POLIANA ȘTEFĂNESCU
ST. ȘTEFĂNESCU
Computing Centre
University of Bucharest

A Simple Monte Carlo Simulation Technique with Applications in Large Systems

It is proposed a simple Monte Carlo simulation procedure for estimating the moments of random variables which are functions having random parameters.

This simulation technique can be also applied to determine different statistic indicators for economic processes and generally can be easily used in large systems with a lot of dependent random variables.

Key words: simulation, Monte Carlo technique, large systems, estimation of moments.

1. INTRODUCTION

1.0. Numerous practical problems require to determine the mean and the variance of a random vector $(Y_1, Y_2, ..., Y_m)$ where each Y_i depends on some random dependent factors $X_1, X_2, ..., X_n$,

$$Y_i = h_i(X_1, X_2, ..., X_n), \quad 1 \leqslant i \leqslant m. \tag{1}$$

We will simplify the exposure supposing that $m = 1$.

1.1. When it is known the probability density function (p.d.f.) $f(x_1, x_2, ..., x_n; a_1, ..., a_p)$ of the random vector $(X_1, X_2, ..., X_n)$ then the (noncentral) moment of order t, $M_t(Y)$, for the random variable Y,

$$Y = h(X_1, X_2, ..., X_n) \tag{2}$$

is given by the formula

$$M_t(Y) = \int_{R^n} (h(x_1, x_2, ..., x_n))^t \cdot f(x_1, ..., x_n; a_1, ..., a_p)\, dx_1\, dx_2 \ldots dx_n. \tag{3}$$

Therefore the mean and the variance of the random variable (r.v.) Y are equal to $M_1(Y)$, respectively $D^2(Y) = M_2(Y) - (M_1(Y))^2$.

It is difficult to apply this technique in practice because:

— Generally, the p.d.f. f isn't known.

— When we suppose the form of the p.d.f. f is known then we must find the estimations $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_p$ of the parameters $a_1, a_2, \ldots, a_p$ from experimental data. So, usually, we must solve a system with nonlinear equations.

Besides, when the number n of the factors X_i increases, it is difficult to have sufficient complete simultaneous experiments regarding the random vector $(X_1, X_2, \ldots, X_n)$.

1.2. Let $c_i, d_i^2, 1 \leqslant i \leqslant n$, be the mean, respectively the variance of the r.v. X_i, $1 \leqslant i \leqslant n$, and r_{ij} be the correlation between X_i and X_j, $1 \leqslant i, j \leqslant n$.

Considering the Taylor's polynomial T_h of degree 2 at the point $c = (c_1, c_2, \ldots, c_n)$ as approximation for the function h, we obtain the r.v. $Y_T = T_h(X_1, X_2, \ldots, X_n)$. The r.v. Y_T is an "approximation" of the r.v. $Y = h(X_1, X_2, \ldots, X_n)$ and hence we can use $M_1(Y_T)$ and $D^2(Y_T)$ as estimations of $M_1(Y)$, respectively $D^2(Y)$.

Therefore we obtain (Hahn, Shapiro [3]):

$$M_1(Y_T) = h(c) + \sum_{i=1}^{n} (d_i^2/2) \cdot \left(\frac{\partial^2 h}{\partial x_i^2}(c)\right) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_i \cdot d_j \cdot r_{ij} \cdot \left(\frac{\partial^2 h}{\partial x_i \cdot \partial x_j}(c)\right)$$

$$D^2(Y_T) = \sum_{i=1}^{n} d_i^2 \cdot \left(\frac{\partial h}{\partial x_i}(c)\right)^2 + 2 \cdot \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_i \cdot d_j \cdot r_{ij} \cdot \left(\frac{\partial h}{\partial x_i}(c)\right) \cdot \left(\frac{\partial h}{\partial x_j}(c)\right). \tag{4}$$

Usually, in practice, the derivatives of the function h in the point c are replaced by divided differences of h.

