2610
17	0.8540	0.8442	0.8509	0.8457

The results in the tables show a good agreement between the distributions of scan statistics for all the compared cases (i.e. Poisson, Alm, $H\&P$ and binomial). For values of k of practical interest (see bellow), the values of $P(S = k)$ are almost equal for both $BUPP$ and $BUBP$.

During various runs it resulted a convergence of the frequencies to the probabilities calculated by Haiman and Preda-2002.

The number of runs $N = 10000$ considered in the tables seems to be large enough to ensure a good estimate of the probability distribution of the scan. Any $N > 10000$ will be more succesful.

The tests done here legitimate both assumptions (Poisson or binomial) for defining, via simulation, the critical value of the scan test. Therefore, in the next section (application) we will use the Poisson process. (Runs for BUBP are time consuming!).

BUBP and BUPP processes may be used as equal alternatives in various applications where discrete random (*uniform*) events can occur on some surface of material or geographic area.

9.3 An Application to Healthcare

Here we present an application of scan statistics to analyze the cancer disease for children under age 16 in the region North Pas de Callé (north of France). The region consists of two departments, each department contains some *arondisments* and an arondisment consists of *cantons.* The data consisted in the number of diseased children in each canton (considered the scan window). The total population in the region is about 573500 inhabitants and total number of ill children is $N = 497$. In one canton of the first department was found the largest number of ill children as beeing 9 from a population of $\pi_1 = 1600$ and in other canton of the second department were found 7 ill children from a population of $\pi_2 = 2300$ inhabitants. These two cantons contain the largest figures of ill children.Administrative authorities want to know if these large figures are *natural* or they are determined by some environmental fators of cantons.(The whole region is a mining region!). Therefore, under the *natural hypothesis* (denoted H_0) we asume that number of diseased children in the region is a BUBP (or BUPP) process and we must test the hypotheses H_{01} and H_{02} that the numbers of 9 respectively 7 ill children are considered *normal* or *dangerous* events from the healthcare point of view. Therefore we are in the theoretical situation discussed in the previous sections.

The collection of data for our application follows from the procedure used in [3,4] which defines the dimensions of the hypothetic geographic region (i.e. the

map) taking into consideration the seize of population in the region and defines the scan window using the size of population in the cantons with the largest number of ill children. As the geographical map of the region is not a regular one, we consider it as a square $[0, W] \times [0, T]$ with $W = T = \sqrt{P}$ where P is the seize of population of the region (in our case $P = 573500$), hence $W = T = 757.3$. Similarly, the scan windows are respectively $u_1 = v_1 = \sqrt{\pi_1} = \sqrt{1600} = 40$, $u_2 = v_2 = \sqrt{\pi_2} = \sqrt{2300} = 47.95$. The intensity of the Poissin process (over the region) is $\lambda = \frac{N}{P} = 0.0008666$ and the parameters for Poisson processes for the two cantons are $\Lambda_1 = \lambda\pi_1 = 1.384$, $\Lambda_2 = \lambda\pi_2 = 1.9918$. To use the bivariate uniform binomial processes, we need to estimate parameters p_1, p_2. These are simply defined as $p_1 = \frac{7}{N} = 0.014$, $p_2 = \frac{9}{N} = 0.018$. Hence, according to (1.5') we have for BUBP the parameters: $\lambda_1' = \Lambda_1/p_1 = 99$,$\lambda_2' = \Lambda_2/p_2 = 110.6$. (For BUBP thesefigures are not used).

In order to test the mentioned hypotheses H_{01}, H_{02} we use the simulation procedure presented in the previous sections. We use also the property of the scan statistics which says that $S((u, v), N, W, T) = S((1, 1), N, W/u, T/u)$. Hence, for the first canton $W_1 := W/u_1 = T_1 := T/v_1 = 747/40 = 18.93$, $W_2 := W/u_2 = T := T/47.35 = 757.3/47.35 = 15.77$.

The simulation of the scan statistics , coresponding to data under Poisson hypothesis, is resumed in the following tables which contain the values of $S = k$ and corresponding frenquences f:

$W = T = 18.93, u = v = 1, \Lambda = 1.3865, N = 100000 = iterations$

k	6	7	8	9	10	11	12	13
f	2576	36833	42728	14302	3006	471	75	9

$W = T = 15.77, u = v = 1, \Lambda = 1.99318, N = 100000 = iterations$

k	7	8	9	10	11	12	13	14	15	16
f	1005	22369	44235	23462	6983	1564	308	66	6	2

From the first table one can see that $P(S \leq 9) = 0.96439$. Therefore H_{01} can be accepted with a risk of $\alpha = 0.03561$. (Hence $k_\alpha = 9$ and the critical region of the scan test is $\mathcal{C} = \{k | k > k_\alpha\}$).

