ALGORYTMY

VOL. V • NO. 9 • 1968

INSTYTUT MASZYN MATEMATYCZNYCH

A L G O R Y T M Y

Vol. V N° 9 1968

P R A C E

Instytutu Maszyn Matematycznych

Adres redakcji: Warszawa, ul. Koszykowa 79, tel.28-37-29

TREŚĆ
CONTENTS

Teoria maszyn
Computer theory

NUMERICAL ANALYSIS

Instytut Maszyn Matematycznych
Algorytmy N° 9

EXPERIMENTAL FUNCTION DETERMINING THE OPTIMAL OVERRELAXATION COEFFICIENT FOR A CERTAIN BOUNDARY PROBLEM WITH AXIAL SYMMETRY

by Teofil PIWECKI
and Zdzisław SOKÓLSKI

Received January 6th, 1967

The purpose of the work is to shorten the computation by reducing the iteration number. This aim was endeavoured to be reached by means of finding certain regularities of overrelaxation coefficient changes and those of the integration area. This can be of important practical significance for engineer computation.

As it is seen from the results given in paper [3] simple relaxation, with the computation schematization required by digital computers, is rather of little use. Therefore, a necessity arises to look for a method of determining the optimal overrelaxation coefficient.

A way to determine the optimal value of overrelaxation coefficient based on experimental results for a problem formulated in [3] is presented below.

The work aimed at gaining a rule for determining the optimal overrelaxation coefficient for a rectangular area with an arbitrary number of node points and arbitrary side ratio. Evidently, the arbitrariness of both node points and side ratio of the rectangle should be conditioned by a physical sense of solution researching, i.e. one should exclude in advance extreme cases for

which side ratio of the rectangle or the number of nodes bend to zero, or take such values that the obtained solution loses its sense resulting from the assumption of the method of finite differences /linear change of function between succeeding nodes of the integration area/.

Areas to be investigated have the following number of node points: 24, 36, 64, 96, 144. Computations have been made for integration areas with side ratio given below in Table No 1.

Table No. 1

Number of node points MxN	Number of lines M	Number of columns N	M/N
24	3	8	3/8
	4	6	4/6
	6	4	6/4
	8	3	8/3
	12	2	12/2
36	3	12	3/12
	4	9	4/9
	6	6	6/6
	9	4	9/4
	12	3	12/3
	18	2	18/2
64	4	16	4/16
	8	8	8/8
	16	4	16/4
	32	2	32/2
96	3	32	3/32
	4	24	4/24
	6	16	6/16
	8	12	8/12
	12	8	12/8
	16	6	16/6
	24	4	24/4
	32	3	32/3
144	6	24	6/24
	9	16	9/16
	12	12	12/12
	16	9	16/9
	18	8	18/8
	24	6	24/6

It is seen from the above table that the investigations did not exhaust all the possibilities concerning the side ratio of the ac-

cepted rectangles. This resulted partly from program possibilities, and first of all from the necessity of limiting the counting time of a digital computer.

The results of investigations are given in figures 1, 2, 3, 4, 5 presenting the dependence between the number of iterations J around the whole area, which is needed to obtain the solution with the given exactness $\varepsilon = 10^{-3}$ and the overrelaxation coefficient H. The following values are also given in these figures.

H_o - optimal value of overrelaxation coefficient,
J_o - number of iterations for $H = H_o$,
J_4 - number of iterations for $H = 4$ /normal relaxation/,
J_4/J_o - ratio of iterations number with $H = 4$ to iterations number with $H = H_o$, giving numerically the profit resulting from the application of the overrelaxation process for appropriate ratio M/N.

In figures 3, 4, 5 the values J_4 and ratio J_4/J_o are not given, because of too long a time needed to obtain the solution with $H = 4$, especially for small values M/N.

It results from the investigations that for a given number of nodes M x N, therefore for a given area, there exists the dependence of the optimal value of the overrelaxation coefficient on the ratio of the area sides, thus, on the ratio M/N. Namely, the optimal overrelaxation coefficient grows together with the growth of the ratio M/N, taking the values from the interval:

$$2.0 < H_o < 4.0 \quad . \qquad /1/$$

The above limitation of the range of H_o results from the accepted interpretation of the overrelaxation process given in [3].

In order to determine the functional dependence of the value of the optimal overrelaxation coefficient on the side ratio of the integration area, the same method has been used as the one applied to elaborate the results of the experimental data.

The values n_0 given in figures 1 to 5 and the corresponding ratio M/N, presented in the logarithmic set, permit to admit that the dependence between the overrelaxation coefficient and the ratio M/N is linear, therefore in a normal set, the dependence $H = f\left(\frac{M}{N}\right)$ is of the power type /fig. 6/.

Taking for greater clearness of drawings

$$H_0 = z + 2.0 ,$$

thus

$$z = H_0 - 2.0 , \qquad /2/$$

we have

$$z = \alpha\left(\frac{M}{N}\right)^{\beta} . \qquad /3/$$

For separate M x N the coefficients α and β are determined by the method of least squares, obtaining

M x N	α	β
24	0.665	0.420
36	0.544	0.436
64	0.405	0.457
96	0.334	0.474
144	0.270	0.506

As it is seen, both coefficients α and β change, depending on the number of nodes M x N. Data from the above table put on the diagram in an logarithmic set are arranged according to fig. 7, approximately along straingh lines, which also shows the power character of the dependences

$$\alpha = \alpha_0 (M \times N)^{\alpha_1} ,$$
$$\beta = \beta_0 (M \times N)^{\beta_1} . \qquad /4/$$

When computing the coefficients by the method of the least squares, one obtains

$$\alpha = 3.280 (M \times N)^{-0.502} ,$$
$$\beta = 0.305 (M \times N)^{0.099} . \qquad /5/$$

Putting dependences /5/ to the formula /3/ and considering the formula /2/ one obtains the experimental function that permits to compute the value of the optimal overrelaxation coefficient for a rectangular area with a given number of nodes N x N and a given side ratio M/N.

$$H_o = 3.280(M\times N)^{-0.502}\left(\frac{M}{N}\right)^{0.305}(M\times N)^{0.099} + 2.0 . \qquad /6/$$

Accepting the coefficients α_1 and β_1 rounding, one has:

$$H_o = \frac{3.28}{\sqrt{M\times N}}\left(\frac{M}{N}\right)^{0.3}(M\times N)^{0.1} + 2.0 . \qquad /7/$$

In case M = N, therefore for squares, the formula /7/ takes a simple form

$$H_o = \frac{3.28}{M} + 2.0 . \qquad /8/$$

In order to check the generality of the above given formulae /6/, /7/, /8/ a series of computations has been performed for areas that are different from those, which are given in table 1, and which served as the basis to establish the above dependences. The results of these control computations are given in table 2.

In spite of the control computations being fragmentary /according to table 2/ it is evident that the overrelaxation coefficients computed from formulae /6/ and /7/ are either really optimal or near to optimal.

Using formulae /6/, /7/ or /8/ one can be sure that the computations are carried out in a way that permits to gain a solution in a time being at least close to the shortest one. This is of a great importance, particularly in computations for engineering purposes, when both the node number and the number of solutions needed for various side ratio are as a rule big, and when an inapropriately accepted overrelaxation coefficient can cause significant time losses.

Table No. 2

Number of node points M x N	M/N	H_o computed from /6/	H	J Number of iterations
4x4	4/4	2.82	2.70 2.76 2.78 2.80 2.81 2.82 2.83 2.84 2.85 2.86 2.90	20 16 15 15 14 14 14 14 15 16 17
6x8	6/18	2.41	2.30 2.36 2.38 2.40 2.41 2.42 2.43 2.44 2.45 2.47 2.50	38 32 30 26 26 27 26 28 32 33 37
4x12	4/12	2.28	2.22 2.23 2.24 2.25 2.26 2.27 2.28 2.29 2.30 2.31 2.32 2.33 2.34	54 50 42 46 45 41 39 41 39 43 44 47 54
20x20	20/20	2.16	2.13 2.16 2.20	76 60 73
8x50	8/50	2.06	2.03 2.06 2.10	351 193 200

We are aware that the empiric character of formula /6/ limits its validity closely to the conditions which it has been established to /boundary conditions, initial data, scheme of node run,

exactness/, what significantly lowers its value. Establishing of further factors influencing the optimal value of the overrelaxation coefficient will demand further tests and investigations.

References

[1] D.N. de G. ALLEN: Relaxation Methods. New York, 1959.

[2] L. COLLATZ: Metody numeryczne rozwiązywania równań różniczkowych. PWN Warszawa, 1960.

[3] Z. SOKÓLSKI, T. PIWECKI: Experimental Determination of Overrelaxation Coefficients for Laplace Equation in Cylinder Coordinates with Mixed Boundary Conditions. Algorytmy No 6, 1966.

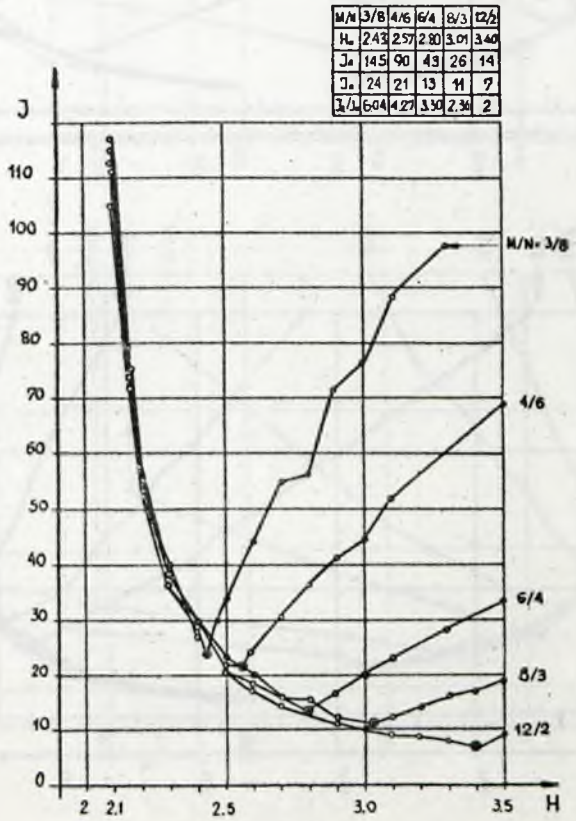


Fig. 1. J = f(H) for MxN = 24

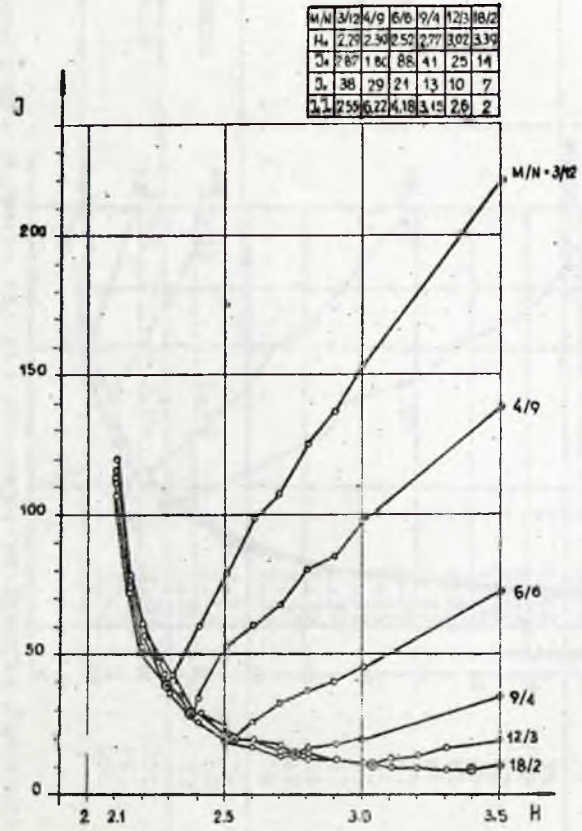


Fig.2. J = f(H) for MxN = 36

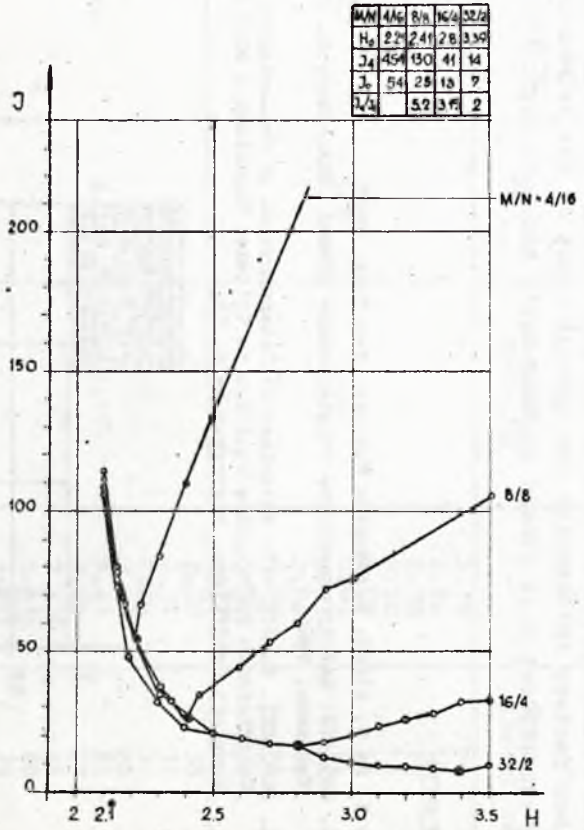
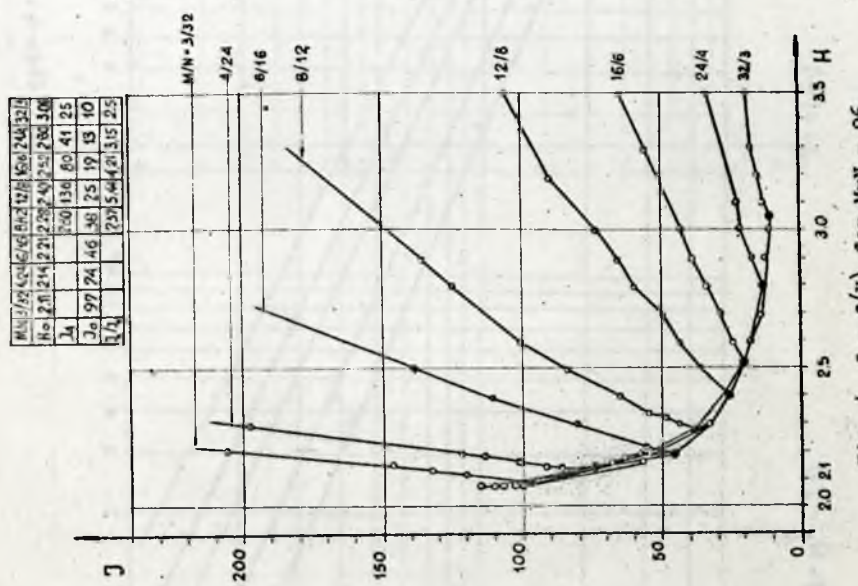
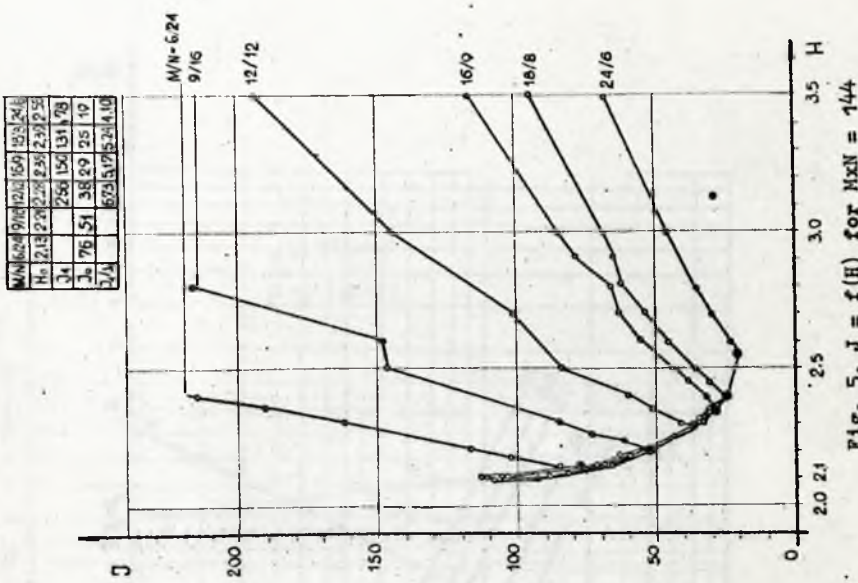


Fig. 3. J = f(H) for MxN = 64



M/N	3/32	4/24	6/16	8/12	12/8	16/6	24/4	32/3
H_0	2.11	2.14	2.21	2.28	2.40	2.58	2.80	3.06
J_1				130	136	80	41	25
J_∞	97	74	46	48	25	19	13	10
J/J_1				2.9	5.4	4.2	3.1	2.5

Fig. 4. J = f(H) for MxN = 96



M/N	6.24	9/16	12/12	16/9	18/8	24/6
H_0	2.13	2.26	2.36	2.36	2.39	2.55
J_1				256	156	131
J_∞	76	54	38	29	25	19
J/J_1				8.9	6.3	6.9

Fig. 5. J = f(H) for MxN = 144

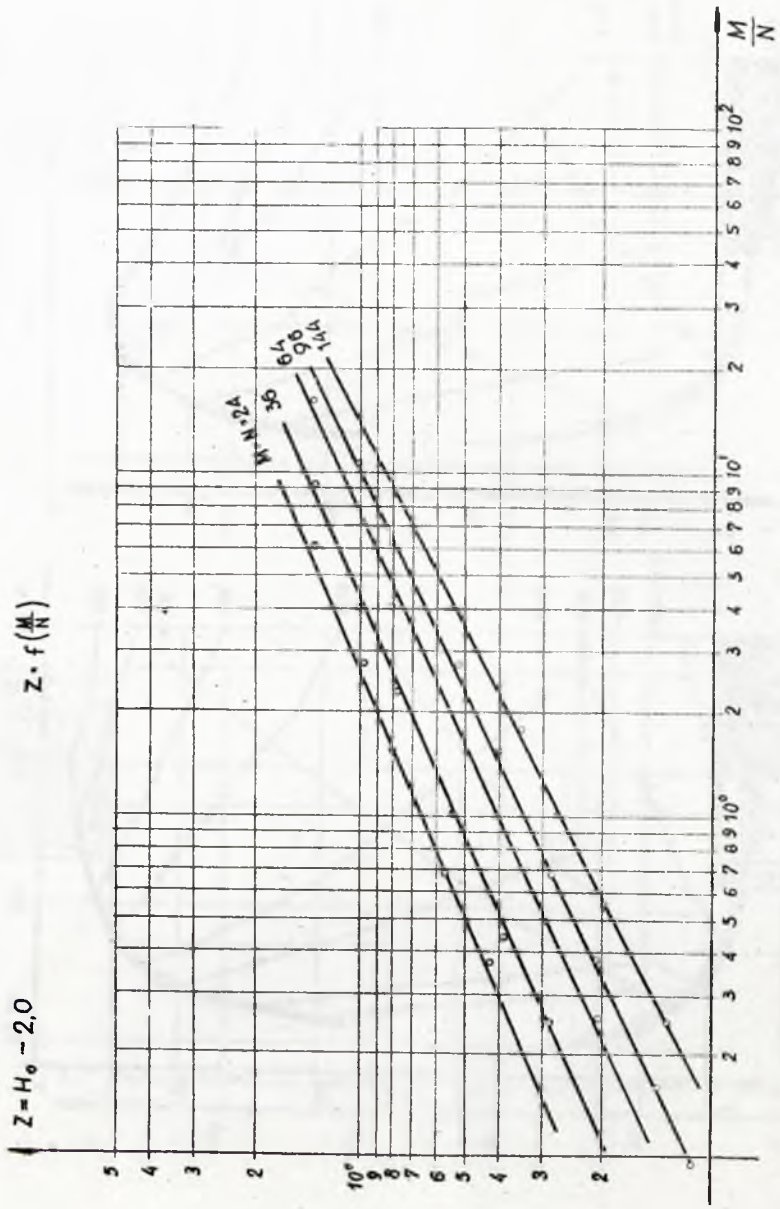


Fig. 6

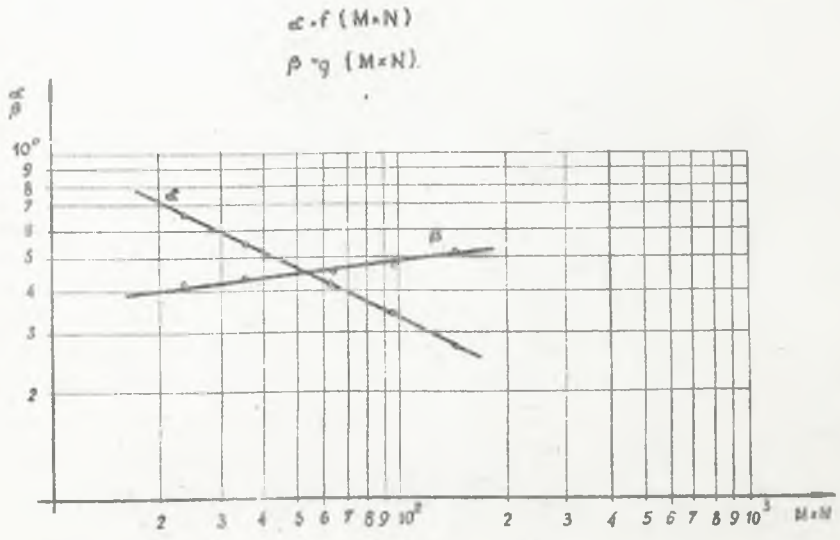


Fig. 7

STATISTICAL APPLICATION

Instytut Maszyn Matematycznych
Algorytmy N° 9

ON THE EFFICIENCY OF MONTE CARLO INTEGRATION THE MONOMIAL $c_{n,k}x_1^n \ldots x_k^n$

by Ryszard Zieliński

Received March 4th 1967

The error $R_{n,k}$ of integration monomial $f(x) = c_{n,k}x_1^n \ldots x_k^n$ over k-dimensional unit cube by the trapeizodal rule is given by /4/. The error $a\sigma_{n,k}(\hat{I})$ of integration f(x) by Monte Carlo method with uniform distribution on the unit cube is defined by /6/. The set $M_a = \{(n,k) : a\sigma_{n,k}(\hat{I}) < R_{n,k}\}$ is called the set of a preference of the Monte Carlo method. In the paper some theorems concerning the set M_a are formulated and then the method of construction of the set is described. An example is given /see fig. 1/.