1.3. A Monte Carlo procedure (Bradley, Fox, Schrage [1]) can also be used to find the estimators $\hat{M}_1(Y)$, $\hat{D}^2(Y)$ for $M_1(Y)$, respectively $D^2(Y)$. So generating q independent random vectors $X^{(j)}$, $X^{(j)} = (X_1^{(j)}, X_2^{(j)}, \ldots, X_n^{(j)})$ $1 \leqslant j \leqslant q$, having $f(x_1, \ldots, x_n; a_1, \ldots, a_p)$ as p.d.f., it follows

$$\hat{M}_1(Y) = \left(\sum_{j=1}^{q} h(x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)})\right) \Big/ q$$

$$\hat{D}^2(Y) = (1/q) \cdot \sum_{j=1}^{q} (h(x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)}) - \hat{M}_1(Y))^2. \tag{5}$$

Generally, it is not so easy to generate efficiently random vectors $(X_1, X_2, \ldots, X_n)$ with a given p.d.f. $f(x_1, x_2, \ldots, x_n)$.

2. A SIMPLE PROCEDURE

2.1. Usually, in practice, we haven't the p.d.f. — s for the r.v. — s X_i but there are known their means, their variances (c_i, respectively d_i^2) and the correlations r_{ij}, $-1 \leqslant r_{ij} \leqslant 1$, between X_i and X_j.

For estimating $M_1(Y)$ and $D^2(Y)$, where the r.v. Y is given by (2), we propose to use the discrete r.v. — s Z_i

$$Z_i = c_i + d_i \cdot B_i \tag{6}$$

where B_i, $1 \leqslant i \leqslant n$, are $\{-1, 1\}$ valued r.v. — s taking the value 1 with probability 1/2,

$$\Pr(B_i = 1) = \Pr(B_i = -1) = 1/2,\ 1 \leqslant i \leqslant n. \tag{7}$$

If we impose that the both random vectors $(X_1, X_2, ..., X_n)$ and $(Z_1, Z_2, ..., Z_n)$ have the same covariance matrix, it follows

$$\text{Corr}\ (B_i, B_j) = r_{ij},\ 1 \leqslant i, j \leqslant n. \tag{8}$$

So $M_1(Y)$ and $D^2(Y)$ can be approximated by $M_1(h(Z_1, Z_2, ..., Z_n))$, respectively $D^2(h(Z_1, Z_2, ..., Z_n))$.

2.2. First we assume that it is known the p.d.f. $f_1(k_1, ..., k_n)$ of the random vector $(B_1, B_2, ..., B_n)$, that is

$$f_1(k_1, k_2, ..., k_n) = \Pr(B_1 = k_1, B_2 = k_2, ..., B_n = k_n),$$
$$k_1, k_2, ..., k_n \in \{-1, 1\} \tag{9}$$

Notation. For every $k_1, k_2, ..., k_n \in \{-1, 1\}$ we denote by $g(k_1, k_2, ..., k_{j-1}, 0, k_{j+1}, ..., k_n)$, $1 \leqslant j \leqslant n$, the following sum $g(k_1, k_2, ..., k_{j-1}, -1, k_{j+1}, ..., k_n) + g(k_1, k_2, ..., k_{j-1}, 1, k_{j+1}, ..., k_n)$.

Applying a Monte Carlo procedure, after q simulation steps, we obtain the estimated values c_0, d_0 for $M_1(h(Z_1, Z_2, ..., Z_n))$, respectively $D^2(h(Z_1, Z_2, ..., Z_n))$.

Algorithm AMC (Monte Carlo algorithm)

Step 0. Inputs: $q, c_1, d_1, c_2, d_2, ..., c_n, d_n, h, f_1$

Step 1. $t \leftarrow 0$; $c_0 \leftarrow 0$; $d_0 \leftarrow 0$;

Step 2. $t \leftarrow t + 1$; If $t > q$ then go to step 5.

Step 3. Generate a random vector $(B_1, B_2, ..., B_n)$ having the p.d.f.

$f_1(k_1, k_2, ..., k_n)$, $k_i \in \{-1, 1\}$, $1 \leqslant i \leqslant n$.