From the second table one can see that $P(S \leq 10) = 0.91071$. The hypothesis H_{02} is also accepted with $\alpha = 0.09929, k_\alpha = 10$, and the critical region $\mathcal{C} = \{k | k > k_\alpha\}$. Since in the second case (the canton 2) there are 7 ill children, and this is the *second large value* in the region, the frequencies in the second table must be moved one step to the left. Therefore for the second large value (i.e. $k = 7$) the critical region is $\mathcal{C} = \{k | k > k_\alpha\}, k_\alpha = 11, \alpha = 0.01946$ and this gives a better reason to accept the hypothesis H_{02}.

In conclusion, the figures of ill children ($k = 9, k = 6$) are *natural*. There are no problems for authorities, concerning the cancer healtcare.

Bibliography

Alm, S.E. (1997). "On the Distributions of Scan Statistics of a Two-dimensional

Poisson Process". *Advances in Applied Probability.*, **1,** 1-18. The article presents some approximates for the bivariate scan statistics.

Anderssen, R.S.,Bloomfeld, P.(1975). "Properties of the random search in Global optimization", *JOTA,Vol.16,Nr.5-6.* The paper proves the importance of using uniform random samples the restriction domain D.

Davison, A.C. and Hinkley,D.V.(1997).*Bootstrap Methods and their Applications,* Cambridge University Press, Cambridge. This is an up-to-date monograph on bootstrap resampling methods and their applications in statistics.

Devroye, Luc.(1986).*Non-Uniform Random Variate Generation,* Springer Verlag, New York. The book, after fourteen years of being published, is still a complete monograph which contains all methods invented in the field of random variate simulation, including algorithms for simulating some stochastic processes and particular types of random vectors also.

Efron, B. and Tibshirani, R.J.(1993).*An Introduction to the Bootstrap,* Chapman & Hall, New York. This is a pioneering book in the field of bootstrap methodology.

Ermakov, E.S.(1971).*Monte Carlo Method and Related Questions,* (Russian), "Nauka" Publishing House, Moscow. The book dominated the literature on Monte Carlo methods in eastern Europe during the 70's. It is written in an accurate mathematical form and discusses some particuler simulation models.

Fishman, G.S.(1996).*Monte Carlo:Concepts, Algorithms and Applications,* Springer Verlag, New York. A modern and consistent monograph studying a wide area of theoretical questions regarding computer simulation and Monte Carlo methods as well as various applications.

Garzia,R.F. and Garzia, M.R.(1990).*Network Modeling, Simulation, and Analysis,* Marcel Dekker, New York. The book does not fit within the framework of this contribution, but it is important for the algorithmic way of analyzing stochastic networks which may help in solving particular types of optimization problems.

Gentle,J.E.(1998).Random Number Generation and Monte-Carlo Methods, (Statistics and Computing Series), Springer Verlag, New York. The book has inspired me to discuss some algorithms for simulating uniformly distributed random numbers; it contains almost all ideas in this respect, underlines the way of analyzing the quality of random numbers and also presents in a modern way the problems related to Monte Carlo methods.

Glaz, J. Naus, J. and Wallenstein, S. (2001). *Scan Statistics,* Springer Verlag, New York, Berlin, Heidelberg. The monograph studies various types of scan statistics in one and two dimensions and gives some applications.

Gnedenko,B.V. (1943)."Sur la Distribution Limite du Terme Maximum d'une Seie Aleatoare", *Anals of Maths, Vol 44, Nr. 3.* Haiman, G. and Preda C. (2002).

"A New Method of Estimating The Distribution for a Two-Dimensioinal Poisson Process". *Methodology and Computing in Applied Probability.* The article uses a random grid to determine lower and upper bounds for the distribution of a bivatiate scan statistics on a rectanmgle.

Hanssmann, F. (1968). *Operations Research in Production and Inventory Control,* John Wiley and Sons, New York, London. The book is an introductory textbook on inventory models.

Knuth, D.E.(1981).*The Art of Computer Programming, Volume 2, Seminumerical Algorithms,* second edition, Addison-Wesley Publishing Company, Reading, Massachusetts. This is the first monograph of the end of the 60's which describes and performs a deep analysis of the arithmrtic problems of computer generation of random numbers and their testing.