INTRODUCTION

If the integral

$$I = \int_{\Omega} \ldots \int f(x_1, \ldots, x_k)\, dx_1 \ldots dx_k \qquad /1/$$

is to be evaluated by the Monte Carlo method, the following procedure is usually applied: let us consider a probability space $(X, \mathfrak{X}, P)$, $\Omega \in \mathfrak{X}$ and $P(\Omega) > 0$. In the simplest case X is equal to Ω and P is the uniform distribution, i.e. if μ is a finite measure on $(X, \mathfrak{X})$, then for each A

$$P(A) = \mu(A)/\mu(X).$$

Usually X is a bounded set in k-dimensional Euclidean space and μ is a Lebesgue measure. Then a random variable $\hat{I}$ is defined in such a way, that $E(\hat{I}) = I$ and $E(\hat{I}^2) < \infty$. Finally, it is assumed that integral I is equal to a value of $\hat{I}$ found in a numerical experiment.

The efficiency of such a procedure will be discussed in the paper.

The problem of the comparison of methods of integration in the class of Monte Carlo methods /i.e. in the class of stochastic algorithms/ has been considered by many authors. For example Zubrzycki [5] studied methods of integration with respect to the value of the variance $D^2(\hat{I})$. Hammersley and Handscomb [3] took the following expression as a measure of the efficiency of method A relative to method B

$$\frac{n_B \, D_B^2(\hat{I})}{n_A \, D_A^2(\hat{I})},$$

where $D_B^2(I)/D_A^2(I)$ is the variance ratio and n_B/n_A is the labour ratio /n_A and n_B are often taken as the number of times of the element of integration being evaluated in each method/. Some results are also known of optimization of the statistical experiment concerning the estimation of integrals, for example the problem of optimum allocation of sampling units [2].

The problem of comparison of the methods of integration in the class of all algorithms /i.e. the class containing deterministic algorithms and stochastic ones/ is more complicated. It is difficult to construct such a measure of efficiency that can be applied to a stochastic algorithm as well as to a deterministic one. In [1] and [4] there are some remarks on the subject in case of seeking extremum.

We shall further discuss the comparison of a simple Monte Carlo method and a simple deterministic method for the evaluation of the integral of the monomial $c_{n,k} x_1^n \ldots x_k^n$ over the k-dimensional unit cube, where $c_{n,k}$ is chosen in such a way, that

$$c_{n,k} \int_0^1 \cdots \int_0^1 x_1^n \cdots x_k^n \, dx_1 \cdots dx_k = 1 \qquad /2/$$

for each n and k, i.e. $c_{n,k} = (n+1)^k$. We shall distinguish a subset of the preference of the Monte Carlo method in the set of all pairs (n,k), $n = 1,2, \ldots$, $k = 1,2, \ldots$.

THE PROBLEM

Let T be the trapezoidal rule or its multidimensional extension. Let $\tilde{I}_{n,k}$ be the value of the integral (1) evaluated by method T with 2^k points of the type $(x_1, \ldots, x_k)$, $x_i = 0$ or 1; then $\tilde{I}_{n,k}$ is equal to the mean of the element of integration over all those points. We have:

$$\tilde{I}_{n,k} = \left(\frac{n+1}{2}\right)^k . \qquad /3/$$

We define $R_{n,k}$:

$$R_{n,k} = |\tilde{I}_{n,k} - I| .$$

Then $R_{n,k}$ is the error of integration by the method T. We have:

$$R_{n,k} = \left(\frac{n+1}{2}\right)^k - 1 . \qquad /4/$$

Let M be the Monte Carlo method of evaluating the integral (1) with probability space $(X, \mathcal{X}, P)$, where X is a k-dimensional unit cube and P is uniform distribution. Let $\hat{I}$ be

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} f_i , \qquad /5/$$

where f_i is a value of the monomial $c_{n,k} x_1^n \ldots x_k^n$ for the

i-th $(i = 1,2, \ldots, N)$ point chosen at random acoording to the distribution P. We assume, that $N = 2^k$, i.e. that the number of times that the element of integration is evaluated is the same in both methods.

Let a be such a positive number that for a fixed value α $(0 < \alpha < 1)$ we have:

$$P\left\{|\hat{I} - I| < a\sigma_{n,k}(\hat{I})\right\} = 1 - \alpha, \qquad /6/$$

where $\sigma^2_{n,k}(I)$ is the variance of the estimator $\hat{I}$.

We have:

$$\sigma^2_{n,k}(I) = \frac{1}{N^2}\sum_{i=1}^{N}\sigma^2_{n,k}(f_i) = N^{-1}\left\{\int_0^1 \ldots \int_0^1 f^2 dx_1 \ldots dx_k - I^2\right\},$$

where $f = c_{n,k}x_1^n \ldots x_k^n$ and $N = 2^k$; hence

$$\sigma^2_{n,k}(\hat{I}) = \left[\frac{(n+1)^2}{2(2n+1)}\right]^k - \left(\frac{1}{2}\right)^k \qquad /7/$$

If α is small enough, the event in brackets on the lefthand side of (6) is "practically certain"; then

$$a\sigma_{n,k}(\hat{I})$$

will be called the error of the method M.

We shall say that the method M is more effective than the method T, if the error of M is smaller than the error of T, i.e. if:

$$a\sigma_{n,k}(\hat{I}) < R_{n,k}.$$

The set

$$M_a = \left\{(n, k): a\sigma_{n,k}(\hat{I}) < R_{n,k}\right\} \qquad /8/$$

will be called the set of a preference of the method M. Further we shall show how the set M_a can be effectively constructed.

FUNDAMENTAL THEOREMS CONCERNING THE SET M_a

Let $N = \{(n,k): n = 1,2, \ldots, k = 1,2, \ldots\}$ and let $M_a^0 = N - M_a$.

Theorem 1. The points $(1,k)$, $k = 1,2, \ldots$, belong to M_a^0.

Proof: it follows immediately from the fact that $R_{1,k} = 0$ for each k, $k = 1,2, \ldots$.

Theorem 2. If $(n,k) \in M_a$, then $(n+1,k) \in M_a$ for any $k \geq 1$ and $n > 1$.

Proof: because $a\sigma_{n,k} > 0$ and $R_{n,k} > 0$, we can write:

$$M_a = \left\{(n,k): a^2 \sigma_{n,k}^2 < R_{n,k}^2\right\} .$$

Then, according to /4/ and /7/, we have:

$$M_a = \left\{(n,k): 2^k a^2 \left[(n+1)^{2k} - (2n+1)^k\right] < \left[(n+1)^k - 2^k\right]^2 (2n+1)^k\right\} .$$

Let us consider the function:

$$\varphi_k(x) = 2^k a^2 \left[\left(\frac{x+1}{2}\right)^{2k} - x^k\right] - \left[\left(\frac{x+1}{2}\right)^k - 2^k\right]^2 x^k .$$

It is easy to see, that:

$$M_a = \left\{(n,k): \varphi_k(2n+1) < 0\right\} . \qquad /9/$$

The function $\varphi_k(x)$ is a polynomial of the variable x. Ordering $\varphi_k(x)$ with respect to the power of x we have:

$$\varphi_k(x) = -2^{-2k} \sum_{i=2k+1}^{3k} \binom{2k}{i-k} x^i + \sum_{i=k+1}^{2k} A_i x^i + B_k x^{k+2^{-k}a^2} \sum_{i=0}^{k-1} \binom{2k}{i} x^i ,$$

where:

$$A_1 = 2^{-k}a^2\binom{2k}{1} - 2^{-2k}\binom{2k}{1-k} + 2\binom{k}{1-k},$$

$$B_k = 2^{-k}a^2\binom{2k}{k} - 2^k a^2 - 2^{-2k} + 2 - 2^{2k}.$$

We have

$$A_1 > 2^{-k}a^2\binom{2k}{1} - 1 + 2\binom{k}{1-k} = 2^{-k}a^2\binom{2k}{1} + 2\left[\binom{k}{1-k} - \frac{1}{2}\right] > 0,$$

$$B_k < 0.$$

Hence, the number of changes of the signs in the series of coefficients of $\varphi_k(x)$ is equal to 3. Then, /according to the theorem of Desoartes/, the polynomial $\varphi_k(x)$ has 3 or 1 real zeros. It is easy to see that

$$\varphi_k(0) = 2^{-k}a^2 > 0,$$

$$\varphi_k(1) = -(1 - 2^k)^2 > 0,$$

$$\varphi_k(3) = 2^k a^2(4^k - 3^k) > 0.$$

Then, for $x > 3$ there is one zero of the $\varphi_k(x)$. If for some $x > 3$ /i.e. for some $n > 1$/ we have $\varphi_k(x) < 0$ /i.e. $a_{6_{n,k}} < R_{n,k}$/, then, this inequality holds for any $x' > x$ /i.e. for any $n' > n$/. But then it follows from $(n,k) \in M_a$ that $(n+1,k) \in M_a$, QED.

Corollary. It follows from the proof of theorem 2 that for any k there exists a unique $n_k > 1$ such that $(n,k) \in M_a$ if and only if $n > n_k$.

Note. The method used in the proof of theorem 2 can be applied for the construction of the set of the preference of the method M for any k in case if n is real /not necessarily natural/.

Theorem 3. If $(n,k) \in M_a$ then $(n,k+1) \in M_a$ for $n \geq 2$.

Proof: we can write the theorem as follows:

$$(a\sigma_{n,k} < R_{n,k}) \Rightarrow (a\sigma_{n,k+1} < R_{n,k+1}),$$

or:

$$\left(a < \frac{R_{n,k}}{\sigma_{n,k}}\right) \Rightarrow \left(a < \frac{R_{n,k+1}}{\sigma_{n,k+1}}\right).$$

To prove the theorem it is sufficient to show that for any $n \geq 2$ we have:

$$\frac{R_{n,k}}{\sigma_{n,k}} \leq \frac{R_{n,k+1}}{\sigma_{n,k+1}},$$

or:

$$\frac{\sigma_{n,k+1}}{\sigma_{n,k}} \leq \frac{R_{n,k+1}}{R_{n,k}}. \qquad /10/$$

Let $s = (2n+2)/(2n+1)$, $r = (n+1)/2$ and $h(x) = (x^{k+1}-1)/(x^k-1)$ Then, the inequality /10/ can be writtem as follows:

$$\frac{1}{2} h(sr) \leq h^2(r). \qquad /11/$$

If $n \geq 2$, then $r > s > 1$. The funotion $h(x)$ is nondecreasing for $x > 1$, hence, $h(s) \leq h(r)$. For $x > 1$ we have $h(x) > 1$ and multiplying the last inequality by $h(r)$ we get $h(s)h(r) \leq h^2(r)$. It can be shown that in the considered interval we have $h(sr) < h(s)h(r)$ and hence /11/, QED.

Theorem 4. If $0 < a < b$, then $M_a \supset M_b$.

Proof: the theorem follows immediately from the very definition of the set M_a.

THE CONSTRUCTION OF THE SET M_a. EXAMPLE

We can construot the set M_a as follows:

1. all points $(1,k)$ belong to the set M_a /it follows from the theorem 1/;
2. we find k_2 such that $(2,k)$ belongs to M_a if and only if $k \geqslant k_2$ /according to theorem 2 such k_2 exists/. All the points (n,k), $n \geqslant 2$, $k \geqslant k_2$ we assign to M_a;
3. we repeat that construction for $k_3, k_4, \ldots,$ until for some m we get $k_m = 1$. It follows from the Corollary that such a number m exists. It ends the construction of M_a.

The set M_3 constructed by the above described methods is shown in the following figure.

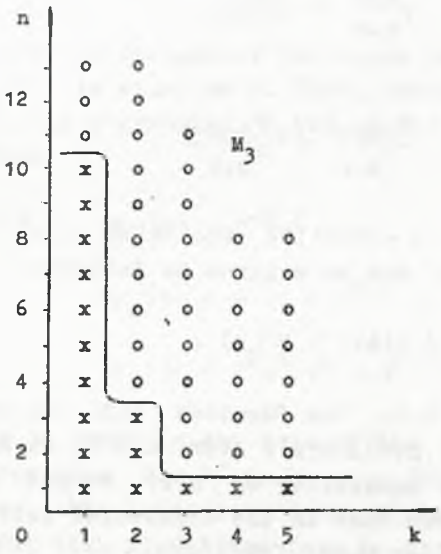


Fig. 1. The set M_3, o - points belong to M_3, x - points belong to M_3^c.

References

[1] S.H. BROOKS: A Comparison of Maximum-seeking Methods. OR, 1959, Vol.1, No. 4.

[2] I.L. FOLKS, Ch.E. ANTLE: Optimum Allocation of Sampling Units to Strata when there Are R Responses of Interest. Journal of Am.Stat.Ass., No 3, 1965.

[3] I.M. HAMMERSLEY, D.C. HANDSCOMB: Monte Carlo Methods. London: Methuen and Co. Ltd, New York, John Wiley and Sons, Inc. 1964.

[4] Л.А. Растригин: Сравнение методов Гаусса, Монте-Карло и случайного поиска при решении систем линейных алгебраических уравнений. Автоматика и вычислительная техника, № 7, Из-во АН Латвийской СССР, Рига 1964.

[5] S. ZUBRZYCKI: Remarks on Random, Stratified and Systematic Sampling in a Plane. Coll. Math., Vol. VI, 1958.

Instytut Maszyn Matematycznych
Algorytmy N° 9

A METHOD OF REALISATION OF MARKOV CHAINS

by Józef WINKOWSKI

Received September 15th, 1967

The need of realization of Markov chains appears sometimes when applying the Monte Carlo Techniques. On the other hand, there exist methods of realization of the sequences of independent random variables. The question arises how to realize the Markov chains using such sequences. In this paper a solution of the problem is given in the case of homogeneous chains with a discrete parameter and finite set of states.

1. PROBABILITIES ON SEMIGROUPS OF MAPPINGS OF A FINITE SET

Let $X=\{1,\dots,n\}$. Let $\mathcal{T}_n$ be the semigroup of all mappings of X into X, G – one of its subsemigroups and $M(G)$ – the set of all probability distributions on G; $\mu(f)$ will denote the value of the distribution μ for the element f and $N(\mu)$ – the support of this distribution i.e. the set of these f for which $\mu(f)>0$.

$M(G)$ is the semigroup with the operation which assigns to the distributions μ, ν their convolution $\mu*\nu$ i.e. the distribution defined by the formula

$$\mu*\nu(f)=\sum_{gh=f}\mu(g)\nu(h), \qquad /1/$$

where $g,h\in G$ and $gh(x)=g(h(x))$. For this operation

$$N(\mu*\nu)=N(\mu)N(\nu). \qquad /2/$$

Namely, if $f\in N(\mu*\nu)$ then $\mu*\nu(f)>0$. There exist $g,h\in G$ such

that $gh=f$ and $\mu(g)>0$, $\nu(h)>0$ i.e. $g\in N(\mu)$, $h\in N(\nu)$. Conversely, if $g\in N(\mu)$, $h\in N(\nu)$ then $\mu(g)>0$, $\nu(h)>0$ so that $\mu*\nu(gh)>0$ and $gh\in N(\mu*\nu)$.

The semigroup $M(G)$ can be monomorphically mapped into $M_n = M(\mathcal{I}_n)$. The monomorphism is realized by mapping φ_G, where $\varphi_G(\mu)(f) = \mu(f)$ for $f\in G$ and $\varphi_G(\mu)(f) = 0$ in the remaining cases. Indeed, $\varphi_G(\mu*\nu)(f)\neq 0$ only for $f\in G$ acoording to /2/. Thus, if $f\in G$,

$$\varphi_G(\mu*\nu)(f) = \mu*\nu(f) = \sum_{gh=f} \mu(g)\,\nu(h),$$

where $g,h\in G$. The distributions μ, ν can be replaced by $\varphi_G(\mu)$ and $\varphi_G(\nu)$ in this sum and it can be extended to $g,h\in\mathcal{I}_n$, because of the vanishing of new components, if any. Hence, $\varphi_G(\mu*\nu) = \varphi_G(\mu)*\varphi_G(\nu)$ and φ_G is monomorphism because it is one-one. These distributions from M_n are the elements of $\varphi_G(M(G))$, the support of which is contained in G.

To every $\mu\in M_n$ one can assign a $n\times n$ -matrix $\Phi(\mu)$ according to the formula

$$\Phi_{xy}(\mu) = \sum_{f(x)=y} \mu(f), \qquad x,y\in X. \qquad /3/$$

This matrix is stochastic. Indeed, for every $x\in X$ we have

$$\sum_{y\in X} \Phi_{xy}(\mu) = \sum_{y\in X} \sum_{f(x)=y} \mu(f) = \sum_{f} \mu(f) = 1.$$

Therefore, Φ is a mapping of the semigroup M_n into the set P_n of all stochastic $n\times n$ -matrices. P_n is a semigroup with the operation of matrix multiplication.

Lemma 1. *Φ is antihomomorphism of M_n into P_n.*

Proof. Let $\mu,\nu\in M_n$. For arbitrary $x,y\in X$ we have

$$\Phi_{xy}(\mu * \nu) = \sum_{f(x)=y} \mu * \nu(f) = \sum_{f(x)=y} \sum_{gh=f} \mu(g)\nu(h) =$$

$$= \sum_{z \in X} \sum_{h(x)=z} \sum_{g(z)=y} \mu(g)\nu(h) = \sum_{z \in X} \Phi_{xz}(\nu)\Phi_{zy}(\mu),$$

and, therefore, $\Phi(\mu * \nu) = \Phi(\nu)\Phi(\mu)$. QED

Lemma 2. *Φ maps M_n onto P_n . For every matrix $Q \in P_n$ there exists a distribution $\mu_Q \in M_n$ such that $\Phi(\mu_Q) = Q$. If Q has k nonvanishing elements then μ_Q can be chosen with at most k elements in support.*

Proof. Let $R^{(0)} = Q$. The elements $r^{(0)}_{xy}$ of $R^{(0)}$ are nonnegative and their sums in rows are 1. Therefore, there exist $y_1, \ldots, y_n$ such that $r^{(0)}_{1y_1} \ldots r^{(0)}_{ny_n} \neq 0$. Let $P^{(1)}$ be the matrix with elements

$$P^{(1)}_{xy} = \begin{cases} 1 & \text{for } y = y_x \\ 0 & \text{for } y \neq y_x \end{cases}$$

and let $\alpha_1 = \min \{ r^{(0)}_{1y_1}, \ldots, r^{(0)}_{ny_n} \}$. The matrix $R^{(1)} = R^{(0)} - \alpha_1 P^{(1)}$ has more vanishing elements than $R^{(0)}$. If there exist nonvanishing elements they are positive. Then all sums of elements of rows are $1 - \alpha_1 > 0$. This results from the properties of $R^{(0)}$ and $P^{(1)}$. Using to $R^{(1)}$ the same procedure as to $R^{(0)}$ we find $\alpha_2 > 0$, matrix $P^{(2)}$ with the structure similar to $P^{(1)}$, $R^{(2)} = R^{(1)} - \alpha_2 P^{(2)}$ etc. For a certain $m \leq k$ we obtain $R^{(m)} = 0$ because the number of vanishing elements increases in every step. Hence

$$Q = \alpha_1 P^{(1)} + \ldots + \alpha_m P^{(m)}, \qquad /4/$$

where $\alpha_1, \ldots, \alpha_m > 0$, each of the matrices $P^{(1)}, \ldots, P^{(m)}$ has exactly one unity in every row and zeros in the remaining places and thus $\alpha_1 + \ldots + \alpha_m = 1$.

For $i = 1, \ldots, m$ we assign the mapping $f_i \in \mathcal{T}_n$ to the matrix P^i, assuming $f_i(x) = y$ if $P^{(i)}_{xy} = 1$. Then we determine the distribution μ_Q on $\mathcal{T}_n$ assuming $\mu_Q(f_i) = \alpha_i$ and $\mu_Q(f) = 0$ for other f. This distribution has the support $N(\mu_Q) = \{f_1, \ldots, f_m\}$ with at most k elements. For arbitrary $x, y \in X$ we have

$$\Phi_{xy}(\mu_Q) = \sum_{f(x)=y} \mu_Q(f) = \sum_{f(x)=y} \sum_{f \in N(\mu_Q)} \mu_Q(f) = \sum_{i=1}^{m} \alpha_i P^{(i)}_{xy} = Q_{xy}$$

i.e. $\Phi(\mu_Q) = Q$. QED.

The procedure which was used in the proof enables us to determine effectively μ_Q for every matrix $Q \in P_n$.

L e m m a 3: For every matrix $Q \in P_n$ with k nonvanishing elements there exist:

- a semigroup $G_Q \subset \mathcal{T}_n$,
- antihomomorphism $\phi : M(G_Q) \to P_n$,
- distribution $\lambda_Q \in M(G_Q)$ with the support $N(\lambda_Q)$ which has at most k elements and generates G_Q,

such that $\phi(\lambda_Q) = Q$.

P r o o f. It is sufficient to take the semigroup generated by $M(\mu_Q)$ as G_Q and to assume $\lambda_Q = \varphi^{-1}_{G_Q}(\mu_Q)$, $\phi = \Phi \varphi_{G_Q}$. QED.

If, for natural i, λ_Q^{i*} denotes the i-th convolution power of λ_Q then, of course, $\phi(\lambda_Q^{i*}) = Q^i$.