Step 4. $Z_1 \leftarrow c_i + d_i \cdot B_i$, $1 \leqslant i \leqslant n$

$h_0 \leftarrow h(Z_1, Z_2, ..., Z_n)$

$c_0 \leftarrow c_0 + h_0$; $\quad d_0 \leftarrow d_0 + h_0 \cdot h_0$;

Go to step 2.

Step 5. $c_0 \leftarrow c_0/q$; $\quad d_0 \leftarrow d_0/q - q \cdot c_0 \cdot c_0$

Print c_0, d_0. STOP.

We can use, for example, the inversion method (Bradley, Fox, Schrage [1]) to generate (in the Step 3 of the AMC algorithm) the random vector $(B_1, B_2, ..., B_n)$ with p.d.f. f_1.

Algorithm AG (algorithm for generating binary random vectors; inversion method).

Step 0. Input: function f_1

Step 1. $j \leftarrow 1$

Generate a uniform (0,1] random variate U

If $U \leqslant f_1(-1, 0, 0, ..., 0)$ then $B_1 \leftarrow -1$;

else $B_1 \leftarrow 1$;

Step 2. $j \leftarrow j + 1$; If $j > n$ then go to step 4.

Step 3. $w \leftarrow f_1(B_1, B_2, ..., B_{j-1}, -1, 0, ..., 0)/f_1(B_1, B_2, ..., B_{j-1}, 0, ..., 0)$

Generate an independent uniform (0,1] random variate U.

If $U \leqslant W$ then $B_j \leftarrow -1$;

else $B_j \leftarrow 1$;

Go to step 2.

Step 4. Print $(B_1, B_2, ..., B_n)$. STOP.

3. EXAMPLES

We will specify some possible forms of the p.d.f. f_1 for the random vector $(B_1, B_2, ..., B_n)$ such as to verify the conditions (7), (8).

3.1. For every $k_1, k_2, ..., k_n \in \{-1, 1\}$ we define the function

$$g_1(n; k_1, k_2, ..., k_n) = (1 + \sum_{1 \leqslant s \leqslant t \leqslant n} k_s \cdot k_t \cdot r_{st})/2^n. \tag{10}$$

Proposition 1. If the function g_1 (formula (10)) is nenegative,

$$g_1(n; k_1, k_2, ..., k_n) \geqslant 0; \; k_1, k_2, ..., k_n \in \{-1, 1\} \tag{11}$$

then the random vector $(B_1, B_2, ..., B_n)$ having g_1 as p.d.f. satisfies the relations (7) and (8).

Proof. From a straight computation it follows the equality

$$g_1(n; 0, 0, ..., 0, k_i, 0, ..., 0, k_j, 0, ..., 0) = (1 + k_i \cdot k_j \cdot r_{ij})/4$$

for every $1 \leqslant i \leqslant j \leqslant n$, that is

$$\Pr(B_i = -1, B_j = -1) = \Pr(B_i = 1, B_j = 1) = (1 + r_{ij})/4$$

$$\Pr(B_i = -1, B_j = 1) = \Pr(B_i = 1, B_j = -1) = (1 - r_{ij})/4 \tag{12}$$

and hence

$$\Pr(B_i = -1) = \Pr(B_i = -1, B_j = -1) + \Pr(B_i = -1, B_j = 1) = 1/2$$

$$\Pr(B_i = 1) = \Pr(B_i = 1, B_j = -1) + \Pr(B_i = 1, B_j = 1) = 1/2$$

$$M_1(B_i) = 0; \quad D^2(B_i) = 1$$

$$\operatorname{Corr}(B_i, B_j) = \Pr(B_i = -1, B_j = -1) - \Pr(B_i = 1, B_j = -1) - \Pr(B_i = -1, B_j = 1) + \Pr(B_i = 1, B_j = 1) = r_{ij}$$

the proposition being proved.