Morgan, Byron T. (1984). *Elements of Simulation.* Chapman & Hall, New York, London. The book is a good textbook containing a large number of exercices related to simulation of random variates.

Pham, D.T. and Karaboga, D.(2000).*Intelligent Optimisation Techniques,* Springer Verlag, Berlin. The book is an introductory text to optimization methods based on *simulated annealing* and genetic algorithms.

Ripley, B.(1986).*Stochastic Simulation,* Wiley, New York. This is a concise text underlining in an accurate way the main problems of Monte Carlo simulation. Various applications are also underlined.

Robert,C.P. and Casella, G.(1999). Monte Carlo Statistical Methods, Springer Verlag, New York. This is a complete monograph on statistical simulation which performs a good analysis of Markov Chain Monte Carlo methods and their applications. Monte Carlo integration and Monte Carlo optimization are also discussed.

Ross, Sheldom M. (1997). *Simulation. Second Edition.* Academic Press, San Diego, New York, London. The book derscribes discrete event simulation approach with application to queueing theory and presents some variance reduction techniques for Monte Carlo calculations.

Rubinstein,Y.,Weissman, I.(1979)."The Monte Carlo Method for Global Optimization", *Cahiers d'Etudes de R.O., Vol. 21, Nr. 2.* The paper underlines the main problems in global optimization by using random search.

Spriet ,Jan, VanstenKiste Ghislain, C. (1982).*Computer Aided Modeling and Simulation.* Academic Press, New York. The book is a monograph dedicated to system modeling, based mainly ob using differential equations and numerical methods.

Suter, Florentina, Vaduva, I., Alexe, Bogdan. (2005). "On Simulation of Poisson Processees Used to Analyse a Bivariate Scan Statistics". *Analele Universitatii*

"Al.I.Cuza" Iasi, Sectia Informatica, Tom XV, p. 23-35. The paper gives an algorithm to estimate via simulation the critical test value for a bivariate scan statistics.

Vaduva, I, Dinescu, C., Savulescu B. (1974). *Mathematical methods for production management,* Educational Publishing House, Bucharest, Vpl. 2. (Published in Romanian).The book contains introduction to queueing models and inventory models and applications of graph theory and optimization problems.

Vaduva, I. (1977). *Computer Simulation Models.* Techn. Pub. House, Bucharest Romania (written in Romanian). The book presents metzhods for simulation of random veriables, random vectors and stochastic processes, and simulation models for queueing and inventory systems.

Vaduva, I., Spircu,L. (1981)."On Convergence of Monte Carlo Method for Optimization", *Proc. Symp, on Economic Cybernetik, Rostok, Germany, 14 p, (mycrofilm).*

Văduva, I. (1994)."Fast algorithms for computer generation of random vectors used in reliability and applications". *Preprint Nr.1603, Januar 1994, TH-Darmstadt.* Here are found methods for simulating various multivariate distributions, including those based on transforming uniformly distributed vectors into other types of random vectors. Algorithms for the simulation of univariate and multivariate distributions used in reliability are particularly discussed.

Zeigler,B.P., Praehofer H. (2000). *Theory of Modeling and Simulation.* The monograph gives a formal description of discrete event systems with application to discrete simulatiom modeling.

Further References on **Scan Statistics**

Glaz, J., Ballakrishnan, B. (1999). *Scan Statistics and Applications,* Birkhäuser, Boston. This isa the forst book on Scan Statistics, ilustrated by various examples and applications.

Glaz, J, Naus, J. and Wallenstein, S. (2001). *Scan Statistics,* Springer Verlag, New York,Berlin, Heilderberg. This book is a mon ograph updating all rtesults on scan statistics in one or many dimensions, refering to rectangular or circular maps.

Haiman, G. and Preda, C. (2002) "A New Method of Estimating The Distribution of Scan Statistics for a Two-Dimensional Poisson Processes" *Advances in Applied Probability,* **1, p.1-18.**

Vaduva,I., Alexe, B. (2006). "On Simulation of a Bivariate Uniform Binomia;l Process to be used for Analyzing Scan Statistics, " *Analele Univ. Bucuresti, Matematica- Informatica, Ano LV, p. 153-164.* The paper defines the Bivariate Uniform Binomial Process, gives algorithms for simulating such a process and describes estimation of probability distribution of a bivariate scan statistics.