When Q is bistochastic then, due to the Birkhoff – von Neumann Theorem /see [2], Th. 11.10/, there exists a decomposition similar to /4/ with the matrices $P^{(1)}, \ldots, P^{(m)}$ which have exactly one unity in every row and exactly one in every column. In this case permutations correspond to the matrices $P^{(1)}, \ldots, P^{(m)}$ and, therefore, G_Q is a group.

2. A REPRESENTATION OF THE HOMOGENEOUS MARKOV CHAIN WITH DISCRETE PARAMETER AND FINITE SET OF STATES

Let $\{\xi_t\}_{t=0,1,\dots}$ be a homogeneous Markov chain with the set $X = \{1, \dots, n\}$ of states. We shall assume that it is defined on the probability space $(\Omega, \mathcal{F}, P)$. Let $Q \in P_n$ be the matrix of transition probabilities of this chain and let G_Q be a subsemigroup as mentioned in Lemma 3. The following theorem on representation holds

T h e o r e m. There exist the probability space $(\bar\Omega, \bar{\mathcal{F}}, \bar P)$, random elements $\gamma_1, \gamma_2, \dots$ of G_Q defined on $(\bar\Omega, \bar{\mathcal{F}}, \bar P)$ and random element ξ of X defined on $(\bar\Omega, \bar{\mathcal{F}}, \bar P)$, such that

/I/ $\xi, \gamma_1, \gamma_2, \dots$ are independent,

/II/ $\gamma_1, \gamma_2, \dots$ have the common distribution λ_Q,

/III/ for arbitrary $t_1 < \dots < t_s$ and $x_1, \dots, x_s$

$$P\{\xi_{t_1} = x_1, \dots, \xi_{t_s} = x_s\} = \bar P\{\eta_{t_1} = x_1, \dots, \eta_{t_s} = x_s\},$$

where $\eta_1 = \xi$, $\eta_t = \gamma_t \cdots \gamma_1(\xi)$ for $t = 1, 2, \dots$.

P r o o f. Let $\Omega' = X$, $\mathcal{F}' = 2^X$. The measure P' determined on $\mathcal{F}'$ by the formula

$$P'(A) = P\{\xi_0 \in A\}$$

corresponds to the chain $\{\xi_t\}_{t=0,1,\dots}$. This measure represents the probability of the initial state.

We define on $(\Omega', \mathcal{F}', P')$ the random element η' of X assuming $\eta'(\omega') = \omega'$.

For $t = 1, 2, \dots$ let $G_t = G_Q$, $\mathcal{F}_t = 2^{G_Q}$ and let P_t be the probability measure on $\mathcal{F}_t$ corresponding to the distribution λ_Q. Let $\Omega'' = \prod_{t=1,2,\dots} G_t$, $\mathcal{F}'' = \prod_{t=1,2,\dots} \mathcal{F}_t$, $P'' = \prod_{t=1,2,\dots} P_t$ and let ω''_t denote the t-th coordinate od $\omega'' \in \Omega''$. We define on $(\Omega'', \mathcal{F}'', P'')$ the random elements $\gamma'_1, \gamma'_2, \dots$ of the semigroup G_Q assuming

$$\gamma'_t(\omega'') = \omega''_t$$

All the elements have the distribution λ_Q and are independent. Therefore, the element $\delta_{v,t} = \gamma'_t \cdots \gamma'_{v+1}$ / $v < t$ / has the distribution $\lambda_Q^{(t-v)*}$.

Let $\bar{\Omega} = \Omega' \times \Omega''$, $\bar{\mathcal{F}} = \mathcal{F}' \times \mathcal{F}''$, $\bar{P} = P' \times P''$ and let

$$\xi(\bar{\omega}) = \eta'(\omega'), \quad \gamma_t(\bar{\omega}) = \gamma'_t(\omega'')$$

for $\bar{\omega} \in \bar{\Omega}$, $t = 1,2,\ldots$, where $\omega' \in \Omega'$, $\omega'' \in \Omega''$, $(\omega', \omega'') = \bar{\omega}$. Then

$$\xi(\bar{\omega}) = \omega', \quad \gamma_t(\bar{\omega}) = \omega''_t.$$

Simultaneously ξ, γ_1, γ_2, ... are independent and $\gamma_1, \gamma_2, \ldots$ have the same distribution λ_Q.

Let us consider the set

$$F = \{\bar{\omega} \in \bar{\Omega} / \eta_{\tau_0}(\bar{\omega}) = y_0, \eta_{\tau_1}(\bar{\omega}) = y_1, \ldots, \eta_{\tau_r}(\bar{\omega}) = y_r\},$$

where $0 = \tau_0 < \tau_1 < \ldots < \tau_r$. We have

$$F = \{(\omega', \omega'') \in \Omega' \times \Omega'' / \omega' = y_0, \delta_{0,\tau_1}(\omega'')[\omega'] = y_1, \ldots, \delta_{0,\tau_r}(\omega'')[\omega'] = y_r\}.$$

If $F_{\omega'}$ denotes the section of this set that corresponds to the given $\omega' \in \Omega'$ then for $\omega' = y_0$

$$F_{\omega'} = \{\omega'' \in \Omega'' / \delta_{0,\tau_1}(\omega'')[y_0] = y_1, \ldots, \delta_{0,\tau_r}(\omega'')[y_0] = y_r\},$$

and $F_{\omega'}$ is empty for the remaining ω'. Hence,

$$\bar{P}(F) = P'\{\omega' = y_0\} \cdot P''(F_{y_0}) = P\{\xi_0 = y_0\} P''(F_{y_0}).$$

Because of the independency of $\gamma'_1, \gamma'_2, \ldots$ we get

$$P''(Fy_0) = P''\{\delta_{0,\tau_1}(\omega^*)[y_0] = y_1, \dots, \delta_{\tau_{r-1},\tau_r}(\omega^*)[y_{r-1}] = y_r\} =$$

$$= P''\{\delta_{0,\tau_1}(\omega^*)[y_0] = y_1\} \dots P''\{\delta_{\tau_{r-1},\tau_r}(\omega^*)[y_{r-1}] = y_r\} =$$

$$= \varphi_{y_0 y_1}(\lambda_Q^{\tau_1 *}) \dots \varphi_{y_{r-1} y_r}(\lambda_Q^{(\tau_r - \tau_{r-1})*})$$

Therefore, in view of Lemma 3,

$$P''(Fy_0) = q_{y_0 y_1}^{(\tau_1)} \dots q_{y_{r-1} y_r}^{(\tau_r - \tau_{r-1})},$$

where $q_{xy}^{(i)}$ denotes an element of the matrix Q^i.

Finally

$$P(F) = P\{\xi_{\tau_0} = y_0, \xi_{\tau_1} = y_1, \dots \xi_{\tau_r} = y_r\}$$

If $t_1 = 0$ in /III/ it is sufficient to assume $r = s-1$, $\tau_0 = t_1, \dots, \tau_r = t_s$, $y_0 = x_1, \dots, y_r = x_s$ for obtaining /III/. If $t_1 > 0$ then assuming $r = s$, $\tau_0 = 0$, $\tau_1 = t_1, \dots, \tau_r = t_s$, $y_1 = x_1, \dots, y_1 = x_s$ we get

$$\bar{P}\{\eta_{t_1} = x_1, \dots, \eta_{t_s} = x_s\} = \sum_{y_0 \in X} \bar{P}(F) = P\{\xi_{t_1} = x_1, \dots, \xi_{t_s} = x_s\}.$$

QED.

It results from the proved theorem that, from the view-point of an observer who is able to observe the phenomena taking place in the set of states only, the processes $\{\xi_t\}_{t=0,1,\dots}$, $\{\eta_t\}_{t=0,1,\dots}$ are not distinguishable. Of course, it is a weaker relation than a stochastic equivalence of both processes, because each of them is defined on a different probability space. The process $\{\eta_t\}_{t=0,1,\dots}$ can be realized by independing sampling of ξ, $\gamma_1, \gamma_2, \dots$ values and by applying in turn mappings $\gamma_1, \gamma_2, \dots$ to ξ.

References

[1] GRENANDER U.: Probabilities on Algebraic Structures. New York 1963.
[2] BERGE C.: Théorie des graphes et ses applications. Paris 1958.

Instytut Maszyn Matematycznych
Algorytmy N° 9

CYFROWE MODELOWANIE PROCESÓW STOCHASTYCZNYCH

Tadeusz BARTKOWSKI
Janusz ŁASKI

Pracę złożono 17.10.1966 r.

Treścią artykułu jest generacja dyskretnych procesów stochastycznych na maszynie cyfrowej ZAM-2.
Źródłem wyjściowych liczb losowych jest generator niezależnych liczb przypadkowych w przedziale /-1, +1/.

1. WSTĘP

Szeroki rozwój techniki cyfrowej umożliwia w coraz większym stopniu modelowanie stochastyczne dyskretnych układów automatycznej regulacji i sterowania. Modelowanie i analiza dyskretnych układów sterowania przy wymuszeniu przebiegami zdeterminowanymi nie oddają w pełni zachowania się układów w warunkach rzeczywistych. Modelowanie stochastyczne polegające na oddziaływaniu na model dyskretnym procesem przypadkowym, typowym dla jego rzeczywistych warunków pracy, umożliwia pełną analizę układów dyskretnych.

Problem cyfrowego modelowania stochastycznego można rozbić na dwa etapy:

- konstrukcję cyfrowego modelu układu dyskretnego oraz
- generację dyskretnego stacjonarnego procesu stochastycznego.

W zastosowaniach często wystarczający jest niepełny opis procesu stochastycznego składający się z jednowymiarowej funkcji gęstości

rozkładu prawdopodobieństwa i z funkcji korelacji. W takich wypadkach zagadnienie generacji dyskretnych procesów stochastycznych można traktować jako realizację ciągu zmiennych losowych o określonym jednowymiarowym rozkładzie prawdopodobieństwa i założonej wartości współczynników korelacji między poszczególnymi elementami ciągu.

Jak wiadomo [11] realizację zmiennej losowej o zadanym rozkładzie prawdopodobieństwa możemy otrzymać poprzez operacje przeprowadzane nad realizacją innej zmiennej losowej o równomiernym rozkładzie prawdopodobieństwa w interesującym nas przedziale zmienności.

Z kolei realizację dyskretnego procesu stochastycznego o założonej funkcji autokorelacji otrzymać możemy w postaci ciągu wyjściowego cyfrowego filtru formującego [5, 7, 8] pobudzanego ciągiem białym [5], tzn. takim którego funkcja autokorelacji $\varphi(i,j)$ między poszczególnymi elementami jest równa zeru, a dla $i = j$ równa jest dyspersji odpowiedniej zmiennej losowej.

Treścią niniejszej pracy jest generacja gaussowskich stacjonarnych procesów stochastycznych na maszynie cyfrowej ZAM-2. Jako pierwotnego źródła liczb przypadkowych użyto generatora liczb przypadkowych o rozkładzie równomiernym w przedziale /-1, +1/. [9].

2. KSZTAŁTOWANIE PROCESU GAUSSOWSKIEGO TYPU BIAŁEGO SZUMU

2.1. Ocena zbieżności rozkładu sumy n zmiennych niezależnych o rozkładzie równomiernym do rozkładu gaussowskiego

Najczęściej wykorzystywanym do modelowania procesem stochastycznym jest proces o rozkładzie normalnym. Jest to uzasadnione wymogami praktyki, gdyż poważna ilość rzeczywistych procesów fizycznych może być w przybliżeniu opisana przez ten rozkład.

Z centralnego twierdzenia granicznego wynika, że dystrybuanta sumy zmiennych niezależnych o jednakowym prawie rozkładu i wartości średniej równej zero, spełniających bardzo ogólne założenia jest zbieżna do dystrybuanty gaussowskiej. Jak się okaże, rozkład sumy

piętnastu zmiennych niezależnych o rozkładzie prostokątnym różni się od rozkładu normalnego o około 1%.

Analizę rozkładu można przeprowadzić na bazie funkcji charakterystycznych.

Wiadomo [4], że funkcja charakterystyczna sumy dwóch wzajemnie niezależnych zmiennych jak równa iloczynowi funkcji charakterystycznych tych zmiennych.

Funkcja charakterystyczna zmiennej o rozkładzie prostokątnym wyraża się wzorem [3]:

$$\varphi_1(v) = \frac{e^{jbv} - e^{jav}}{j\,v(b-a)} . \qquad /1/$$

Na podstawie twierdzenia o funkcji charakterystycznej sumy n zmiennych otrzymujemy

$$\varphi_n(v) = \left[\varphi_1(v)\right]^n = \left[\frac{e^{jbv} - e^{jav}}{jv(b-a)}\right]^n , \qquad /2/$$

gdzie $\varphi_1(v)$ jest funkcją charakterystyczną zmiennej o rozkładzie prostokątnym.

$\varphi_n(v)$ - funkcja charakterystyczna sumy n zmiennych o rozkładzie prostokątnym,
v - parametr funkcji charakterystycznej,
b - górna granica rozkładu prostokątnego,
a - dolna granica rozkładu prostokątnego.

W związku z tym, że liczby o rozkładzie równomiernym generujemy w przedziale

od a = - 1,
do b = + 1 [9], to podstawiając te wartości do /2/ otrzymujemy

$$\varphi_n(v) = \left(\frac{\sin v}{v}\right)^n . \qquad /3/$$

Rozkład prawdopodobieństwa sumy n zmiennych o rozkładzie prostokątnym $y = x_1 + x_2 + \ldots + x_n$ otrzymać można korzystając z od-

wrotnej transformacji Fouriera

$$p_n(y) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \varphi_n(v) e^{-jyv}\, dv = \frac{1}{\pi} \int_0^{\infty} \left(\frac{\sin v}{v}\right)^n \cos y\, v\, dv \,. \qquad /4/$$

Aby ocenić zbieżność $p_n(y)$ do rozkładu normalnego

$$p(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{y^2}{2\sigma^2}\right\} \qquad /5/$$

musielibyśmy wyliczyć wartości funkcji /4/ dla $n = 1,2, \ldots, N$, co jest o tyle niewygodne, że obliczanie całki niewłaściwej na maszynie cyfrowej trwa bardzo długo.

W związku z tym wykorzystamy zależność podaną przez Cramera [4] na rozkład sumy n zmiennych niezależnych o rozkładzie prostokątnym w przedziale /0,1/:

$$p_n(x) = \frac{1}{(n-1)!} \sum_{\substack{k=0 \\ x-k>0}}^{q} (-1)^k (x-k)^{n-1} \binom{n}{k}, \qquad /6/$$

gdzie $0 < x < n$.

Na każdym odcinku $l - 1 < x < l$ wartość $p_n(x)$ otrzymuje się sumując poszczególne składniki aż do wartości $q = l - 1$. Oznacza to, że poszczególne argumenty $x, x-1, \ldots, x-k$ muszą być dodatnie.

Wartością średnią zmiennej losowej X jest $\left(\frac{n}{2}\right)$, a dyspersją $\frac{n}{12}$. Aby móc ocenić stopień zbieżności $p_n(x)$ do rozkładu normalnego musimy ją przekształcić na zmienną losową o stałej wartości średniej i dyspersji.

Wprowadzamy więc zmienną standaryzowaną

$$X_{n\,st} = \frac{X_n - n/2}{\sqrt{n/12}} \,. \qquad /7/$$

Tak określona zmienna losowa ma wartość oczekiwaną równą zeru, a dyspersję stałą i równą jedności.

Jak wiadomo, gęstość rozkładu prawdopodobieństwa zmiennej losowej $Y = \varphi(X)$ będącej funkcją zmiennej X można określić ze wzoru:

$$p(y) = f[\psi(y)] \cdot [\psi'(y)], \qquad /8/$$

gdzie $\psi(y) = x$, a $f(x)$ - gęstość prawdopodobieństwa zmiennej X.

Na tej podstawie gęstość $p_{n\ st}(x)$ będzie równa

$$p_{n\ st}(x) = \sqrt{\frac{n}{12}}\, p_n\left(\frac{n}{2} + x\sqrt{\frac{n}{12}}\right). \qquad /9/$$

Rozpatrzmy teraz różnicę $\delta_n(x)$ między funkcją $p_{n\ st}(x)$ a funkcją

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}},$$

$$\delta_n(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} - \sqrt{\frac{n}{12}}\, p_n\left(x\sqrt{\frac{n}{12}} + \frac{n}{2}\right). \qquad /10/$$

Funkcja $p_{n\ st}(x)$ jest określona w przedziale $\left(-\sqrt{3n}; +\sqrt{3n}\right)$. Poza nim jest równa zeru.

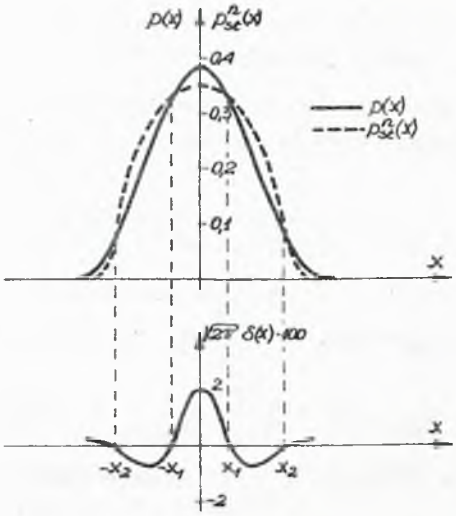
W związku z tym, że począwszy od $n = 3$ $p_{n\ st}(x)$ jest ciągła i różniczkowalna, te same właściwości posiadać będzie $\delta_n(x)$.

Ponieważ

$$\int_{-\infty}^{+\infty} p(x)\,dx = 1 \qquad /11/$$

oraz $$\int_{-\infty}^{+\infty} p_{n\ st}(x)\,dx = 1, \qquad /12/$$

to $$\int_{-\infty}^{+\infty} \left[p(x) - p_{n\ st}(x)\right] dx = 0, \qquad /13/$$



Rys. 1. Gęstości prawdopodobieństwa: p(x) rozkładu normalnego, $p_{n\ st}(x)$ sumy n - zmiennych o rozkładzie prostokątnym, i różnica między nimi $\delta_n(x)$

czyli

$$\int_{-\infty}^{+\infty} \delta_n(x)\,dx = 0 . \qquad /14/$$

W związku z tym, że $\delta_n(x)$ jest funkcją parzystą, to uwzględniając /14/ otrzymamy:

$$\int_{0}^{\infty} \delta_n(x)\,dx = 0 . \qquad /15/$$

Oznacza to, że $\delta_n(x)$ musi mieć conajmniej jedno miejsce zerowe, gdzie funkcje $p(x)$ i $p_{n\ st}(x)$ przyjmują te same wartości. W rzeczywistości, jak pokazują obliczenia, punktów takich jest dwa.

Jako kryterium zbieżności można wybrać

a/ maksymalną wartość modułu $\delta_n(x)$

$$\max \left| \delta_n(x) \right| \qquad /16/$$

lub

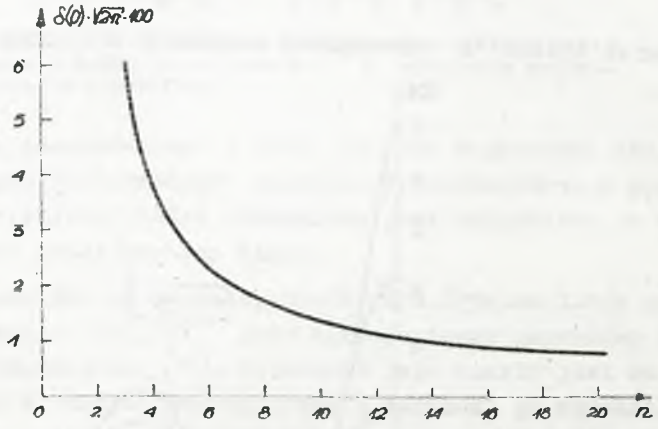
b/ maksymalną wartość różnicy między dystrybuantami

$$\max |\Delta F| = \max |F(x) - F_{n\,st}(x)|. \qquad /17/$$

Przeprowadzono obliczenia dla obu wypadków.

Jeśli chodzi o ujęcie pierwsze to z pobieżnych już oględzin można wywnioskować, że $\delta_n(x)$ w przedziale $(0,\infty)$ ma trzy ekstrema, z których jak wykazują obliczenia największą co do modułu wartość ma maksimum dla $x = 0$.

Rys. 2 przedstawia znormalizowaną w stosunku do $1/\sqrt{2\pi}$ wartość $\delta_n(0)$ w procentach.



Rys. 2. Zależność największego odchylenia gęstości rozkładu prawdopodobieństwa sumy n zmiennych o rozkładzie równomiernym od rozkładu gaussowskiego w funkcji ilości składników n /w procentach/

Inny pogląd na to zagadnienie daje rys. 3, na którym pokazano przebieg funkcji $\delta_n(x) / 1/\sqrt{2\pi}$ dla kilku wartości n.

Jeśli chodzi o ocenę największej różnicy między dystrybuantami $F(x)$ i $F_{n\,st}(x)$ to ponieważ

$$\int_{-\infty}^{+\infty} \delta_n(x)\, dx = 0,$$

zachodzi również relacja

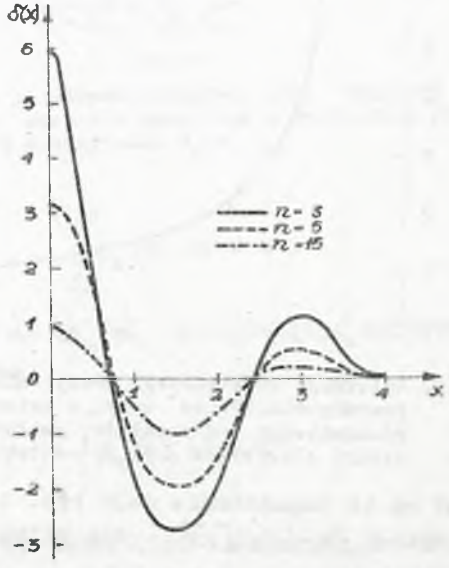
$$\int \delta_n(x)\,dx = -\int \delta_n(x)\,dx\,, \qquad /18/$$

$$x,\delta_n(x) < 0 \qquad x,\delta_n(x) > 0\,,$$

tzn. pole pod krzywą równe jest polu ponad nią. Wynika stąd, że wystarczy stablicować całkę /rys. 1/

$$\Delta F_{n\ max} = \int_{x_1}^{x_2} \delta_n(x)\,dx$$

dla różnych ilości n sumowanych zmiennych o rozkładzie prostokątnym.