Remark 1. For every $-1 \leqslant r_{st} \leqslant 1$ it results

$$g_1(n;\ k_1, k_2, ..., k_n) \leqslant (1 + n \cdot (n+1)/2))/2^n \leqslant 1$$

but we haven't always the inequality (11).

Supposing $r_{ij} = r$, $1 \leqslant i < j \leqslant n$, if $k_1 = k_2 = ... = k_m = -1$ and $k_{m+1} = k_{m+2} = ... = k_n = 1$ we deduce

$$g_1(n; k_1, k_2, ..., k_n) = 1 + r \cdot (n \cdot (n-1)/2 - 2 \cdot m \cdot (n-m))$$

and hence for $0 \leqslant r \leqslant 2/n$ it follows (11).

Considering the variable w instead of $2^j \cdot g_1(B_1, B_2, ..., B_j, 0, ..., 0)$ the AG algorithm becomes

Algorithm AG1

Step 0. Inputs: n; r_{ij}, $1 \leqslant i < j \leqslant n$;

Step 1. $w \leftarrow 1$; $j \leftarrow 1$;

Generate a uniform (0,1] random variate U.

If $U < 0.5$ then $B_j \leftarrow -1$;

else $B_j \leftarrow 1$.

Step 2. $j \leftarrow j + 1$; If $j > n$ then go to step 4.

Step 3. $S \leftarrow B_1 \cdot r_{1j} + B_2 \cdot r_{2j} + ... + B_{j-1} \cdot r_{j-1,\,j}$.

Generate a uniform (0,1] random variate U.

If $2.U \cdot w \leqslant w - S$ then $B_j \leftarrow -1$;

else $B_j \leftarrow 1$;

$w \leftarrow w + B_j \cdot S$; Go to step 2.

Step 4. Print $(B_1, B_2, ..., B_n)$. STOP.

Remark 2. Imposing $\Pr(B_i = -1) = \Pr(B_i = 1) = \Pr(B_j = -1) = \Pr(B_j = 1) = 1/2$ and $\mathrm{Corr}(B_i, B_j) = r_{ij}$ we deduce (12).

Therefore the r.v. — s B_i and B_j are independent if and only if $r_{ij} = 0$.

For independent r.v.— s, B_1, B_2, B_3 ,..., B_n from algorithm AG1 we obtain:

Algorithm AG1I (independent r.v. — s)

Step 1. For $1 \leqslant j \leqslant n$ do:

Generate a uniform (0,1] random variate U.

If $U \leqslant 1/2$ then $B_j \leftarrow -1$

else $B_j \leftarrow 1$

Step 2. Print $(B_1, B_2, ..., B_n)$. STOP.

3.2. Linear case. We will generate the binary random vector $(B_1, B_2, ..., B_n)$ so that

$$\text{Corr}(B_i, B_{i+1}) = r_{i,i+1}, \ 1 \leqslant i \leqslant n-1. \tag{13}$$

Proposition 2. Let g_2 be the function

$$g_2(n; k_1, k_2, \ldots, k_n) = ((1 + k_1 k_2 r_{12}) \cdot (1 + k_2 k_3 r_{23}) \ldots (1 + k_{n-1} k_n r_{n-1,n}))/2^n \tag{14}$$

where

$$k_1, k_2, \ldots, k_n \in \{-1, 1\} \text{ and } r_{i,i+1} \in [-1, 1], \ 1 \leqslant i \leqslant n-1.$$

Then the random vector $(B_1, B_2, \ldots, B_n)$ having g_2 as p.d.f. satisfies the relations (7) and (13).

Proof. Considering the equality

$$g_2(n; k_1, k_2, \ldots, k_j, 0, \ldots, 0) = g_2(j; k_1, k_2, \ldots, k_j) \ ; \ 2 \leqslant j \leqslant n; \tag{15}$$

the proof of Proposition 2 is similarly as for Proposition 1.