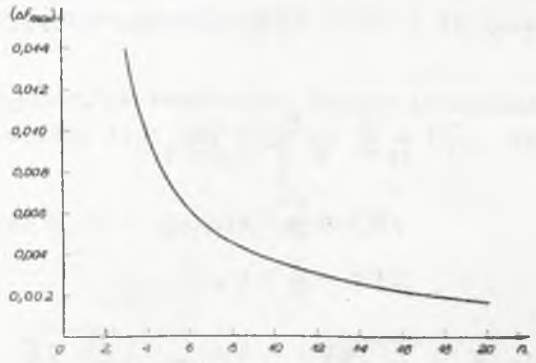


Rys. 3. Funkcja $\delta_n(x)$ dla różnych wartości n

Rezultaty zostały przedstawione na rysunku 4.

2.2. Generacja procesu gaussowskiego na maszynie cyfrowej

Do obliczeń obrano n = 15; dla takiej ilości sumowanych zmiennych o rozkładzie równomiernym funkcja gęstości rozkładu sumy różni



Rys. 4. Maksymalna rożnica między dystrybuantą rozkładu normalnego a funkcją rozkładu sumy n zmiennych prostokątnych /w procentach/

się od rozkładu gaussowskiego o około 1%. Dla większości zastosowań praktycznych jest to dokładność zupełnie wystarczająca, a poza tym wykorzystanie większej ilości składników jest utrudnione ze względu na małą szybkość pracy maszyny ZAM-2.

Jak już wspomniano na wstępie, pierwotnym źródłem liczb przypadkowych w przedziale $(-1, +1)$ jest elektroniczny generator liczb o rozkładzie równomiernym [9]. Działanie jego oparte jest na zjawiskach szumowych w lampie jonowej; szum z neonówki po wzmocnieniu i ograniczeniu w amplitudzie podany jest na licznik przejść przez zero. Czas liczenia określony jest przez synchronizację zewnętrzną, a jego minimalna wartość wynosi około 0,1 milisekundy. Parzystej ilości impulsów zliczonych w czasie trwania bramki określonej przez tę synchronizację przypisujemy wartość zero, nieparzystej - jedynkę. W ten sposób otrzymujemy ciąg impulsów zerojedynkowych o równym prawdopodobieństwie wystąpienia zera i jedynki

$$p(0) = p(1) = \frac{1}{2} . \qquad /19/$$

Wyodrębniając z tego ciągu zespoły po 36 miejsc /długość słowa w maszynie ZAM-2/ otrzymujemy liczby przypadkowe o rozkładzie równomiernym w przedziale /-1, +1/ /pozycja znaku jest również określana losowo/.

Pobieranie liczby do pamięci dokonuje się za pomocą specjalnego rozkazu [9].

W związku z niekontrolowanymi czynnikami wpływającymi na pracę generatora wartość $p(0) = \frac{1}{2}$ na ogół nie jest utrzymywana i wynosi

$$p(0) = \frac{1}{2} + \varepsilon ,$$

$$p(1) = \frac{1}{2} - \varepsilon ,$$

gdzie $|\varepsilon| < 10^{-2}$.

Zjawisko to powoduje wolne zmiany rozkładu prawdopodobieństwa liczb otrzymywanych z generatora, trudne jednakże do analitycznego ujęcia. Wobec tego jednak, że liczby gaussowskie wytwarzamy poprzez sumowanie pierwotnych liczb o rozkładzie zbliżonym do równomiernego, ostateczny kształt rozkładu gaussowskiego zostaje zachowany; przesunięciu ulega jedynie wartość średnia.

Ponieważ dyspersja zmiennych X_i o rozkładzie prostokątnym w przedziale /-1, +1/ wynosi 1/3, to wartości zmiennej standaryzowanej o rozkładzie gaussowskim liczono według wzoru

$$Y_{n\,st} = \frac{1}{\sqrt{5}} \sum_{i=1}^{15} X_i . \qquad /20/$$

Przy tych założeniach liczba gaussowska jest generowana w przedziale $(-3\sqrt{5}, +3\sqrt{5})$.

Do weryfikacji hipotezy o rozkładzie normalnym tych liczb zastosowano test χ^2; cały przedział zmienności podzielono na 100 części i liczono ilość liczb, które należały do każdego podprzedziału. Jak wiadomo [4] test χ^2 można zastosować wtedy, gdy najmniejsza ilość oczekiwanych realizacji w danym podprzedziale przekracza wartość 10; wobec tego połączono pierwsze i ostatnie 30 podprzedziałów w dwie większe grupy, co dało ostatecznie 41 stopni swobody.

Dla stopni swobody przekraczającej liczbę 30, zmienna $\sqrt{2\chi^2}$ ma w przybliżeniu rozkład normalny o wartości oczekiwanej $\sqrt{2n-1}$

i dyspersji równej jedności [4].
Funkcja

$$\varphi(x) = \int_{x}^{\infty} e^{-\frac{t^2}{2}} dt$$

osiąga wartość 0,05 w punkcie $x = 1.7$
stąd:

$$\sqrt{2\, X^2_{41;0.05}} = 1.7 + \sqrt{2x\ 41}\ ; \quad X^2_{41;0.05} = 58.$$

Na rysunku 5 przedstawiono znormalizowane liczebności poszczególnych podprzedziałów liczone według wzoru

$$p_i(\Delta X_i) = \frac{V_i}{Nx\,\Delta X}\ , \qquad /21/$$

gdzie

V_i - ilość trafień w podprzedział,

$N = 10000$ - ogólna ilość liczb wziętych do prób,

$\Delta X = \frac{3x\sqrt{5}}{50}$ - szerokość każdego podprzedziału.

Poddano również testowaniu wartość średnią; ponieważ średnia w próbce ma rozkład normalny $N(0, 6_N)$ gdzie $6_N = \frac{6}{\sqrt{N}} = \frac{1}{100}$ - odchylenie standartowe w próbce, to przedział ufności na poziomie istotności 0.05 można obliczyć jako $(-1.96x10^{-2}, +1.96x10^{-2})$.

Otrzymane rezultaty nie przeczą hipotezie o normalnym rozkładzie badanych liczb; oczywiście dla jej przyjęcia należałoby zweryfikować hipotezę alternatywną.

Wyniki testowania dyspersji podane są w następnym paragrafie.

3. Kształtowanie rozkładu o zadanej funkcji korelacji

Z kolei zajmiemy się analizą kształtowania ciągów losowych o zadanej funkcji autokorelacji. Analizę tę przeprowadzimy w oparciu o transformatę "Z" [6]

Jeśli $x_i = x(iT)$ jest funkcją określoną w dyskretnych momentach czasu, gdzie T jest okresem próbkowania, to jej "z" transformatora określona jest wzorem

$$X(z) = \sum_{i=0}^{\infty} x_i \, z^{-i} . \qquad /22/$$

Szereg /29/ jest zbieżny, jeśli wyrazy ciągu $\{x_i\}$ spełniają warunek:

$$\limsup_{i \to \infty} \sqrt[i]{|x_i|} < \infty . \qquad /23/$$

Gęstością widma dyskretnego procesu stacjonarnego nazywamy funkcję, do której zbieżny jest dwustronny szereg potęgowy

$$\psi(z) = \sum_{n=-\infty}^{+\infty} \phi(n) z^{-n} , \qquad /24/$$

gdzie

$$\Phi(n) = E\{x(i)\, x(i+n)\} \qquad /25/$$

jest funkcją autokorelacji procesu. Warunkiem istnienia $\psi(z)$, jest spełnienie warunku:

$$\limsup_{n \to \infty} \sqrt[n]{\varphi(n)} < 1 . \qquad /26/$$

Funkcja korelacji związana jest z gęstością widma równaniem:

$$\Phi(n) = \frac{1}{2\pi j} \oint_{\Gamma} z^{+n} \psi(z) \frac{dz}{z} , \qquad /27/$$

gdzie Γ jest konturem, po którym odbywa się całkowanie.

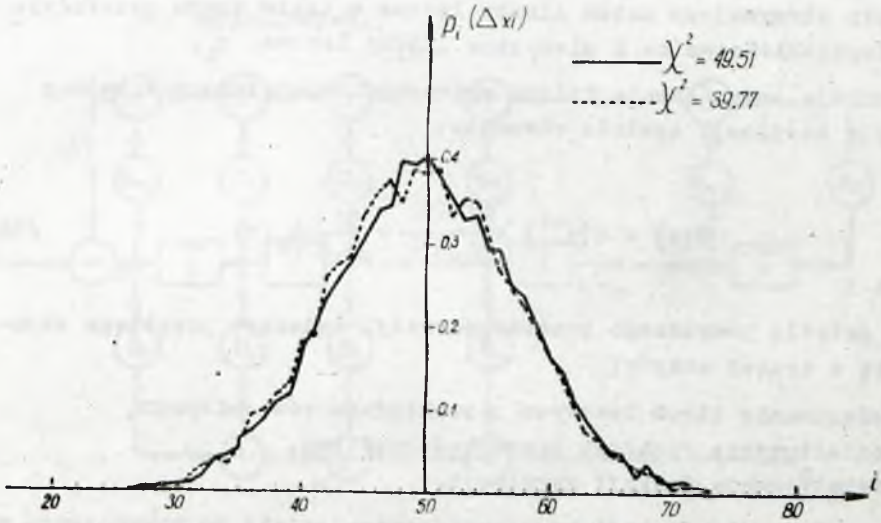
Gęstość widmowa ψ_y sygnału y, na wyjściu stabilnego układu impulsowego określona jest wzorem:

$$\psi_y(z) = H(z) \cdot H(z^{-1})\psi_x(x(z)), \qquad /28/$$

gdzie $H(z)$ oznacza funkoję przenoszenia układu impulsowego /filtru cyfrowego/

$$H(z) = \sum_{k=0}^{\infty} h_k z^{-k}. \qquad /29/$$

Z zależności /28/ widać, że za pomocą układu liniowego opisanego funkcją $H(z)$, można ciąg liczb losowych o znanej gęstości widmowej $\psi_x(z)$ przekształcić w inny ciąg liczb losowych o zadanej gęstości widmowej $\psi_y(z)$. Jako ciąg pierwotny liczb losowych o zadanym rozkładzie prawdopodobieństwa będziemy brać liczby losowe wzajemnie niezależne.



Rys. 5. Znormalizowane liczebności poszczególnych podprzedziałów dla dwóch ciągów po 10.000 elementów w każdym

Funkcja korelacji takiego ciągu liczb wzajemnie niezależnych wynosi:

$$\Phi_x(n) = \begin{cases} 0 & \text{dla} \quad n \neq 0 \\ \sigma_x^2 & \text{dla} \quad n = 0 \end{cases} . \qquad /30/$$

Gęstość widmowa takiego ciągu liczb losowych przyjmuje bardzo prostą postać:

$$\Psi_x(z) = \sigma_x^2 . \qquad /31/$$

Problem kształtowania liczb losowych o zadanej funkcji korelacji sprowadza się zatem do liniowego przekształcenia niekorelowanych liczb losowych x_i za pomocą filtru cyfrowego o odpowiednio dobranej funkcji przenoszenia $H(z)$.

Ponieważ filtr cyfrowy jako układ liniowy nie zmienia postaci rozkładu prawdopodobieństwa pojedynczej zmiennej losowej na jego wyjściu otrzymujemy zatem liczby losowe o takim samym rozkładzie prawdopodobieństwa co i pierwotne liczby losowe x_i.

Funkcja przenoszenia filtru cyfrowego zapewniającego żądaną funkcję korelacji spełnia równanie:

$$H(z) \cdot H(z^{-1}) = \frac{\Psi_y(z)}{\Psi_x(z)} = \frac{1}{\sigma_x^2} \Psi_y(z) . \qquad /32/$$

W świetle powyższego proces generacji żądanego przebiegu składa się z trzech etapów:

a/ generowanie liczb losowych o rozkładzie równomiernym,
b/ kształtowanie rozkładu prawdopodobieństwa,
c/ kształtowanie funkcji korelacji.

Kształtowanie funkcji korelacji sprowadza się do wyznaczenia z zależnośoi /32/ struktury filtru cyfrowego, a następnie na podstawie struktury algorytmu realizującego filtr cyfrowy.

Funkcja przenoszenia filtru cyfrowego przyjmuje na ogół postać funkcji wymiernej.

$$H(z) = \frac{b_0+b_1 z^{-1}+ \ldots +b_m z^{-m}}{a_0+a_1 z^{-1}+ \ldots +a_m z^{-n}}, \qquad /33/$$

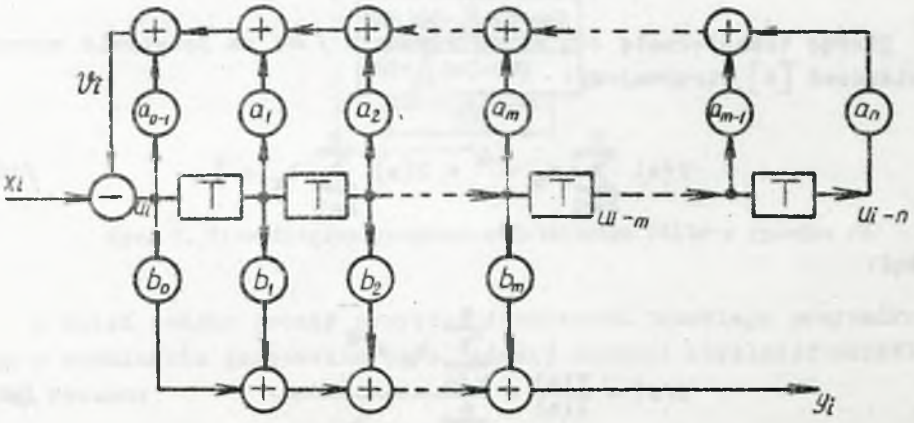
gdzie a_i i b_i są współczynnikami.

Warunkiem realizowalności fizycznej filtru cyfrowego jest nierówność:

$$n \geqslant m \qquad /34/$$

Dla tego przypadku struktura filtru cyfrowego przyjmuje postać pokazaną na rys. 6. Na rys. 6 oznaczono przez

T – element wprowadzający opóźnienie liczb o jeden okres,
+ – węzeł sumujący,
– – " odejmujący,
a_i – układy mnożące liczby przez stały współczynnik,
x_i – ciąg liczb wejściowych,
y_i – " " wyjściowych.



Rys. 6. Ogólny schemat blokowy filtru cyfrowego

Filtr cyfrowy przedstawiony na rys. 6 opisują następujące równania różnicowe:

a/ ciąg liczb na wyjściu węzła odejmującego

$$U_i = x_i - v_i \,, \qquad /35/$$

b/ ciąg liczb gałęzi sprzężania zwrotnego

$$v_i = (a_0 - 1)\,U_i + \sum_{k=1}^{n} a_k\, U_{i-k} \,, \qquad /36/$$

c/ ciąg liczb wyjściowych

$$y_i = \sum_{k=0}^{m} b_k\, U_{i-k} \,. \qquad /37/$$

Po elementarnych przekształceniach otrzymujemy równanie różnicowe wiążące ciąg liczb wejściowych z ciągiem liczb wyjściowych

$$\sum_{k=0}^{n} a_k\, y_{i-k} = \sum_{k=0}^{m} b_k\, x_{i-k} \,. \qquad /38/$$

Biorąc transformatę obu stron równania /38/ na podstawie znanych twierdzeń [6] otrzymujemy:

$$Y(z) \sum_{k=0}^{n} a_k\, z^{-k} = X(z) \sum_{k=0}^{m} b_k\, z^{-k} \,, \qquad /39/$$

skąd:

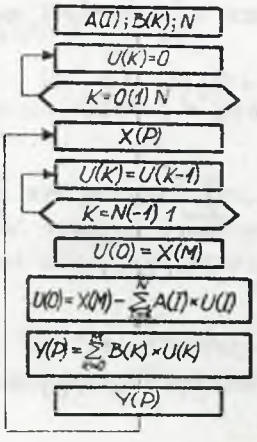
$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k/0}^{m} b_k\, z^{-k}}{\sum_{k/0}^{n} a_k\, z^{-k}} \qquad /40/$$

Dla uproszczenia struktury filtru zakłada się na ogół

$$a_0 = 1.$$

Założenie to nie zmienia ogólności rozważań, a upraszcza nieco obliczenia. Oznacza to, że do struktury filtru nie wchodzi zakreso-wana gałęź na rys. 6.

Algorytm realizujący powyższą ideę kształtowania funkcji korela-cji można w prosty sposób zamodelować na maszynie cyfrowej. Schemat blokowy takiego programu pokazano na rys. 7.



Rys. 7. Flow-diagram programu modelującego filtr z rysunku /8/

Z kolei podamy prosty przykład formowania przebiegu przypadkowe-go o rozkładzie gausowskim oraz żądanej funkcji korelacji określo-nej wzorem:

$$\Phi_y(n) = \sigma_y^2 e^{-\alpha|n|} \qquad /41/$$

Współczynniki korelacji pierwotnych liczb losowych są określone wzorem

$$\Phi_x(n) = \begin{cases} \sigma_x^2 = 1 & \text{dla} \quad n = 0 \\ 0 & \text{dla} \quad n \neq 0 \end{cases} ; \qquad /42/$$

zaś ich transformata korelacyjna wynosi:

$$\Psi_x(z) = \sigma_x^2 = 1 . \qquad /43/$$

Żądaną gęstość widmową ciągu $\{x_i\}$ o funkcji korelacji określonej wzorem /41/ wyznaczamy z zależności na transformatę dwustronną:

$$\Psi_y(z) = \Psi_+(z) + \Psi_-(z) , \qquad /44/$$

gdzie

$$\Psi_+(z) = \sum_{n=0}^{\infty} \Phi(n)\, z^{-n} = \frac{\sigma_y^2 z}{z-\beta} , \qquad /45/$$

$$\beta = e^{-\alpha} , \qquad /46/$$

$$\Psi_-(z) = \Psi_+(z^{-1}) - \Phi(0) = \frac{\sigma_y^2 \beta z}{1-\beta z} , \qquad /47/$$

skąd otrzymujemy:

$$\Psi_y(z) = \frac{\sigma_y^2 (1-\beta^2) z}{(1-\beta z)(z-\beta)} . \qquad /48/$$

Ze wzoru /32/ otrzymujemy:

$$H(z) \cdot H(z^{-1}) = \frac{\sigma_y^2 (1-\beta^2) z}{\sigma_x^2 (1-\beta z)(z-\beta)} . \qquad /49/$$

Stąd funkcja przenoszenia filtru cyfrowego wynosi:

$$H(z) = \sqrt{\frac{\sigma_y^2}{\sigma_x^2}} \frac{\sqrt{1-\beta^2}\, z^{-1}}{1-\beta z^{-1}} = \frac{K z^{-1}}{1-\beta z^{-1}}, \qquad /50/$$

gdzie:

$$K = \frac{\sigma_y}{\sigma_x}\sqrt{1-\beta^2}. \qquad /51/$$

Łatwo sprawdzić, że funkcja $H(z)$ dana wzorem /50/ nie jest jedyną funkcją przenoszenia spełniającą równanie /49/.

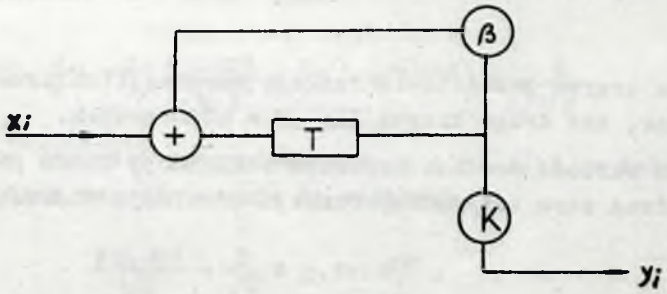
Każdy filtr cyfrowy opisany przez funkcję typu

$$H_1(z) = \frac{K z^{-m}}{1-\beta z^{-1}}$$

również dokonuje formowania procesu o gęstości widmowej określonej przez /48/.

Fakt ten da się łatwo wytłumaczyć tym, że obecność czynnika z^{-m} w liczniku wyrażenia /50/ oznacza po prostu opóźnienie sygnału wejściowego o m okresów, co dla ciągu białego nie gra oczywiście żadnej roli.

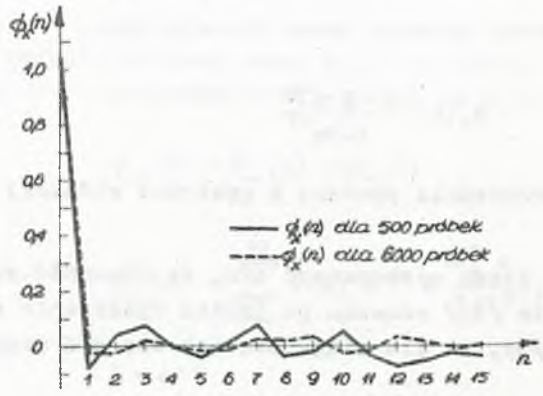
Algorytm transformaty filtru cyfrowego $H(z)$ dany wzorem /50/ zrealizowano według schematu blokowego pokazanego na rys. 8.



Rys. 8. Układ filtru o funkcji przenoszenia $H(z) = \frac{Kz^{-1}}{1-\beta z^{-1}}$

Przeprowadzono modelowanie na maszynie cyfrowej prooesu o funkcji korelacji danej wzorem /41/ dla wartości parametru $\alpha = 0.2$, dyspersji $6_y^2 = 1$.

Na rysunku 9 pokazano funkcję korelacji $\hat{\Phi}_x(x)$ przypadkowych, znormalizowanych liczb gaussowskich utworzonych poprzez sumowanie 15-tu liczb przypadkowych o rozkładzie równomiernym, zaś w tabelce zestawiono wyniki eksperymentalne.