Remark 3. From (14) and (15) it follows for every $2 \leqslant j \leqslant n$

$$g_2(n; B_1, B_2, \ldots, B_{j-1}, -1, 0, \ldots, 0)/g_2(n; B_1, B_2, \ldots, B_{j-1}, 0, \ldots, 0) =$$

$$= (1 - k_{j-1} \cdot r_{j-1,j})/2. \tag{16}$$

Using the formula (16) it results a simpler version of the algorithm AG:

Algorithm AG2 (linear case)

Step 1. Generate a uniform (0,1] random variate U.

If $U \leqslant 1/2$ then $B_1 \leftarrow -1$

else $B_1 \leftarrow 1$

Step 2. For $2 \leqslant j \leqslant n$ do:
Generate an independent uniform (0, 1] random variate U;

If $U \leqslant (1 - B_{j-1} \cdot r_{j-1,j})/2$ then $B_j \leftarrow -1$;

else $B_j \leftarrow 1$;

Step 3. Print $(B_1, B_2, B_3, \ldots, B_n)$. STOP.

3.3. Tree case. We will impose to have

$$\text{Corr}\ (B_1, B_i) = r_{1i}; \ -1 \leqslant r_{1i} \leqslant 1, \ 2 \leqslant i \leqslant n. \tag{17}$$

Proposition 3. If the p.d.f. g_3 of the binary random vector $(B_1, B_2, \ldots, B_n)$ has the form

$$g_3(n; k_1, k_2, \ldots, k_n) = ((1 + k_1 k_2 r_{12}) \cdot (1 + k_1 k_3 r_{13}) \ldots (1 + k_1 k_n r_{1n}))/2^n \tag{18}$$

where $k_1, k_2, \ldots, k_n \in \{-1, 1\}$ and $-1 \leqslant r_{1j} \leqslant 1$, $2 \leqslant j \leqslant n$, then the equalities (7) and (17) are fulfilled.

Proof. For every $k_1, k_2, \ldots, k_n \in \{-1, 1\}$ it results $0 \leqslant g_3(n; k_1, k_2, \ldots, k_n) \leqslant 1$ and besides we have the equality

$$g_3(n; k_1, k_2, \ldots, k_{j-1}, 0, k_{j+1}, \ldots, k_n) = g_3(n-1; k_1, k_2, \ldots, k_{j-1}, k_{j+1}, \ldots, k_n) \tag{19}$$

for any $2 \leqslant j \leqslant n$.

Applying successively formula (19) we finally obtain the relation

$$\Pr(B_1 = k_1,\ B_j = k_j) = g_3(2;\ k_1,\ k_j) = (1 + k_1 \cdot k_j \cdot r_{1j})/4; \qquad 2 \leqslant j \leqslant n,$$

which implies (7) and (17).

Remark 4. From the equalities (18) and (19) we obtain

$$g_3(n;\ B_1,\ B_2,\ ...,\ B_{j-1},\ -1,\ 0,\ ...,\ 0)/g_3(n;\ B_1,\ B_2,\ ...,\ B_{j-1},\ 0,\ ...,\ 0) =$$
$$= (1 - k_1 \cdot r_{1j})/2$$

for every $2 \leqslant j \leqslant n$ and hence the algorithm AG becomes:

Algorithm AG3 (tree case)

Step 1. Generate a uniform (0,1] random variate U.

If $U \leqslant 1/2$ then $B_1 \leftarrow -1$;

else $B_1 \leftarrow 1$;

Step 2. For $2 \leqslant j \leqslant n$ do:

Generate an independent uniform (0,1] random variate U;

If $U \leqslant (1 - B_1 \cdot r_{1j})/2$ then $B_j \leftarrow -1$;

else $B_j \leftarrow 1$;

Step 3. Print $(B_1,\ B_2,\ ...,\ B_n)$. STOP.

4. APPLICATIONS

The described algorithms were written in FORTRAN and were run on a Romanian FELIX C-256 computer. The subroutine RND from the RAVAGE subprogram library [6] was used for generating a uniform (0,1) random variate U.