Rys. 9. Funkcja korelacji liczb o rozkładzie gaussowskim na wyjściu filtru cyfrowego

Pierwsza krzywa przedstawia funkcję korelacji obliczoną dla $N = 500$ próbek, zaś druga krzywa dla $N = 6000$ próbek.

Ponieważ wartość średnia krótkich realizacji liczb przypadkowych nie jest równa zeru estymatory funkcji korelacji liczono na podstawie wzoru:

$$\hat{\Phi}(n) = \frac{1}{N} \sum_{i=1}^{N} X_i X_{i+n} - \bar{X}^2, \qquad /52/$$

gdzie $\bar{X}$ jest wartością średnią liczb losowych liczoną za N bek.

Dokonano obliczeń przedziałów ufności dla estymatorów dyspersji i funkcji korelacji.

Jeśli chodzi o dyspersję, to rozkład jej estymatora liczonego według wzoru

$$s^2 = \frac{1}{N} \sum_{i=1}^{N} \left(x_i - \bar{X}\right)^2 , \qquad /53/$$

gdzie $\bar{X}$ - wartość średnia w próbce ma rozkład /dla dużych N/ normalny

$$N\left(\frac{N}{N-1}\,\sigma^2, \frac{\sigma}{N}\sqrt{2(N-1)}\right) [4] .$$

Otrzymujemy więc na poziomie istotności $\alpha = 0.05$ przedział ufności $\left(-3.5 \times 10^{-2} + 1, + 3.5 \times 10^{-2} + 1\right)$.

Obliczanie funkcji korelacji możemy traktować jako estymację współczynnika korelacji ρ dwóch zmiennych losowych. Wtedy [4] współczynnik korelacji R w próbce ma rozkład normalny ,

$$N\left(\rho, \frac{1-\rho^2}{\sqrt{N}}\right) .$$

Można więc wyznaczyć wartość połowy symetrycznego przedziału ufności na poziomie $\alpha = 0.05$ jako

$$|R-\rho| = |\Delta R| = \left(1-\rho^2\right)\frac{\Phi 0.025}{\sqrt{N}} = \left(1 - e^{-0.4n}\right) \times \frac{2}{\sqrt{N}} .$$

Otrzymane wartości zestawiono dla $n = 1 \ldots 5$ w tabelce; dla $n > 5$, ΔR dąży asymptotycznie do wartości

$$\frac{\Phi 0.025}{\sqrt{N}} = \frac{2}{\sqrt{N}} \cong 2.5 \times 10^{-2} ,$$

gdzie $\Phi_\delta = x_0$, dla którego $\frac{1}{\sqrt{2\pi}} \int_{x_0}^{\infty} e^{-\frac{t^2}{2}}\, dt = \delta$

i $N = 6000$.

n	ΔR
1	0.008
2	0.014
3	0.0175
4	0.0200
5	0.02150

W przypadku ciągu białego kładziemy $\varrho = 0$; otrzymujemy więc przedział ufności $1 \pm 2.5 \times 10^{-2}$.

W tabeloe umieszczonej poniżej zostały podane wyniki liczbowe; jak widać nie ma podstaw do odrzucenia takiej hipotezy, że funkcja korelacji generowanego prooesu na odcinku /0,15/ ma postać $e^{-0.2n}$.

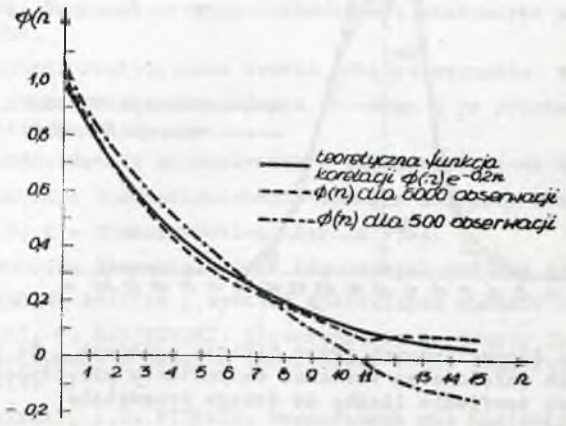
Uwzględnienie większych wartości n jest utrudnione ze względu na małą szybkość pracy maszyny ZAM-2.

	Funkcja korelacji przed filtrem			Funkcja korelacji po filtrze	
n	N = 500	N = 6000		N = 500	N = 6000
0	+ 1,0684	+ 1,0372	+ 1,0000	+ 1,0644	+ 1,0090
1	- 0,0666	- 0,0125	+ 0,8187	+ 0,8652	+ 0,8200
2	+ 0,0415	- 0,0119	+ 0,6703	+ 0,7285	+ 0,6670
3	+ 0,0607	+ 0,0115	+ 0,5488	+ 0,6084	+ 0,5468
4	+ 0,0034	- 0,0003	+ 0,4493	+ 0,4940	+ 0,4450
5	- 0,0157	- 0,0049	+ 0,3679	+ 0,3958	+ 0,3621
6	- 0,0133	- 0,0026	+ 0,3012	+ 0,3192	+ 0,2968
7	- 0,0746	+ 0,0119	+ 0,2466	+ 0,2531	+ 0,2455
8	- 0,0113	+ 0,0039	+ 0,2019	+ 0,1738	+ 0,2006
9	- 0,0033	+ 0,0104	+ 0,1653	+ 0,1101	+ 0,1633
10	+ 0,00357	- 0,0146	+ 0,1353	+ 0,0521	+ 0,1299
11	+ 0,0408	- 0,0122	+ 0,1108	- 0,0148	+ 0,1090
12	- 0,0684	+ 0,0182	+ 0,0907	- 0,0697	+ 0,0985
13	- 0,0432	+ 0,0027	+ 0,0743	- 0,0923	+ 0,0857
14	- 0,0124	- 0,0043	+ 0,0608	- 0,01055	+ 0,0764
15	- 0,0342	- 0,0007	+ 0,0498	- 0,012040	+ 0,0729

Na rys. 10 pokazano z kolei funkcję korelacji znormalizowanyoh liozb losowyoh o rozkładzie gaussowskim na wyjściu filtru cyfrowe-

go opisanego wzorem /50/, przyczym współczynnik α wynosił 0,2, zaś dyspersja $\sigma_y^2 = 1$.

Na rys. 10 wykreślono 3 krzywe: teoretyczną funkcję korelacji $\Phi_y(n) = e^{-0.2n}$, obliczoną funkcję korelacji dla $N = 500$ próbek oraz $N = 6000$ próbek.



Rys. 10. Funkcja korelacji liczb na wyjściu filtru cyfrowego

Testowano również rozkład prawdopodobieństwa generowanych ciągów testem χ^2 z 19 stopniami swobody.

Otrzymane wyniki wskazują na to, że rzeczywiście filtr cyfrowy nie zmienia postaci rozkładu prawdopodobieństwa.

Należy zaznaczyć, że jeśli wartość średnia X obliczana w próbce złożonej z N elementów nie jest równa zeru, to wartość średnia procesu po filtrze będzie większa o

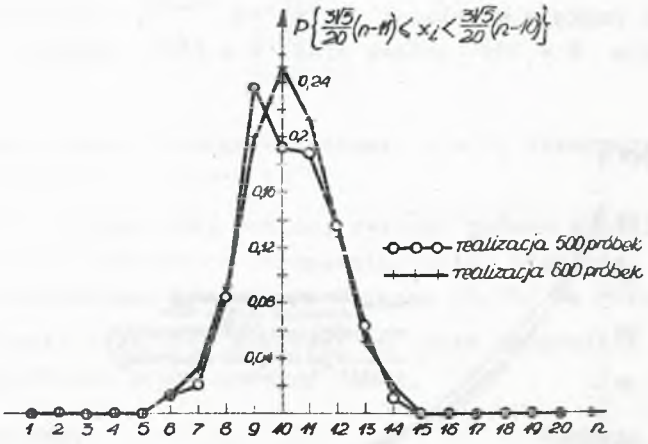
$$K = \frac{\sqrt{1-\beta}^2}{1-\beta} \qquad /54/$$

i równa wzmocnieniu filtru.

Rysunek 11 przedstawia znormalizowane częstości trafienia elementów ciągu na wejściu filtru w każdy z 20-tu podprzedziałów na

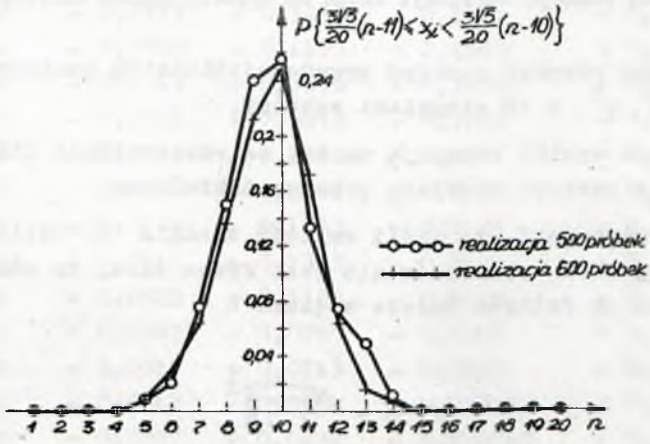
jaki podzielono odcinek $(-3\sqrt{5}, +3\sqrt{5})$, czyli

$$\hat{p}\left[\frac{3\sqrt{5}}{10}(n-11) \leq x_i \leq \frac{3\sqrt{5}}{10}(n-10)\right]. \qquad /55/$$



Rys. 11. Rozkład liczb losowych przed filtrem cyfrowym . Na osi rzędnych odłożone są wartości estymatorów prawdopodobieństwa trafienia liczby do danego przedziału

Podobne wyniki dla ciągu na wyjściu filtru przedstawia rys.12.



Rys. 12. Rozkład liczb losowych po filtrze cyfrowym

Na zakończenie autorzy pragnęliby podziękować mgr inż. W Porębskiemu z Katedry Teorii Sterowania Politechniki Gdańskiej za uwagi i dyskusję w czasie przygotowywania tej pracy.

Literatura

[1] M. FISZ: Rachunek prawdopodobieństwa i statystyka matematyczna. W-wa 1958.

[2] J. SEIDLER: Statystyczna teoria odbioru sygnałów. W-wa 1963.

[3] B.R. LEWIN: Teoria słuczajnych procesow i je primienienije w radiotechnikie

[4] H. CRAMER: Metody matematyczne w statystyce. W-wa 1958.

[5] P.D. KRUTKO: Statisticzeskaja dinamika impulsnych sistem. Moskwa 1963.

[6] R. VICH: Z - Transformation. Berlin 1964.

[7] L.N. WOŁGIN: Elementy Teorii impulsowych układów regulacji. W-wa 1964.

[8] L.T. KUZIN: Analiza i synteza dyskretnych układów sterowania. W-wa 1965.

[9] J. ŁASKI, T. BARTKOWSKI: Elektroniczny generator liczb przypadkowych do współpracy z maszyną ZAM-2 Beta. Prace IMM, Tom·IV, B 18/31/, W-wa 1967.

[10] J.S. BENDAT, A.G. PIERSOL: Measurement and Analysis of Random Data. New York 1966.

[11] N.P. BUSLENKO: Metod statisticeskich ispytanij /Monte Carlo/ i jevo realizacja na cifrowych wycislitelnych masinach. Moskwa 1961.

DIGITAL SIMULATION OF STOCHASTIC PROCESSES

Summary

Given a method of forming gaussian process with an exponential function of correlation. The first chapter contains an evaluation of the convergance of the sum of n random variables with a gaussian distribution. Given the synthesis of a digital filter forming the sequence with a given function of correlation from the white noise, on the basis of transformation "Z". Finally, the results of the process simulation are presented of exponantial autocorrelation function on the digital computer ZAM-2.

THEORY OF PROGRAMMING

Instytut Maszyn Matematycznych
Algorytmy N° 9

ON A CERTAIN PROBLEM OF EDITION

by Jan WIERZBOWSKI

Received July 15th, 1967

The paper contains an attempt of algorithmization of a certain problem in the field of data processing. The problem deals with the programming of edition of the matrix with a big number of constant elements. The algorithm given can serve as the basis to develop a standard program for a digital computer.

INTRODUCTION

One of the characteristic features of programs in the field of data processing is the output of a large number of results. This fact involves the necessity of the result output in the form of editions that are fixed in advance, sufficiently readable for a wide circle of users.

A payroll can be an example of such an edition. Individual columns correspond here to various components of the sum to be paid - basic payment, additions, deductions etc. but the lines correspond to separate employees. Such an edition can be accepted as a table containing some, but not all elements.

Up till now each case of such an edition was solved separately by programmers. Therefore, it is purposeful to take a common algorithm, which would solve the above problem in a possibly simple way. The present paper is destined to make such an algorithm.

Problem formulation

Let us assume that the given set Z, consisting of ordered pairs of numbers, satisfies for certain natural numbers M and N the following conditions:

if $\langle i,j \rangle \in Z$ then $i \leq M, j \leq N$, /1/

for all $j \leq N$ there exists $i \leq M$ such that $\langle i,j \rangle \in Z$, /2/

for all $i \leq M$ there exists $j \leq N$ such that $\langle i,j \rangle \in Z$. /3/

Assume further, that on the set Z the function f is determined, the values of which belong to the set of real numbers. The so-determined function f will be called generalized matrix.

Note that if the set Z satisfies the condition /1/ and the following conditions

for each $i \leq M$ and $j \leq N$ $\langle i,j \rangle \in Z$, /2a-3a/

the function f is a matrix in a commonly accepted sense.

If the condition /2a-3a/ occurs then conditions /2/ and /3/ are true, thus the term "generalized matrix" is justified.

For simplification we shall write $f(i,j)$ instead of $f(\langle i,j \rangle)$ to denote the elements of matrix f. The matrix $\bar{f}$ will be called the extension of the generalized matrix f if:

1. $\bar{f}$ is determined on the set $\bar{Z}$ consisting of all pairs of natural numbers $\langle i,j \rangle$, for which $i \leq M$, $j \leq N$,

2. $$\bar{f}(i,j) = \begin{cases} f(i,j) & \text{for } \langle i,j \rangle \in Z, \\ \lambda & \text{for } \langle i,j \rangle \bar{\in} Z, \end{cases} \qquad /4/$$

where λ is an arbitrary real number.

Note that the above definition determines uniformly the extension of an arbitrary generalized matrix with the assumption that the number λ is being fixed.

In a special case the matrix itself can be its extension.

Further, let us assume that λ is a certain number given in advance. These auxiliary ideas being introduced we can formulate the proper problem.

Assume, that a certain matrix f is stored in a digital computer in an arbitrary form, and that we want to obtain this matrix in an editional form of the extension of matrix $\bar{f}$. As the matrix $\bar{f}$ is a rectangular one, the commonly accepted form of a rectangular matrix record will be accepted as its editional form.

However, assume that a digital computer output device permits to print only K columns and, in connection with it, the editional process should be divided into R steps, where:

$$R = \bar{E}\left(\frac{N}{K}\right)^{*)} \qquad /5/$$

printing in the p step, the columns

$$(p-1)K + 1,\ (p-1)K + 2,\ \ldots,\ \min(pK,\ N). \qquad /6/$$

In the above formulation, the printing of columns is thought to be the printing of sequential elements of the first line and of the cited columns, then, of the second line and of the cited columns, and so on.

The problem solution

The purpose of the present paper is to build a relatively simple algorithm which would make it possible to write the matrix f in the described way.

In some special cases the formulated problem is being solved in a very simple way, e.g.:

1. if f is a rectangular matrix stored in a digital computer by means of lines or columns,
2. if $K \geqslant N$, and the matrix f is stored in lines.

However, a general case requires more complex methods. For its solution we determine the funotion g on the set Z as follows:

$$g(i,j) = KME\left(\frac{j-1}{K}\right) + Ki + j\ . \qquad /7/$$

Let us sort the file of operands and values of function f, /more strictly speaking, a set of elements $\langle i,j,\ f(i,j)\rangle$ / according to the increasing values of the function g. Then, the values of the function $f(i,j)$ will be recorded in the sequence,

*) $\bar{E}(x)$ denotes the smallest integer number not being smaller than x and $E(x)$ is entier.

they should be printed. In order to show this it suffices to prove that the following three conditions are satisfied:

1. all elements to be printed in any earlier step appear in the object set before the elements which are to be printed in a later step;
2. among elements to be printed in one step, the ones which are to be printed in any earlier line appear earlier;
3. among elements to be printed in the same step and in the same line, the elements of earlier columns appear first.

Let us now prove these three conditions.

Assume that $f(i_1,j_1)$ is to be printed in step p_1, and $f(i_2,j_2)$ - in step p_2, where $p_1 \leq p_2 - 1$. From formula /6/ it results that

$$(p_1 - 1)K + 1 \leq j_1 \leq \min(p_1K, N), \qquad /8/$$

$$(p_2 - 1)K + 1 \leq j_2 \leq \min(p_2K, N). \qquad /9/$$

From definition /7/ it results that

$$g(i_1,j_1) = KME\left(\frac{j_1-1}{K}\right) + Ki_1 + j_1 \leq KM(p_1-1) + KM+p_1K = KMp_1+Kp_1 \qquad /10/$$

$$g(i_2,j_2) = KME\left(\frac{j_2-1}{K}\right) + Ki_2 + j_2 \geq KM(p_2-1) + K + (p_2-1)K + 1 \geq$$

$$\geq KMp_1 + Kp_1 + K + 1 > KMp_1 + Kp_1, \qquad /11/$$

thus

$$g(i_1,j_1) < g(i_2,j_2), \qquad /12/$$

which proves that condition 1 is satisfied.

Assume now, that $f(i_1,j_1)$ and $f(i_2,j_2)$ are to be printed in the same step, i.e. for a certain p:

$$(p-1)K+1 \leq j_1 \leq \min(pK,N), \qquad /13/$$

$$(p-1)K+1 \leq j_2 \leq \min(pK,N), \qquad /14/$$

and $i_1 \leq i_2 - 1$. Then

$$\begin{aligned} g(i_1,j_1) &= KME\left(\frac{j_1-1}{K}\right) + Ki_1 + j_1 = \\ &= KM(p-1) + Ki_1 + j_1 \leq \qquad /15/ \\ &\leq KM(p-1) + Ki_1 + pK, \end{aligned}$$

$$\begin{aligned} g(i_2,j_2) &= KME\left(\frac{j_2-1}{K}\right) + Ki_2 + j_2 = \\ &= KM(p-1) + Ki_2 + j_2 \geq \\ &\geq KM(p-1) + K(i_1+1) + (p-1)K+1 = \\ &= KM(p-1) + Ki_1 + pK + 1 > \\ &> KM(p-1) + Ki_1 + pK, \end{aligned}$$

thus

$$g(i_1,j_1) < g(i_2,j_2), \qquad /16/$$

which proves that condition 2 is satisfied too.

Finally, let us assume that $f(i_1,j_1)$ and $f(i_2,j_2)$ are to be printed in the same step, i.e. formulae /13/, /14/ occur and $i_1 = i_2$ and $j_1 \leq j_2 - 1$.

Then

$$\begin{aligned} g(i_1,j_1) &= KME\left(\frac{j_1-1}{K}\right) + Ki_1 + j_1 = \\ &= KM(p-1) + Ki_1 + j_1, \end{aligned} \qquad /17/$$

$$g(i_2,j_2) = KME\left(\frac{j_2-1}{K}\right) + Ki_2 + j_2 =$$

$$= KM(p-1) + Ki_1 + j_2 \geqslant$$

$$\geqslant KM(p-1) + Ki_1 + j_1 + 1 > \qquad /18/$$

$$> KM(p-1) + Ki_1 + j_1,$$

which ends the proof of the condition 3 and the entire consideration.

Now, let us estimate the values of the function g. As this is easy to be seen

$$g(i,j) \geqslant Ki + j \geqslant K + 1. \qquad /19/$$

Simultaneously

$$g(i,j) \leqslant KM\left(\frac{j-1}{K}\right) + Ki + j \leqslant$$

$$\leqslant M(N-1) + KM + N = M(N + K - 1) + N. \qquad /20/$$

From formulae /19/ and /20/ we have

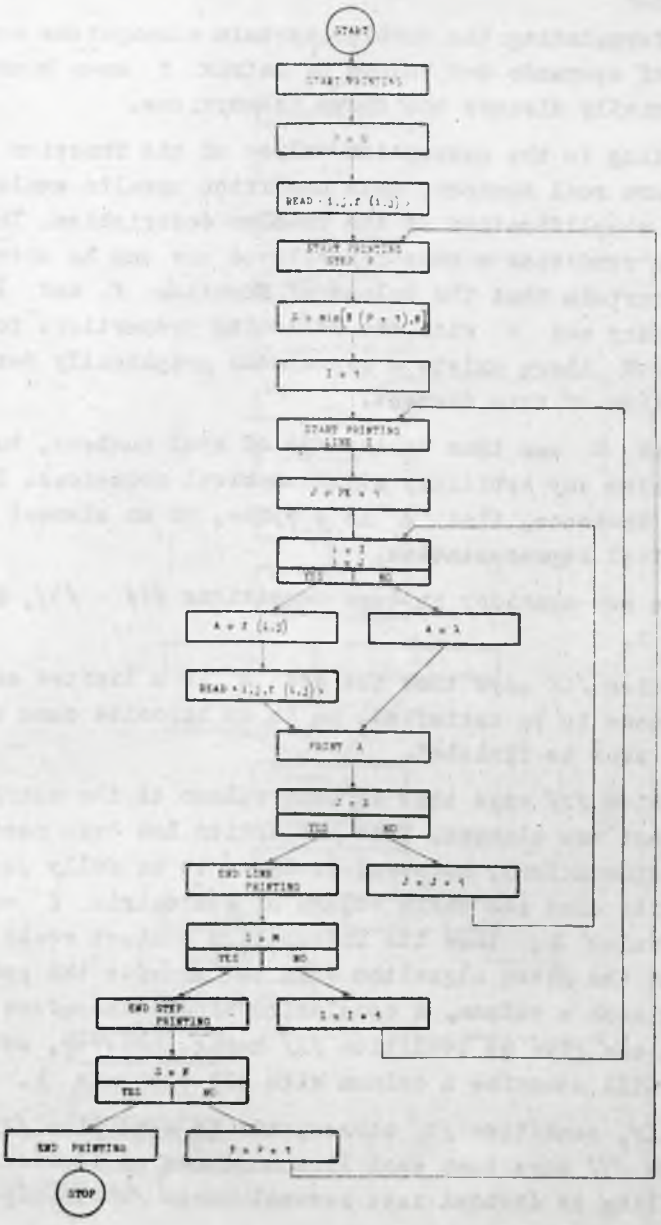
$$K + 1 \leqslant g(i,j) \leqslant M(N + K - 1) + N. \qquad /21/$$

Flow diagram

On the basis of the above considerations, an flow diagram can be built, precising in detail the algorithm of printing the matrix f. Assume that at the beginning and at the end of the matrix f, as well as before and after each step and line, some additional information, headings and so on can be printed.

To simplify the flow diagram let us assume that the set of elements $< i,j,f(i,j) >$ comprises $< M, N_1, \lambda >$ too, where $N_1 > N$, and that the entire set is ordered according to the function g. As it is easy to see the element $< M, N_1, \lambda >$ is the last one in this set. It only serves to ascertain that all elements $<i,j,f(i,j)>$ for $i,j \in Z$ had already been entered into the computer /e.g. from a magnetic tape/.

The scheme No 1 illustrates the discussed flow diagram.



Scheme 1

Final notes

When formulating the problem certain assumptions concerning the set of operands and values of matrix f have been made. Now, let us briefly discuss the above assumptions.

According to the assumption values of the function f as well as λ, are real numbers. This condition results exclusively because of simplification of the problem description. Instead of the above condition a more generalized one can be accepted, which would ascertain that the values of function f and λ belong to an arbitrary set W with the following properties: for each element $w \in W$ there exists a synonimous graphically determined representation of this element.

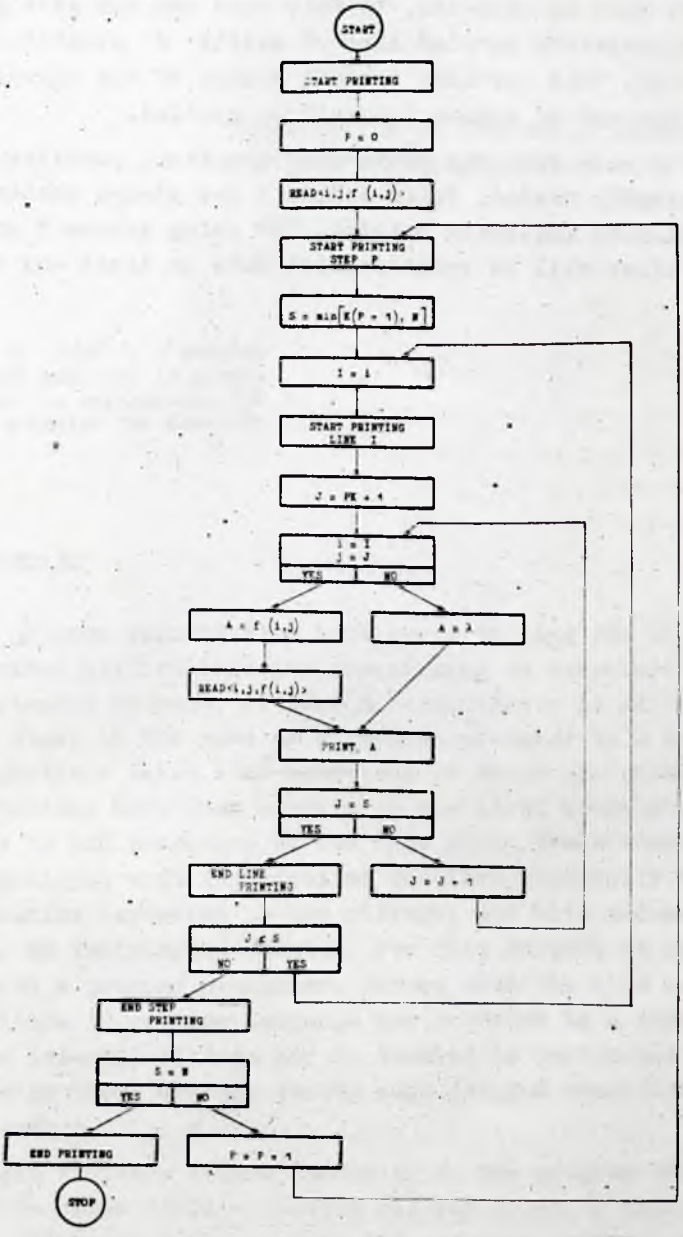
The set W can thus be the one of real numbers, but it also can comprise say arbitrary alphanumerical sequences. It may happen, for instance, that λ is a space, or an element having empty graphical representation.

Let us now consider in turn conditions /1/ - /3/, determining the set Z.

Condition /1/ says that the set Z is a limited one. This condition needs to be satisfied, as in an opposite case the process couldn't ever be finished.

Condition /2/ says that in each column of the matrix f there is at least one element. This assumption has been used during further considerations, however, it seems to be fully justified. In an opposite case the whole column of the matrix f would have had the value λ, thus its information content would be negligible. But the given algorithm does not provide the possibility of omitting such a column. A conclusion should therefore be drawn that one can give up condition /2/ being, however, aware that the results will comprise a column with all elements λ.

Finally, condition /3/ corresponds to condition /2/ for lines. Condition /3/ says that each line contains at least one element. As each line is divided into several parts /obviously if $N > K$/


START
START PRINTING
P = 0
READ<i,j,f (i,j)>
START PRINTING STEP P
S = min[K(P + 1), N]
I = 1
START PRINTING LINE I
J = PK + 1
I = I
J = J
YES
NO
A = f (I,J)
A = λ
READ<I,J,f(I,J)>
PRINT A
J = S
YES
NO
END LINE PRINTING
J = J + 1
J ≤ S
NO
YES
END STEP PRINTING
S = N
YES
NO
END PRINTING
P = P + 1
STOP

Scheme 2

which are printed in subsequent steps, it can occur that some parts will have no elements. In this case one can give up printing the appropriate part of line of matrix f consisting of values λ only. This requires a small change of the algorithm, i.e. scheme 2 instead of scheme 1 should be applied.

As it is seen from the above consideration, condition /3/ is not essentially needed. Using scheme 1 one always obtains all the lines from 1 to M being printed, but using scheme 2 only these parts of lines will be printed which have at least one element.

Instytut Maszyn Matematycznych
Algorytmy N° 9

ORGANISATION OF SEGMENT EXCHANGE IN ALGOL FOR ZAM-21-ALFA AND ZAM-41-ALFA COMPUTERS

by Ludwik CZAJA

Received November 7th, 1967

A method of running a program divided into segments is given. This method is implemented for the ALGOL compiler for ZAM-ALFA computers.

1. THE PROBLEM

Large program segmentation into parts filling the storage always provides difficulties for programming on computers with rather not extended storage. If such a segmentation is automatic, which can take place in the case of a program produced by a translator, several problems arise some solutions of which are given below. These solutions have been applied to the ALGOL translator now implemented to ZAM computers of the type ALFA. These computers are neither equipped with instructions destined especially to make use of information segmented in the storage, nor with address modification by an instruction counter. For this purpose it is necessasy to build a program mechanism. Assume that the size of a program, written in machine language and produced by a translator exceeds the internal storage and is located in peripheral storage. Then, the problems arising during such program execution would be the following:

1. Where variable values appearing in the program should be located. In other words - whether all variables of the actually executed block are to be kept in the internal storage or whether they should be divided into groups independent of the program

block structure. Because of a faster access to the variable we have chosen the first way /described in [1]/.

If there are many data, the programmer uses the standard procedure for rewriting data into and from the peripheral storage.

2. In which way is the program to be divided into parts called to the internal storage for execution, i.e. are the parts to be equal or not depending on the content. If we divide it into equal parts /formally/, then on the one hand, we would be able to do it by means of a more simple translator which would not bother about the translator program semantics and, on the other hand, the program calling out these parts to the internal storage would also be simpler. If it is divided into unequal parts depending on its semantics, e.g. taking care not to cut internal loops, we shall get a program divided "intelligently", as if it had been made directly by man. In our translator we have applied the segmentation into equal parts /segments/. We have chosen this way, first of all, because of a simpler program calling segments. In an opposite case the program would take more space in the internal storage, therefore, less space would be available for the object program and, besides, its action would be slower.

The execution of a segmented program requires the solution of:

a/ the determination of the segment that is to be cancelled in order to provide space for an actually needed segment /the problem of the segment choice/, and

b/ the updating of all the relative addresses appearing in the segment introduced to the internal storage /segment readdressing/.

Some solutions of these problems are given below.

2. THE PROBLEM OF THE SEGMENT CHOICE

The area of the internal storage for the object program segments is determined at the given moment by the number of active /i.e. accessible from the program place which is just being executed - see [1]/ variables and by the size of arrays. Of course, this area decreases with each input to a block or procedure, and increase with

each output. If at the given moment the area is filled up with segments, so that there is no place for the segment called out to act, the decision should be taken, which segment should be replaced by the segment needed. This problem can be solved in several ways: the self-learning method /ATLAS/, the method of erasing the segment which was entered first /GIER/, the equivalent method of the latter - of a cyclic choice, and a random choice method. This paper does not aim at discussing the above methods. It is a complex problem, requiring many statistical considerations. We choose the random method that is in a certain sense a self-learning one. Namely, let us assume that there is in the program a loop passing through a certain number of segments, and let us consider two cases:

a. This loop is fully enclosed in the internal storage /together with the variables on which it operates/,
b. it is not so.

In the case a, all the segments through which the loop passes will be in the internal storage after several runs of the loop. In case b, the segments that are in the internal storage through which the loop does not pass, will be eliminated from the storage after a certain number of runs of the loop. This is indicated by some statistical considerations we are not mentioning here, and the remarks we made are only to explain in which sense the random method is self-learning. This fact and the simplicity of implementation decided about its acceptance to ALGOL in ZAM-ALFA computer.

1.1. Detailed dexcription of the solution of segment exchange

The program is divided into segments, each containing 128 words per 24 bits, and is located on a drum. The segments are numbered: 0,1, ... Words in the segment are also numbered: 0,1, ... 127. The number of the given segment is written in word No 0 of this segment. The list of segments on the drum begins at a certain determined place, the segments are of a determined length, therefore, the segment number is determined by its drum address and conversely. We form the so- called segment table, i.e. to every segment 1 word is assigned in which, during the run of the object program,

the address of the beginning will be written /i.e. the address of the zero word/ in the peripheral storage of this segment if it is in the internal storage. If it is not, the signal of its absence is written in the segment table in the word assigned to the absent segment /this signal is the address of the appropriate location of the program of segment exchange/. The words of the segment table are ordered in the internal storage according to the numeration of represented segments: word No. 0,1, ...

A controlling extra-code instruction /see [2] and [4]/ serves for communication between segments. The form of this instruction is SEG /n, m/, where n, m define the appropriate number of the segment and its word number, which is entered in. It is the so-called segment programmed jump. One can refer from one segment to another only by the instruction SEG. Jumps inside the segment /more generally: relative instructions/ will be discussed in section II.

To describe the algorithm of segment exchange, let us call the pair (x,y) internal storage area, where x, y are addresses and $x < y$, and let us introduce the following denotations:

pw_{max} - the last address of the internal storage,

L - the address of the first free location for the program i.e. the boundary between the area for data and the program /see [1]/,

γ - number of activated segments, where the segment is called activated if it is in the internal storage; and this fact is marked in the segment table. At the beginning $\gamma = 0$.

α - The address in the internal storage that causes the segment actually brought from the drum to enter the area $(\alpha, \alpha + 127)$,

δ - the smallest address of the activated segment in the internal storage, i.e. $\delta = pw_{max} - \gamma.128$. At the beginning $\delta = pw_{max}$,

SS[n] - the content of the n-th location in the segment table.

Now, the algorithm of segment exchange can be given:

BEGINNING OF ALGORITHM

1. Go to segment table to its n-th location. This results in the following action: if the called segment /number n/ is in the internal storage, the jump to the SS[n] + m location will be made. In the opposite case - go to point 2 of this algorithm.
2. Is the area (L, δ) big enough to comprise the segment? If not, go to point 4; if so - go to point 3.
3. $\gamma := \gamma + 1$; $\delta := \delta - 128$; $\alpha := \delta$; go to point 5.
4. Sample from numbers 1,2, ... γ one number, and denote it by k; $\alpha := pw_{max} - k.128$;
   Under the address α there is a zero word of a certain activated segment which is to be deleted, i.e. the called out segment is being entered to its place. The zero word of the deleted segment comprises its number. Let is be the number j. Write the address of this algorithm point 2 in the j location of the segment table. This makes the segment number j inactive. Go to point 5.
5. Write number α in the n-th location of the segment table. Bring from the drum the segment number n to the area $(\alpha, \alpha + 127)$ and readdress the segment brought /see p. II/. Go to location $\alpha + 1 + m$.

END OF ALGORITHM

As above said, the area destined for segments /i.e. area (L, pw_{max})/ decreases with each entry to a block or procedure and increases with each exit. In case of exit - nothing is to be done, as the fact of the (L, pw_{max}) area increase will be, if necessary used by the program of segment exchange, in order to introduce new segments to it. In case of entry - it should be checked, whether the decrease of the (L, pw_{max}) area did not provide the necesity to delete some segments from the internal storage /those which are below the boundary as the result of the increase of the boundary L/. Because of this, every increase of value L is not a simple addition, but it is made by means of a special programmed instruction ZWL p /"increase value L by p"/.

The algorithm defining ZWL is the following:

BEGINNING OF ALGORITHM

1. L := L + p; go to point 2.
2. Is $L \geqslant \delta$? If so, go to point 3, if not, go out from the Algorithm /return/.
3. At the address δ there is the zero word of the segment which is to be deleted. It contains the number j of this segment. Write the address of this algorithm point 2 of segment exchange in the j-th place of the segment table. This makes the segment numbered j inactive. Go to point 4.
4. $\gamma := \gamma - 1$; $\delta := \delta + 128$; If there is no place for segments, /i.e. $\gamma = 0$/, signalize storage overflow. In the opposite case go to point 2.

END OF ALGORITHM

3. SEGMENT READDRESSING

As already said, ZAM computers do not have hardware address modification by an instruction computer. Therefore let us build an appropriate program. Let the location number in the same segment be the parameter /address part/ of an instruction in the segment. This address will be called relative towards the beginning of the segment. In order to make the segment operate, the address of this segment beginning in the intenal storage should be added to all relative addresses.

The investigation of all instructions in the segment, in order to detect those with relative addresses, is most time consumming. Hence, there remains the problem of finding a good algorithm of chosing instructions with relative addresses. For the purpose of the described algorithm let us assume that instructions with relative addresses are connected in a chain. This implies the following form of an instruction with a relative address:

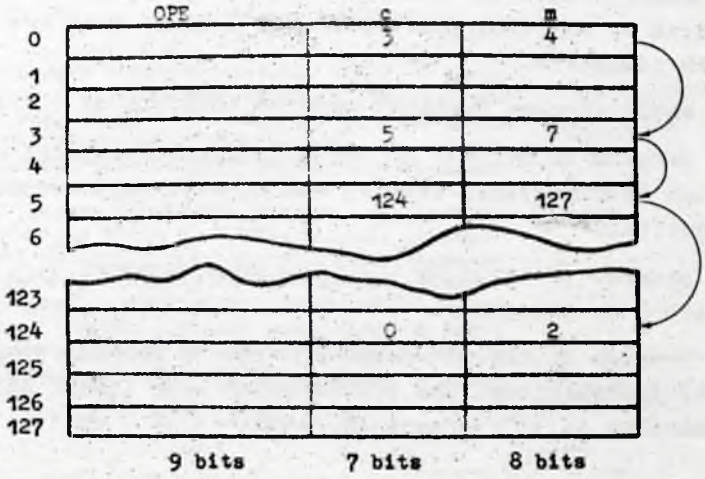
OPE/c,m/, where

OPE - is the name of the operational part

c - the relative address next in this segment of instruction with a relative address /the chain link/. If such a one does not exist then c = 0.

m - parameter of the operation OPE /i.e. relative address of the place where OPE acts/.

Moreover, we assume, that the segment zero word is of the same form, only m means the segment number, and OPE is arbitrary /e. g. 0/.



An example is given of segment number 4, having instructions with relative addresses 7, 127, 2 in locations 3, 5, 124. The algorithm of the segment readdressing is the following:

BEGINNING OF ALGORITHM

1. Take into account the zero word of the segment. Go to point 3.
2. Keep the value of the considered word. Form the word OPE + m and write it in the segment in the place of the word considered. OPE and m respectively are the operational part and its parameter in the considered word. Number α means the address of the beginning of the segment in the internal storage. Thus, we have the readdressed instruction. Go to point 3.

3. If the content of c in the considered word equals 0 /i.e. if there are no relative addresses in the segment/, finish the algorithm. Else, take into account the word numbered c. Go to point 2.

END OF ALGORITHM

FINAL NOTES

In order to make the effectiveness of the described method clear, some technical data of the segment exchange program are given below. First of all they are useful for readers that are familiar with ZAM computers

1. The whole segment exchange program consists of 62 instructions.
2. The program of point 1 of the algorithm of segment exchange, together with register keeping and reproducing, consists of 10 instructions.
3. The program of sampling a number among numbers 1,2, ... γ comprises 5 instructions.
4. The program of the algorithm of segment readdressing consists of 12 instructions. The readdressing loop itself chosing instructions to be readdressed, consists of 7 instructions.

I would like to communicate that the above described problems were developed together with Mr Piotr Szorc.

References

[1] CZAJA L., SZORC P.: Storage Allocation for ALGOL. Algorytmy No. 7, 1967.

[2] CZAJA L., SZORC P.: Implementation of ALGOL for ZAM Computers. Algorytmy No. 7, 1967.

[3] NAUR P.: Revised ALGOL 60. Report ACM vol. 6.

[4] ŁUKASZEWICZ L.: Informacje wstępne o rodzinie maszyn matematycznych ZAM. Prace Instytutu Maszyn Matematycznych, C 1/4/, 1965.

Instytut Maszyn Matematycznych
Algorytmy N° 9

TRANSLATOR JĘZYKA ALGOL DLA MASZYNY UMC-10

Jerzy LESZCZYŃSKI

Pracę złożono 14.11.1967 r.

Praca zawiera opis podzbioru języka ALGOL 60 przystosowanego do realizacji na maszynach matematycznych UMC-10 nazywanego UMC-ALGOL. Język UMC-ALGOL powstaje z podzbioru SUBSET-ALGOL-IFIP przez odrzucenie warunkowych wyrażeń arytmetycznych, przełączników oraz łańcuchów. Ponadto podano zwięzły opis translatora języka UMC-ALGOL wykonany dla maszyny UMC-10, zwracając uwagę na rozmieszczenie informacji w pamięci maszyny oraz sposób identyfikowania informacji zawarty w systemie wskaźników. Ponadto roczna eksploatacja translatora potwierdziła celowość wybranej wersji języka oraz użytych metod translacji, otwierając gronu użytkowników maszyn UMC-10 możliwość praktycznego korzystania z języka ALGOL.

WSTĘP

W Katedrze Budowy Maszyn Matematycznych Politechniki Warszawskiej został opracowany pod kierownictwiem autora niniejszego artykułu podzbiór języka ALGOL dla maszyn UMC 10 nazwany UMC-ALGOL oraz wykonany został translator tego języka dla maszyn UMC 10.

W wyborze podzbioru języka UMC-ALGOL za podstawę przyjęto zalecenia opracowane przez Międzynarodową Federację Przetwarzania Informacji określane jako podzbiór ALGOL 60 /IFIP/ [1]. Zasadniczą część prac nad wyborem podzbioru języka i programem translatora wykonano w 1966 roku. Pierwszy raport na temat języka UMC-ALGOL i translatora dla maszyn UMC 10 został zamieszczony w MASZYNACH MATEMATYCZNYCH Nr 2/1967 [2].

Blisko roczna eksploatacja translatora potwierdziła w pełni celowość wybranej wersji języka oraz użytych metod translacji, otwie-

rając szerokiemu gronu użytkowników maszyn UMC 10 możliwości efektywnego korzystania z międzynarodowego bogatego dorobku publikacyjnego w języku ALGOL 60 oraz współczesnych metod programowania zadań dań numerycznych.

2. JĘZYK UMC-ALGOL

Język UMC-ALGOL jest podzbiorem języka wzorcowego ALGOL 60 utworzonym przez nałożenie na język wzorcowy szeregu ograniczeń. Język UMC-ALGOL operuje alfabetem różnym od alfabetu języka wzorcowego, a dostosowanym do postaci urządzeń przygotowania taśmy dla maszyn UMC 10 /dalekopisy firmy LORENZ/.

Do alfabetu UMC-ALGOL należą litery, cyfry, znaki i symbole słowne:

- Litery od A do Z w zestawie występującym w kodzie międzynarodowym Nr 2 /zamiast liter dużych mogą być litery małe/.
- Cyfry od 0 do 9.
- Znaki działań arytmetycznych

  \+ - */.
- Znaki inne

  . ( ) : = ; , :
- Symbole słowne relacji

  'LESS' 'LESSEQ' 'EQUAL' 'NOT EQUAL'

  'GREATER' 'GREEQ'.
- Symbole słowne działań logicznych

  'NOT' 'OR' 'AND'.
- Symbole słowne wartości logicznych

  'TRUE' 'FALSE'.
- Symbole słowne operatorów następstwa 'GO TO' 'IF' 'THEN' 'ELSE' 'FOR' 'DO'.
- Symbole słowne przerywników

  'STEP' 'UNTIL' 'WHILE' 'COMMENT'.