4.1. Using the algorithm AG1 we generated 3000 random vectors $(B_1,\ B_2,\ ...,\ B_n)$ so that $r_{ij} = 1/n$, $1 \leqslant i < j \leqslant n$, where $2 \leqslant n \leqslant 10$. The computer and theoretical results were closed. So, in the case $n = 4$, the empiric mean and variance for $(B_1,\ B_2,\ ...,\ B_n)$ were $(0.011, -0.005, 0.023, -0.027)$, respectively $(1.0, 1.0, 0.999, 0.999)$. We also obtained the values $\hat{r}_{12} = 0.235$, $\hat{r}_{13} = 0.259$, $\hat{r}_{14} = 0.226$, $\hat{r}_{23} = 0.251$, $\hat{r}_{24} = 0.264$, $\hat{r}_{34} = 0.255$ as estimations for the correlations r_{ij}.

4.2. Let Z_i $1 \leqslant i \leqslant n$, be the r.v.—s defined by (6). Then the noncentral moment of order t, $M_t(Z)$, of the r.v. $Z = h(Z_1, Z_2, ..., Z_n)$ is given by

$$M_t(Z) = \sum_{k_1} \sum_{k_2} \cdots \sum_{k_n} h^t\,(c_1 + k_1 d_1,\ ...,\ c_n + k_n d_n) \cdot f_1(k_1,\ k_2,\ ...,\ k_n) \qquad (20)$$

where $f_1(k_1, k_2, k_3, ..., k_n)$ is the p.d.f. of $(B_1,\ B_2,\ ...,\ B_n)$, $k_i \in \{-1, 1\}$.

The formula (20) (for $n = 2,3$ and $t = 1,2$) was applied in [2] to find some safety coefficients in constructions.

But it is difficult to obtain the form of the p.d.f. $f_1(k_1, ..., k_n)$ when the number n of variables increases. However, supposing the independence of the r.v.—s $B_1, B_2, ..., B_n$ it follows $f_1(k_1, ..., k_n) = 1/2^n$.

4.3. Considering $f_1(k_1, k_2, ..., k_n) = g_1(n; k_1, k_2, ..., k_n)$ (formula (10)) as p.d.f. for the random vector $(B_1, B_2, ..., B_n)$ it results the following algorithm for computing $M_t(Z)$ (formula (20)) with $Z = h(c_1 + d_1 \cdot B_1, c_2 + d_2 \cdot B_2, ..., c_n + d_n \cdot B_n)$:

Algorithm MOMN (t) (noncentral moment of order t)

Step 0. Inputs: $n, t, h, c_1, d_1, c_2, d_2, ..., c_n, d_n; r_{ij}, 1 \leqslant i < j \leqslant n$.

Step 1. $M_t \leftarrow 0$; $k_i \leftarrow -1$, $1 \leqslant i \leqslant n$;

Step 2. $p \leftarrow 1 + \sum_{1 \leqslant i < j \leqslant n} k_i \cdot k_j \cdot r_{ij}$

Step 3. $Z_i \leftarrow c_i + k_i \cdot d_i$; $1 \leqslant i \leqslant n$.

$M_t \leftarrow M_t + p \cdot h^t (Z_1, Z_2, Z_3, ..., Z_n)$

Step 4. $s \leftarrow 1$

For $1 \leqslant j \leqslant n$ do:

If $s = 0$ then go to step 2;
If $k_j = 1$ then $k_j \leftarrow -1$;

else $k_j \leftarrow 1$; $s \leftarrow 0$;

Step 5. If $s = 0$ then go to step 2.

Step 6. $M_t \leftarrow M_t/2^n$; Print M_t; STOP.

Remark 5. We tested the MOMN(t) algorithm considering:
— a linear function h,

$$h(x_1, x_2, ..., x_n) = a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + ... + a_n \cdot x_n$$

with $a_0 = 7$, $a_j = 10 - j$, $1 \leqslant j \leqslant n$;

— a normal distribution for the r.v.—s X_j, $1 \leqslant j \leqslant n$, with mean $c_j = 20 + j$, variance $d_j = j$, and correlation $r_{ij} = 1, r_{ij} = 1/n, 1 \leqslant i < j \leqslant n$.