- Symbole słowne nawiasów

  'BEGIN' 'END'.

- Symbole słowne mian

  'BOOLEAN' 'INTEGER' 'REAL' 'ARRAY' 'PROCEDURE'.

- Symbole słowne specyfikatorów

  'LABEL' 'VALUE' .

- Symbole słowne inne

  'READ' 'PRINT' 'WRITE' 'CODE W20' 'LINE' 'STOP'.

Język UMC-ALGOL ma ponadto ustalone /nie wymagające deklarowania/ nazwy:

EXP ABS SQRT SIN COS ARCTAN IN SIGN ENTIER
FROMDRUM TODRUM oraz DRUMPLACE.

Nazwy te nie są zastrzeżone.

Dla uwypuklenia indywidualnych cech składni UMC-ALGOL przedstawiono poniżej ograniczenia syntaktyczne w oparciu o opis języka wzorcowego ALGOL 60. Część ograniczeń pokrywa się z ograniczeniami przyjętymi w podzbiorze języka ALGOL 60 /IFIP/.

Opis ograniczeń nawiązuje do odpowiednich paragrafów opisu języka wzorcowego ALGOL 60 [3]. Raz opisane ograniczenie jest obowiązujące dla całego opisu.

2.1. Skreślić małe lub duże litery w zależności od rodzaju czcionek w posiadanych urządzeniach do przygotowywania taśmy.

2.3. Skreślić:

- operator arytmetyczny + ↑ ,
- " logiczny ≡ ⊃ ,
- nawias [ ] ‘ ’ ,
- miano 'OWN' 'SWITCH',
- specyfikator 'STRING',
- przerywnik ␣ 10,

Operatory relacji i pozostałe operatory logiczne zastępuje się słowami przytoczonymi w wykazie symboli słownych.

2.4. Długość nazw dowolna. Rozróżnia się nazwy poprzez identyfikację pięciu początkowych znaków.

2.5. W zakresie parametrów liczbowych pisanych w programie nie dopuszcza się zapisu liczb w postaci wykładniczej.

2.6. Dopuszcza się jedynie łańcuch prawidłowy jako komentarz przy opisach w nagłówkach procedur.

3.1. Zastępuje się nawiasy kwadratowe okrągłymi przy zmiennych ze wskaźnikami.

3.2. Łańcuch, nazwa przełącznika i nazwa procedury z parametrami nie może być parametrem aktualnym.

3.3. Nie dopuszcza się stosowania warunkowych wyrażeń arytmetycznych.

3.4. Symbole operatorów relacji i operatorów logicznych zastępuje się symbolami słownymi przytoczonymi w wykazie symboli słownych.

3.5. Dopuszcza się wyrażenia mianujące tylko w postaci etykiet. Nie dopuszcza się stosowania przełączników.

4.1. Bez zmian.

4.2. Bez zmian.

4.3. Instrukcja skoku tylko według etykiety, jako jedynego dopuszczalnego wyrażenia mianującego.

4.4. Bez zmian.

4.5. Bez zmian.

4.6. Instrukcja 'FOR' może posiadać tylko jeden element wykazu 'FOR' dowolnego rodzaju.

4.7. To samo ograniczenie co w p. 3.2.

Rodzaj i typ parametru aktualnego musi być zgodny z rodzajem i typem parametru formalnego. Nie dopuszcza się użycia procedur rekursywnych i rekursywnego wywołania procedur, z wyjątkiem rekursywnego wywołania procedur standartowych.

5.1. Skreślić typ lokalny własny.

5.2. Każda nazwa tablicy powinna mieć wymieniony wykaz par granicznych.

5.3. Skreślić w całości.

5.4. Zbiór specyfikacji nie może zawierać łańcuchów, przełączników oraz procedur z parametrami. Zbiór wartości nie może zawierać nazw zmiennych ze wskaźnikami. Zbiór specyfikacji winien w pełni odpowiadać zbiorowi parametrów formalnych zarówno pod względem ilości, jak i kolejności parametrów formalnych.
Nagłówek procedury jest jedną jednostką syntaktyczną i nie wolno dzielić go średnikami. W ciele procedury wolno używać tylko nazw poprzednio opisanych.
Wejście jest oznaczone symbolem słownym 'READ', a wyjście symbolem słownym 'PRINT'.

Z lewej strony tych symboli występuje nazwa zmiennej, której jest przyporządkowywana wartość lub której wartość jest drukowana.

Z prawej strony symbolu 'PRINT' występuje wzorzec. Wzorzec pisany jest w postaci liczby sześciocyfrowej /trzy członowej/, w której dwie pierwsze cyfry oznaczają ilość spacji przed wyprowadzoną liczbą minus ilość cyfr części całkowitej, dwie następne cyfry wzorca - ilość cyfr części ułamkowej /dla liczb typu real/, dwie ostatnie - ilość spacji po liczbie. Sposób drukowania wartości nie jest zależny od typu zmiennej, a od rodzaju wzorca. Jeśli dwie pierwsze cyfry wzorca są różne od 0, liczba zostaje wydrukowana w postaci dziesiętnej lub całkowitej. Jeśli dwie pierwsze cyfry wzorca są równe 0, liczba zostaje wydrukowana w postaci wykładniczej. Jeśli trzecia i czwarta cyfra jest równa 0, liczba jest drukowana jako całkowita, a w przypadku przeciwnym jako ułamkowa.

Wartości typu boolean pisane i drukowane są jako 0 lub 1.

Z prawej strony symbolu 'READ' pisana jest liczba służąca do rozróżniania urządzeń czytających.

Sposób zapisu czytanych wartości jest zależny od typu zmiennej.

Wprowadzenie instrukcji napisanych w kodzie W-20 jest realizowane przez 'CODE W20'. Treść programu napisanego w kodzie W-20 lokuje się między średnikami bezpośrednio po symbolu 'CODE W20'.

Do programu pisanego w języku UMC-ALGOL można dołączać procedury, których ciała są programami pisanymi w kodzie W20, z uwzględnieniem następujących uwag:

- wszystkie parametry formalne prooedury są umieszczone na liście wartości,
- tekst programu jest pisany w adresach względnych,
- adresy parametrów są określone względem liczby umieszczonej na części adresowej, na miejscu o adresie względnym - 1, w ten sposób, że liczbę tę należy traktować jako adres ostatniego parametru formalnego,
- adres przedostatniego parametru formalnego i pozostałych określony jest przez dodanie kolejno 1 do adresu parametru ostatniego,
- adres odpowiadający nazwie procedury określamy przez dodanie zwiększonej o jeden ilości parametrów formalnych do liczby zapisanej na miejscu o adresie względnym - 1.

W celu umożliwienia organizacji wydawnictwa przewidziano symbole:

'LINE' - zmiana linii,
'WRITE' - drukowanie tekstu.

Ostatni symbol umożliwia wstawienie w dowolne miejsce programu łańcucha znaków napisanych między średnikami bezpośrednio po symbolu 'WRITE'. Łańcuch ten zostaje wypisany w czasie wykonania programu.

3. TRANSLACJA I WYKONANIE PROGRAMU

- Maksymalna długość programu wynikowego ułożonego przez translator UMC-ALGOL wynosi około 2200 rozkazów w kodzie maszyny UMC.
- Maksymalna ilość nazw zmiennych wynosi około 90 bez zwijania struktury blokowej.

- Maksymalna ilość miejsc zajmowanych przez zmienne ze wskaźnikami w programie jest zależna od ilości użytych zmiennych oraz długości programu i jest uzupełnieniem do 2500 sumy tych dwóch wielkości.
- Maksymalna ilość parametrów stałych w programie wynosi 400 liczb typu integer albo 200 liczb typu real.
- Komunikacja z pamięcią bębnową odbywa się za pomocą procedur TODRUM i FROMDRUM oraz zmiennej DRUMPLACE wg zasad obowiązujących dla GIER-ALGOLu. Obszary pamięci bębnowej nie są zwijane wraz z blokami programu.

Symbole, adresy i opisy zmiennych, etykiet, procedur występujących w programie są drukowane w czasie translacji.

Symbole, adresy i opisy zmiennych, etykiet i procedur są drukowane w momentach zwijania bloków programu dla nazw o obszarze istnienia podlegającym likwidacji.

Drukowanych jest pięć początkowych /nie licząc spacji/ znaków nazwy, adres programu, adres miejsca roboczego i opisy w postaci kolejnych liter, w których:

R - oznacza typ real,
I - " " integer,
B - " " boolean,
S - " specyfikację,
A - " strukturę array,
P - " " procedure,
L - " miano label.

Na zakończenie procesu translacji drukowane są ostatnie adresy: programu, listy parametrów, listy miejsc roboczych oraz suma kontrolna programu i wypisywane jest hasło: OK.

Wykonanie ułożonego programu jest inicjowane przez przyciśnięcia litery P na klawiaturze dalekopisu - monitora.

Na zakończenie programu drukowane jest hasło: THANK YOU. Program można wykonywać wielokrotnie.

Translator UMC-ALGOL zawiera w sobie system operacyjny umożliwiający znalezienie określonych wartości w pamięci operacyjnej ma-

szyny oraz kontrolę i ewentualną zmianę tych wartości.

System operacyjny jest dostępny w fazie wykonywania programu tzn.:

- bezpośrednio po zakończeniu translacji programu po wypisaniu hasła: OK,
- w przypadku wypisania hasła: ERROR,
- po zakończeniu obliczeń programu i wypisaniu hasła: THANK YOU.

System operacyjny wywołuje się przez napisanie litery O na dalekopisie - monitorze.

Sprawdzenia wartości zmiennych użytych w programie można dokonać, posługując się tekstem drukowanym w czasie translacji programu, który podaje adres bezwzględny pamięci przypisywany określonej zmiennej oraz typ tej zmiennej.

System operacyjny drukuje wartość wskazanej adresem komórki pamięci po wypisaniu na monitorze tekstu:

adres, PR dla zmiennych typu real,
adres, PI dla zmiennych typu integer,
adres, PB dla zmiennych typu boolean.

Wprowadzenia dowolnej wartości pod wskazany adres komórki pamięci można dokonać przez system operacyjny wypisując teksty:

adres, RR wartość, dla zmiennych typu real,
adres, RI wartość, dla zmiennych typu integer,
adres, RB wartość, dla zmiennych typu boolean.

System operacyjny umożliwia ponadto, wstawienie lub skasowanie w dowolnym miejscu programu stopu warunkowego oraz dokonanie skoku do określonego miejsca w programie poprzez teksty:

adres, ST stop warunkowy,
adres, GO skok do miejsca programu.

W powyższych tekstach na miejscu adres i wartość należy wypisywać na monitorze rzeczywiste adresy pamięci i wartości wprowadzane do pamięci.

Translator wykrywa pewną ilość błędów formalnych i składniowych w programach napisanych w UMC-ALGOLu.

W przypadku wykrycia błędu w czasie translacji drukuje się hasło: FAIL oraz dwie liczby całkowite, umożliwiające identyfikację rodzaju błędu. W przypadku wykrycia błędu w czasie wykonywania obliczeń drukuje się hasło: ERROR oraz również dwie liczby całkowite, umożliwiające identyfikację rodzaju błędu.

W obydwu przypadkach praca maszyny ulega zatrzymaniu.

4. TRANSLATOR UMC-ALGOL

Translator UMC-ALGOL został wykonany dla maszyny UMC10 o długości słowa 34 bity, pamięci operacyjnej 4096 słów oraz pamięci bębnowej 16 384 słów [4].

Kolejne pozycje słowa maszyny numerowane są od strony prawej ku lewej od pozycji 1 do pozycji 34. Pozycja 1 jest najmniej znaczącą pozycją słowa, a pozycja 34 najwięcej znaczącą. Zgodnie z przyjętą notacją liczb przy podstawie - 2 nieparzyste pozycje słowa maszynowego związane są z ujemnymi, a parzyste pozycje z dodatnimi wagami binarnymi o wykładnikach równych zmniejszonemu o jeden numerowi pozycji.

Adresy pamięci operacyjnej są liczbami z przedziału - 1365 ÷ + 2730 notowanymi na bitach 1 ÷ 12 słowa maszynowego.

Adresy pamięci bębnowej są liczbami z przedziału - 5461 ÷ +10929 notowanymi na bitach 1 ÷ 14 słowa maszynowego. Pamięć bębnowa podzielona jest na sekcje po 2048 słów numerowane od 1 do 8, poczynając od sekcji zawierających adresy ujemne.

Translator UMC-ALGOL składa się z dwóch części:

części tłumaczącej program źródłowy na program wynikowy oraz części zawierającej podprogramy wykorzystywane przez program wynikowy w czasie jego wykonywania. Część pierwsza będąca właściwym translatorem w dalszym ciągu niniejszego opisu nazywana będzie mianem TRANSLATOR, a część druga mianem BIBLIOTEKA.

Każda z wymienionych wyżej części posiada własny administrator ulokowany w pamięci operacyjnej przez cały okres działania tejże części, obszar miejsc na blok wołany z bębna oraz obszar miejsc roboczych przenoszących informacje pomiędzy blokami.

Translator z administratorem oraz biblioteka z administratorem ulokowane są na bębnie, zajmując miejsca od - 1837 do + 2730, tj. sekcje 2, 3, 4. Sekcje 5, 6, 7, 8 pamięci bębnowej wykorzystywane są do pamiętania dużych zbiorów informacji w czasie wykonywania programu.

Translator dzieli się na 8 nierównych bloków. Administrator translatora jest ulokowany w pamięci operacyjnej przez cały czas trwania procesu tłumaczenia. Oprócz administratora w pamięci operacyjnej jest ulokowany zawsze jeden z 8 bloków translatora. Bloki translatora są przepisywane na jeden i ten sam obszar w pamięci operacyjnej. Procesem przepisywania steruje administrator translatora w zależności od rodzaju analizowanego fragmentu programu.

Ostatnią czynnością administratora translatora jest ściągnięcie z bębna administratora biblioteki na te same miejsca pamięci operacyjnej.

Biblioteka dzieli się na dwa nierówne bloki. Administrator biblioteki jest ulokowany w pamięci operacyjnej przez cały czas działania programu wynikowego /w czasie wykonywania obliczeń/. Oprócz administratora biblioteki w pamięci operacyjnej ulokowany jest zawsze blok biblioteki zawierający podprogramy arytmetyki i logiki programu wynikowego. Blok biblioteki zawierający podprogramy wprowadzania i wyprowadzania danych jest przepisywany każdorazowo przed użyciem odpowiedniego podprogramu, a następnie natychmiast zastępowany blokiem zawierającym podprogramy arytmetyki i logiki.

Translator ulokowany jest w pamięci bębnowej na miejscach 0 ÷ 2730 w następującej kolejności bloków:

Miejsce wolne	Administrator /blok 0/	Blok 8	Blok 7	Blok 6	Blok 5	Blok 4	Blok 3	Blok 2	Blok 1
+2730	+2572	+2487	+1505	+1356	+1158	+951	+454	+338	+96 0

Biblioteka ulokowana jest w pamięci bębnowej na miejscach -1837 ÷ 0 w następujące kolejności bloków:

Blok 1	Blok 2	Administrator /Blok 0/
0 -753	-1446	-1837

Od miejsca +2731 w górę w pamięci bębnowej lokowane są tablice bębnowe programu wynikowego /przez TODRUM, FROMDRUM/. Miejsca poniżej -1837 w pamięci bębnowej nie są wykorzystywane ani przez translator UMC-ALGOL ani przez program wynikowy.

W czasie translacji programu źródłowego rozmieszczenie informacji w pamięci operacyjnej przedstawia się następująco:

Administrator /Blok 0/	M. ROB. GŁÓWNE	Blok translatora	PARAMETRY PROGRAMU WYNIKOWEGO	PROGRAM WYNIKOWY	M. ROB. STRUKTURY	LISTA OPISÓW	M. ROB. CZYTANIA
-1365	-1280	-1200	-200 +201		+2401		+2600 +2730

W czasie wykonywania programu wynikowego rozmieszczenie informacji w pamięci operacyjnej przedstawia się następująco:

Administrator biblioteki /Blok 0/	Blok pdp algebraicznych lub blok pdp wydawniczych biblioteki	PARAMETRY PROGRAMU WYNIKOWEGO	PROGRAM WYNIKOWY	MIEJSCE ROBOCZE TABLIC /obszar dynamiczny/	MIEJSCA ROBOCZE PROGRAMU WYNIKOWEGO
-1365	-1000	-200	+201		+2730

M. ROB. GŁÓWNE - stanowią zbiór przełączników, liczników i innych pojedynczych miejsc roboczych translatora. Na końcu tego zbioru formowany jest stos operatorów deszyfratora arytmetyki.

M. ROB. STRUKTURY - zawierają stos opisujący strukturę programu. Stos ten jest zwijany i rozwijany zgodnie ze zwijaniem i rozwijaniem struktury blokowej programu, opisów procedur oraz układu instrukcji if, for. Stos ten ogranicza od góry długość programu wy-

nikowego. W programach rozbudowanych strukturalnie maksymalny stos nie osiąga na ogół adresu +2350.

M. ROB. CZYTANIA - lista symboli odczytanej sekwencji wyrażeń programu źródłowego podlegającej analizie. Analizie podlegają sekwencje zawarte pomiędzy sąsiednimi symbolami ";" lub "end". W czasie analizy na miejscach roboczych czytania budowany jest stos identyfikatorów deszyfratora arytmetyki.

LISTA OPISÓW - zawiera informacje odpowiadające opisom występującym w programie. Lista ta jest zwijana i rozwijana zgodnie ze zwijaniem i rozwijaniem struktury blokowej programu źródłowego. Na liście opisów lokowane są również informacje o etykietach i instrukcjach skoku goto występujących w programie. Na każdy opis, etykietę, instrukcję goto występujące w programie rezerwowane są dwie pozycje listy opisów.

PARAMETRY PROGRAMU WYNIKOWEGO - lista na której lokowane są parametry liczbowe występujące w programie. Początek listy wykorzystywany jest na lokowanie ogólnych informacji o programie oraz parametrów uniwersalnych a mianowicie na miejscach:

+200 - adres pierwszego miejsca wolnego za programem wynikowym. Adres ten określa jednocześnie bazę, na której są rozwijane w górę obszary dynamiczne dla tablic w czasie wykonywania programu.

+199 - adres pierwszego miejsca wolnego przed listą parametrów programu wynikowego.

+198 - parametr uniwersalny "0".

+197 - parametr uniwersalny "1".

+196 - adres określający długość listy miejsc roboczych programu wynikowego, która budowana jest w dół poczynając od adresu +2730.

+195 - adres początku obszaru pamięci bębnowej wykorzystywanego do pamiętania dużych zbiorów informacji w czasie wykonywania programu. Adres ten wyraża się liczbą +2731.

Parametry uniwersalne "0", "1" są wykorzystywane w całym programie wynikowym bez wielokrotnego zajmowania miejsc na liście parametrów.

Parametry wyrażone liczbami typu integer zajmują po jednym miejscu na liście parametrów. Parametry wyrażone liczbami typu real zajmują po dwa miejsca na liście parametrów, z których pierwsza zawiera ujemną część ułamkową, a druga wykładnik liczby. Konwersja liczb zapisanych w powyższej formie na konwencjonalną postać zmienno-przecinkową i jest pierwszą czynnością wykonywaną w bibliotece po wywołaniu programu. Liczby w konwencjonalnej postaci zmienno-przecinkowej zapisywane są na miejscach wykładników poprzedniego zapisu.

Na liście parametrów lokowane są również miejsca robocze instrukcji for ... step, które nie mogą być rozwijane i zwijane zgodnie ze strukturą blokową programu wynikowego. Na każdą instrukcję for ... step zajmowane są dwa miejsca na liście parametrów.

PROGRAM WYNIKOWY. Podczas przetwarzania informacji źródłowej /program w UMC-ALGOLu/, w sposób określony przez program translatora, powstaje program wynikowy zapisywany w pamięci w kodzie wewnętrznym maszyny. Sekwencje rozkazów w kodzie wewnętrznym maszyny odpowiadają poszczególnym fragmentom programu źródłowego. Program wynikowy zajmuje miejsca począwszy od komórki +201 i może sięgać aż do miejsc roboczych struktury to znaczy do adresu około +2350.

Poszczególne bloki translatora pełnią następujące funkcje:

Blok 1 - program wywołania translatora, formowanie i rozmieszczenie parametrów na MIEJSCACH ROBOCZYCH translatora, umieszczenie opisów podprogramów standartowych na LIŚCIE OPISÓW, druk na monitorze hasła: ALGOL,

Blok 2 - program czytający, formowania nazw i operatorów ze znaków dalekopisowych i lokowanie ich na MIEJSCACH ROBOCZYCH CZYTANIA, formowanie parametrów liczbowych i lokowanie ich na LIŚCIE PARAMETRÓW PROGRAMU WYNIKOWEGO,

Blok 3 - analiza błędów wykrytych w procesie translacji, druki kontrolne,

Blok 4 - analiza zawartości MIEJSC ROBOCZYCH CZYTANIA, formowanie stosu opisującego strukturę programu, lokowanie na LIŚCIE OPISÓW informacji zawartych w opisach integer, real, boolean, array, procedure oraz etykietach i instrukcjach goto, analiza nagłówka procedury,

Blok 5 - analiza stosu opisującego strukturę programu, zamykanie bloków programu i procedur, formowanie adresów dla instrukcji goto,

Blok 6 - analiza i zwijanie instrukcji if, for, formowanie odpowiednich fragmentów programu wynikowego,

Blok 7 - analiza łańcuchów o postaci ciągu instrukcji zapisanego w języku W20, formowanie odpowiednich fragmentów programu wynikowego,

Blok 8 - analiza wyrażeń arytmetycznych i logicznych, formowanie odpowiednich fragmentów programu wynikowego.

Zasadniczy ciąg programu translatora stanowią bloki 2, 4, 5, tworząc układ zamknięty, z którego wyjście następuje przy zamknięciu programu wynikowego przez przejście do biblioteki.