Then the r.v. $W = h(X_1 X_2, ..., X_n)$ has a normal distribution and besides

$$M_1(W) = a_0 + a_1 \cdot c_1 + a_2 \cdot c_2 + ... a_n \cdot c_n \tag{21}$$

$$M_2(W) = M_1(W^2) = (M_1(W))^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} a_i \cdot a_j \cdot d_i \cdot d_j \cdot r_{ij}.$$

Applying the algorithm MOMN(t) we computed the moments $M_t(Y)$ for the r.v. $Y = h(Z_1, Z_2, ..., Z_n)$, where the r.v.—s Z_i were given by (6), following (7) and (8).

We obtained the same 7 semnificative digits for $M_t(W)$ and $M_t(Y)$, with $t = 1, 2$ and $2 \leqslant n \leqslant 10$.

Algorithm MOMN(t) was used in practice, to estimate some stability coefficients in constructions, for n random correlation factors, $n \geqslant 5$ (see also [5]).

4.4. The algorithm MOMN(t) can be used especially to determine different statistic indicators for economic processes.

Practically this procedure can't be applied when the number n of variables increases.

For solving some estimation problems in large systems, with a lot of dependendent random variables (Voronov [7]), we propose a Monte Carlo simulation method based on the algorithm AMC.

REFERENCES

1. BRATLEY, P., FOX, B. L., SCHRAGE, L. E.: *A Guide to Simulation.* Springer Verlag, New York, 1983.
2. CAUWELAERT, van F., BOURDEAU, P. L., PERWEZ, A.: *Approximations ponctuelles des distributions de probabilité par la méthode de Rosenblueth; Application à l'estimation du risque de rupture d'un bajoyer.* Annales des travaux publiques de Belgiques, no. 5, 1985, p. 459—469.
3. HAHN, J. G.., SHAPIRO, SAMUEL, S.: *Statistical Models in Engineering.* John Wiley and Sons, New York, 1967. (Mir, Moskow, 1969).
4. KNUTH, E., DONALD: *The Art of Computer Programming.* Vol. 2, Seminumerical Algorithms Addison-Wesley, Reading Mass., 1973.
5. ŞTEFĂNESCU, POLIANA, STEFĂNESCU St., POPESCU, M.: *On Some Probabilistic Models with Applications in Constructions.* INFOTEC'88 Symposium, Bucharest, September, 1988, vol. 2, p. 44—47 (in Romanian).
6. VĂDUVA, I., STOICA, M., ODAGESCU, I.: *Simulation of Economic Processes.* Technical Publishing House, Bucharest, 1983 (in Romanian).
7. VORONOV, A. A.: *Management and Control in Large Systems.* MIR Publishers, Moskow, 1986.

UNE TECHNIQUE SIMPLE DE SIMULATION MONTE-CARLO APPLICABLE AUX GRANDS SYSTÉMES

L'on propose une procédure simple de simulation Monte Carlo pour estimer les moments des variables aleatoires qui sont fonctions aux paramètres aléatoires. Cette technique de simulation peut aussi être appliquée pour déterminer les différents indices statistiques des processus économiques et qui peut être facilement appliquée aux grands systèmes ayant une multitude de variables aléatoires dépendants.

Mots-clé: simulation, méthode Monte Carlo, variables aléatoires, systèmes de grandes dimensions.

ПРОСТАЯ ПРОЦЕДУРА MONTE-CARLO СИМУЛЯЦИИ С ПРИЛОЖЕНИЯМИ В КРУПНЫХ СИСТЕМАХ

Педлогается простая процедура симуляции типа для оценки моментов алеаторных перемен, являющихся функциями с алеаторными параметрами. Эта процедура симуляций также может быть примененапри определении различных статистических показателей для экономических процессов и вообщеона может быть легко использована в крупных системах с множеством зависимых алеаторных перемен.

Ключевые слова: симуляция, метод Monte-Carlo, алеаторные переменные, крупноразмерные систем.