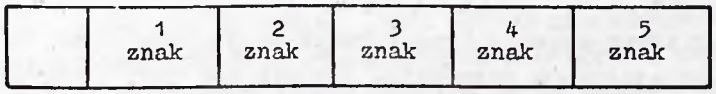
Bloki 7 i 8 komunikują się z blokiem 4 i są mu podporządkowane. Blok 6 komunikuje się z blokiem 5 i jest mu podporządkowany. Blok 3 jest wykorzystywany przez wszystkie pozostałe bloki programu translatora.

Instrukcje programu wynikowego są formowane we wszystkich blokach programu translatora, lecz olbrzymia ich większość powstaje w wyniku działania bloków 6 i 8.

Identyfikacja informacji ulokowanej na MIEJSCACH ROBOCZYCH oraz LIŚCIE OPISÓW translatora opiera się o system wskaźników identyfikacji określających jednoznacznie typ i rodzaj symbolu podlegającego identyfikacji.

Formowanie wskaźników identyfikacji oraz korzystanie z nich przeplata się wzajemnie w różnych blokach translatora. Kryteria interpretacji wskaźników są różne w różnych blokach translatora i wynikają z funkcji wykonywanych przez bloki translatora. Tak więc system wskaźników identyfikacji ma strukturę wieloszczeblową oraz wielopłaszczyznową w odniesieniu do bloków translatora i stadium analizy programu źródłowego. Jednakże pewne ogólne zasady systemu wskaźników identyfikacji obowiązują w całym procesie translacji.

MIEJSCA ROBOCZE CZYTANIA zawierają informacje odczytane z taśmy programu źródłowego i przekształcone przez program czytający. Na informacje te składają się NAZWY i OPERATORY. Słowo zawierające nazwę składa się z łańcucha znaków definiujących nazwę oraz identyfikującej nazwę jak typ informacji.



Każdy znak łańcucha definiującego nazwę zapisany jest na 6-ciu bitach słowa maszynowego. Nazwę jako typ informacji określa bit 31 = 1. Nazwę będącą symbolem języka /ujętą w nawiasy "'"/ określa bit 34 = 1 oraz bit 31 = 1.

Parametry liczbowe wprowadzane są przez program czytający bezpośrednio na LISTE PARAMETRÓW PROGRAMU WYNIKOWEGO, a na ich miejsce, na MIEJSCACH ROBOCZYCH CZYTANIA lokowane są specjalne nazwy. Nazwy te zamiast łańcucha znaków zawierają na części adresowej adres LISTY PARAMETRÓW, oraz bit 31 = 1 i dodatkowo:

- dla parametrów typu integer bit 30 = 1,
- dla parametrów typu real bit 29 = 1,
- dla parametrów typu boolean, tj. dla symboli true i false bit 28 = 1.

Słowo zawierające operator określa bit 31 = 0.

Na końcu sekwencji znaków programu źródłowego podlegającej analizie tzn. przy wczytaniu symbolu ";" lub "end" umieszczane jest słowo zawierające wszystkie pozycje wyzerowane.

LISTA OPISÓW zawiera informacje pozwalające na identyfikację nazw oraz instrukcji goto występujących w programie źródłowym. Początek LISTY OPISÓW zawiera informacje o nie wymagających deklarowania funkcjach i zmiennych języka UMC-ALGOL.

Każdy z opisów zawiera adres miejsca roboczego, część identyfikującą nazwę jako typ informacji oraz dla procedur drugi adres

określający początek procedury w programie wynikowym. Na miejsca robocze zajmowane są kolejne pozycje LISTY MIEJSC ROBOCZYCH PROGRAMU WYNIKOWEGO budowanej w dół poczynając od adresu +2730.

Pierwsze pozycje listy MIEJSC ROBOCZYCH PROGRAMU WYNIKOWEGO zajęte są na stałe przez PROGRAM WYNIKOWY a mianowicie na miejscach:

+2730 - licznik obszaru dynamicznego miejsc roboczych tablic występujących w programie. Licznik ten jest ustawiany w początkowej fazie wykonywania programu wynikowego na pozycję określającą adres pierwszego wolnego miejsca pamięci po programie wynikowym,

+2729 - miejsce robocze zmiennej własnej DRUMPLACE pełniącej rolę licznika obszaru pamięci bębnowej. Licznik ten jest ustawiany w początkowej fazie wykonywania programu wynikowego na pozycję określającą adres pierwszego miejsca obszaru pamięci bębnowej tj. wartości +2731.

Część identyfikująca nazwę jako typ informacji może zawierać następujące wskaźniki identyfikacji na LIŚCIE OPISÓW:

integer	- bit 30 = 1,
real	- bit 29 = 1,
boolean	- bit 28 = 1,
array	- bit 27 = 1,
procedure	- bit 26 = 1,
label	- bit 25 = 1,
specyfikacja	- bit 31 = 1,
etykieta	- bit 33 = 1,
instrukcja goto	- bit 34 = 1,
funkcja języka	- bit 32 = 1.

Wskaźniki identyfikacji występują pojedynczo lub w określonych niesprzecznych wewnętrznie kompletach. Wskaźnik procedury jest ustawiany w pozycji bit 26 = 1 dopiero w procesie zwijania procedury, co umożliwia realizację podstawień na nazwę procedury wewnątrz ciała procedury.

Mechanizm zwijania LISTY OPISÓW umożliwia dostęp do nazwy procedury po jej zwinięciu, czyniąc jednocześnie niedostępnymi zmien-

ne lokalne i parametry formalne procedury. Mechanizm ten zapewnia również sukcesywne przenoszenie informacji o instrukcjach goto do bloków zewnętrznych w przypadku braku odpowiednich etykiet w bloku zwijanym.

Na pierwszej pozycji LISTY OPISÓW ulokowane jest zero sygnalizujące początek listy w procesie jej przeszukiwania. Przeszukiwanie listy odbywa się od końca. LISTA OPISÓW przeszukiwana jest przy identyfikacji symboli występujących w programie źródłowym, poczynając od aktualnego stanu licznika aż do momentu identyfikacji symbolu według odpowiedniej pozycji listy lub znalezienia na liście symbolu "zero".

W wyniku przedstawionego wyżej sposobu przeszukiwania LISTY OPISÓW zapewnia się zachowanie obszarów istnienia nazw zgodne z hierarchią bloków programu i kolejnością występowania opisów.

Proces identyfikacji etykiet realizowany jest niezależnie od procesu identyfikacji innych symboli, co umożliwia stosowanie w tych samych blokach programu identycznych symboli nazw i etykiet.

Zwinięcie procedury powoduje przesunięcie licznika LISTY OPISÓW do pozycji leżącej powyżej opisu nazwy procedury. W ten sposób zostaje zachowana dostępność nazwy procedury w bloku szerszym, w którym nazwa jej może się pojawić w instrukcji wywołania procedury, a zapewniona jest niedostępność nazw parametrów formalnych i nazw lokalnych ciała procedury. Jednocześnie odzyskuje się dostępność symboli opisanych w szerszym bloku programu, a identycznych z symbolami parametrów formalnych procedury lub nazw lokalnych ciała procedury.

Zwinięcie bloku programu w zasadzie nie różni się niczym od zwinięcia ciała procedury poza tym, że to ostatnie musi uwzględniać zwinięcie opisów nazw parametrów formalnych procedury.

Literatura

[1] Report on SUBSET ALGOL-60 /IFIP/.

[2] Jerzy LESZCZYŃSKI, Roman JANKOWSKI, Jerzy SZEWCZYK: ALGOL DLA MASZYN UMC-10. Maszyny Matematyczne Nr 2, 1967.

[3] Stefan PASZKOWSKI: Język ALGOL 60. W-wa, 1965. Opis języka wzorcowego. Dodatek A.

[4] Jerzy SZEWCZYK: Uniwersalna maszyna cyfrowa UMC-10. Maszyny Matematyczne Nr 1, 1966.

ALGOL LANGUAGE TRANSLATOR FOR UMC-10 COMPUTERS

Summary

The paper contains the description of ALGOL 60 language subset adapted for implementation with UMC-10 computers, and called UMC-ALGOL. The UMC-ALGOL language is developed from the subset SUBSET-ALGOL-IFIP by means of rejecting conditional arithmetic expressions, switches and chains. Moreover, a brief description of the translator of UMC-ALGOL, realized for UMC-UMC-10 computer, is given, with consideration for the distribution of information in the computer storage and the way of identification of information, comprised in the indicator system..

The purposefulness of the chosen version of the language and the translation methods used, was ascertained by more than one year of the translator exploitation. This provides a large range of UMC-10 users with the possibility of using the ALGOL language in practice.

```
'begin''comment' przyklad nr 1 obliczanie rzeczywistych
                 pierwiastkow n wielomianow 2-go stopnia ;

   'real'a,b,c,delta  'integer'i,n; n'read'1 'write';

   a       b       c        x1       x2
                                      ;
   'for'i:=1 'step'1 'until'n 'do'
   'begin'
      'line'
      a'read'1; a'print'020200;
      b'read'1; b'print'020200;
      c'read'1; c'print'020202;

      delta:=b*b-4*a*c

      'if' delta'greeq'0
      'then'
      'begin'
         ((-b-sqrt(delta))/(2*a))'print'020200;
         ((-b+sqrt(delta))/(2*a))'print'020200
      'end'
      'else' 'write';nie ma pierw rzecz;
   'end';
'end'

p

+10,   -1,+1,+1,  -1,+2,+2,  -1,+3,+3,  -1,+4,+4,  -1,-1,-1,
       -2,-2,-2,  -3,-3,-3,  -4,+0,+1,  +1,+1,+0,  +2,+0,+0,

t
algol
     n      0  2723  i
     i      0  2724  i
delta       0  2725  r
     c      0  2726  r
     b      0  2727  r
     a      0  2728  r

   400   184  2710      4423733263
o k

    a       b       c        x1       x2

  -1.00   1.00   1.00      1.62  -0.62
  -1.00   2.00   2.00      2.73  -0.73
  -1.00   3.00   3.00      3.79  -0.79
  -1.00   4.00   4.00      4.83  -0.83
  -1.00  -1.00  -1.00  nie ma pierw rzecz
  -2.00  -2.00  -2.00  nie ma pierw rzecz
  -3.00  -3.00  -3.00  nie ma pierw rzecz
  -4.00   0.00   1.00      0.50  -0.50
   1.00   1.00   0.00     -1.00   0.00
   2.00   0.00   0.00      0.00   0.00

thank you
```

```
'begin' 'comment' przyklad nr 2 odejmowanie macierzy

   'procedure' subm(a,b,c,m,n) 'array'a,b,c'integer'm,n;
   'begin''integer'i,j;
      'write';


   roznica macierzy
                  ;
      'for'i:=1'step'1'until'm'do' 'line'
       'for'j:=1'step'1'until'n'do'
       'begin'
          c(i,j):=a(i,j)-b(i,j);
          c(i,j)'print'030300
       'end';
   'end';

   'procedure'in(a,m,n) 'array'a 'integer'm,n;
   'begin''integer'i,j;
      'for'i:=1'step'1'until'm'do'
       'for'j:=1'step'1'until'n'do'
         a(i,j)'read'1;
   'end';

   'begin''integer'k,l;
      k'read'1;
      l'read'1
      'begin' 'array'aa(1:k,1:l),bb(1:k,1:l),cc(1:k,1:l);
         in(aa,k,l);
         in(bb,k,l);
         subm(aa,bb,cc,k,l)
         'line'
         subm(cc,bb,aa,k-2,l)
       'end'
   'end'
'end'

p
+6,+4,

+10,+10,+10,+10,
+20,+20,+20,+20,
+30,+30,+30,+30,
+40,+40,+40,+40,
+50,+50,+50,+50,
+60,+60,+60,+60,

+1,+1,+1,+1,
+2,+2,+2,+2,
+3,+3,+3,+3,
+4,+4,+4,+4,
+5,+5,+5,+5,
+6,+6,+6,+6,
```

```
t
algol

    j      0  2721  i
    i      0  2722  i
    n   2723  2723  si
    m   2724  2724  si
    c   2725  2725  sra
    b   2726  2726  sra
    a   2727  2727  sra

    j      0  2708  i
    i      0  2709  i
    n   2710  2710  si
    m   2711  2711  si
    a   2712  2712  sra

   cc      0  2693  ra
   bb      0  2698  ra
   aa      0  2703  ra

    l      0  2704  i
    k      0  2705  i

   in    383  2713  p
 subm    209  2728  p

  650   184  2672    -4136064401

c k


   roznica macierzy

    9.000    9.000    9.000    9.000
   18.000   18.000   18.000   18.000
   27.000   27.000   27.000   27.000
   36.000   36.000   36.000   36.000
   45.000   45.000   45.000   45.000
   54.000   54.000   54.000   54.000


   roznica macierzy

    8.000    8.000    8.000    8.000
   16.000   16.000   16.000   16.000
   24.000   24.000   24.000   24.000
   32.000   32.000   32.000   32.000

thank you
```

COMPUTER THEORY

Instytut Maszyn Matematycznych
Algorytmy № 9

METHODS OF HIGH-SPEED MULTIPLICATION IN BINARY NUMBER SYSTEMS

by Pelagia WALIGÓRSKA
Bartłomiej GŁOWACKI
Andrzej ZIEMKIEWICZ

Received January 9th, 1968

Given a definition of the transformation of the multiplier binary notation corresponding to the improved method of ternary multiplication by shifting over 0's and 1's [1], [2], [3], and detailed evaluation of the average number of additions and subtractions for multiplying by this method. New easily implemented methods with an identic number of additions and subtractions are presented. Complete usefulness of the given methods to multiply numbers in sign-magnitude and two's complement representation is proved.

1. INTRODUCTION

In [1], [2], [3] the improved method of ternary multiplication by shifting over 0's and 1's is presented. The method is distinguished from among other known ones because of the fact that the sequence of operations of addition, subtraction and shifting, corresponding to the operation of multiplication, contains the smallest number of addition and subtraction operations. However, this method implementation is little economical because of a significant increase of hardware [2], [3].

The paper presents a definition of the transformation of the multiplier binary notation, which is the basis of the above mentioned method.

A detailed evaluation of the average number of additions and subtractions is given.

Starting from the above-mentioned definition, the method of multiplication by shifting one position is given. The number of additions and subtractions is the same as in the previous method. This method implementation is very easy.

By modifying the above method, a method of multiplication by shifting two positions was obtained. The number of additions and subtractions is the same as in the previous metho.. This modified method provides an almost twofold increase of multiplication speed, and requires a slight increase of hardware.

Complete usefulness of the given methods to the multiplication of numbers in two's complement representation is given. Moreover, it has been proved that the application of these methods allows to avoid the correction of the product, usually carried out in other methods.

2. MULTIPLICATION OF POSITIVE BINARY NUMBERS

2.1. Improved method of ternary multiplication by shifting over 0's and 1,s from right to left.

Let the multiplier be a positive binary number of the form

$$a = \sum_{i=0}^{n-1} a_i \, 2^{i-n} , \quad \text{where} \quad a_i \in \{0,1\} ,$$

which will be treated as a number with infinite representation

$$a = \sum_{i=-\infty}^{\infty} a_i \, 2^{i-n} , \qquad /1/$$

where $a_i \in \{0,1\}$ for $0 \leq i < n$,

$a_i = 0$ for $i < 0$ and $i \geq n$.

Let us define the transformation of the multiplier notation /1/ into the following notation

$$a = \sum_{i=-\infty}^{\infty} \alpha_i \, 2^{i-n} , \qquad /2a/$$

where

$$\alpha_i \in \{-1, 0, 1\} ,$$

which corresponds to the improved method of multiplication by shifting over 0's and 1,s [1], [2], [3].

If there exists such an indicator i, that $(a_{i+1}, a_i) = (1,1)$, we form the finite sequence of indicators $\{i_k\}$

$k = 0,1, \ldots, 2L+1$, where L is a number depending on representation /1/

thus:

$$i_0 = \min \{ i: (a_{i+1}, a_i) = (1,1) \},$$
$$i_1 = \min \{ i: (a_{i+1}, a_i) = (0,0), \; i > i_0 \},$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$i_{2l} = \min \{ i: (a_{i+1}, a_i) = (1,1), \; i > i_{2l-1} \},$$
$$i_{2l+1} = \min \{ i: (a_{i+1}, a_i) = (0,0), \; i > i_{2l} \},$$

where $l = 1,2, \ldots, L$, and L equals the greatest l, for which this iteration can be made.

Let

$$J = \bigcup_{l=0}^{L} J_l ,$$

where

$$J_l = \{ i: i_{2l} < i < i_{2l+1} , \qquad l = 0,1, \ldots, L \} ,$$

and

$$P = \{ i: i = i_{2l}, \qquad l = 0,1, \ldots, L \} ,$$
$$N = \{ i: i = i_{2l+1}, \quad l = 0,1, \ldots, L \} .$$

From the definition of the sequence i_k it results that $a_{i_{2l+1}} = 0$, thus from the identity

$$\sum_{p=i_{2l}}^{i_{2l+1}-1} a_p 2^p + \sum_{p=i_{2l}}^{i_{2l+1}-1} \bar{a}_p 2^p = 2^{i_{2l+1}} - 2^{i_{2l}},$$

where $\bar{a}_p$ is the complement a_p, one obtains

$$\sum_{p=i_{2l}}^{i_{2l+1}} a_p 2^p = \sum_{p=i_{2l}}^{i_{2l+1}-1} a_p 2^p = 2^{i_{2l+1}} + \sum_{p=i_{2l}}^{i_{2l+1}-1} (-\bar{a}_p) 2^p - 2^{i_{2l}} \qquad /3/$$

For any number a the dependence is valid

$$a = \sum_{i \notin J \cup P \cup N} a_i 2^{i-n} + \sum_{l=0}^{L} \sum_{p=i_{2l}}^{i_{2l+1}-1} a_p 2^{p-n}$$

moreover $\bar{a}_{i_{2l}} = 0$ from the definition of the sequence $\{i_k\}$. Thus, applying the identity /3/ one obtains

$$a = \sum_{i \notin J \cup P \cup N} a_i 2^{i-n} + \sum_{l=0}^{L} \left[2^{i_{2l+1}} + \sum_{p=i_{2l}+1}^{i_{2l+1}-1} (-\bar{a}_p) 2^{p-n} - 2^{i_{2l}} \right].$$

Representation /1/ is equivalent to representation /2a/ in which

$$\alpha_i = \begin{cases} a_i & \text{for} \quad i \notin J \cup P \cup N, \\ -\bar{a}_i = a_i - 1 & \text{for} \quad i \in J, \\ -1 = a_i - 2 & \text{for} \quad i \in P, \\ 1 = a_i + 1 & \text{for} \quad i \in N. \end{cases} \qquad /2b/$$

In a particular case, if such an indicator i does not exist that $(a_{i+1}, a_i) = (1,1)$, $J \cup N \cup P$ is an empty set, then $\alpha_i = a_i$ for every i.

Let us evaluate this method by computing the average number of additions and subtractions which are necessary to multiply two n-bit numbers.

It is proved in [5] that the number of k-bit sequences of similar digits /eg. only one's/ in 2^n different n-bit numbers is

$$N_n^k = \begin{cases} (3 + n - k)2^{n-(k+2)} & \text{for} \quad k < n, \\ 1 & \text{for} \quad k = n . \end{cases}$$

It is proved in [5] that the average number of additions and subtractions is the following

$$I_n = \begin{cases} 2^{-n}\left(N_n^1 + 2\sum_{k=2}^{n} N_n^k\right) & \text{for} \quad n \leqslant 4 , \\ 2^{-n}\left(N_n^1 + 2\sum_{k=2}^{n} N_n^k - 2^{n-3}\sum_{i=2}^{n-3}\sum_{k=1}^{E\left(\frac{n-i-1}{2}\right)} 2^{-2k}\right) & \text{for} \quad n > 4 , \end{cases}$$

where

$2^{-n} N_n^1$ - is the average number of isolated one's. Every such a one requires at most one operation,

$2^{-n}\sum_{k=2}^{n} N_n^k$ - is the average number of sequences composed of at least two one's. Each such sequence requires at most two operations,

$2^{-3}\sum_{i=2}^{n-3}\sum_{k=1}^{E\left(\frac{n-i-1}{2}\right)} 2^{-2k}$ - is the average number of saved operations for both former cases, when $a_i = 0$ and $i \in J$.

After computing, we obtain

$$I_n \approx \begin{cases} \frac{3}{8} n + \frac{1}{4} & \text{when} \quad n \leqslant 4 , \\ \frac{1}{3} n + \frac{4}{9} & \text{when} \quad n = 2k, \ k = 3,4, \ldots, \\ \frac{1}{3} n + \frac{155}{288} & \text{when} \quad n = 2k-1, \ k = 3,4, \ldots, \end{cases}$$

Average number of addition and subtraction for some number lengths are given below

n	16	24	32	48	56	64
I_n	5,75	8,45	11,15	16,45	19,15	21,85

2.2. Ternary multiplication by shifting one position from right to left

Let $\chi_A(x)$ denote the characteristic function of the set A, namely

$$\chi_A(x) = \begin{cases} 0 & \text{for} \quad x \notin A \\ 1 & \text{for} \quad x \in A \,. \end{cases}$$

As sets J, P and N are disjoint, the dependence /2b/ can be written in the equivalent form

$$\alpha_i = a_i - \chi_J(i) - 2\chi_P(i) + \chi_N(i) \,. \qquad /2c/$$

Let $\{\beta_i\}$ be a sequence of digits belonging to the set $\{-1,0,1\}$ defined from equations

$$\begin{aligned} a_i + p_{i-1} &= \beta_i + 2p_i \qquad \text{where} \qquad p_i \in \{0,1\} \quad \text{and} \\ p_i &= a_{i+1}\, a_i + p_{i-1}(a_{i+1}\, a_i)^2 \qquad \text{for} \quad i \geqslant 0 \,, \qquad /4/ \\ p_i &= 0 \qquad\qquad\qquad\qquad\quad \text{for} \quad i < 0 \,. \end{aligned}$$

It is proved in [5] that for p_i determined by formulae /4/

$$p_i = \chi_{J \cup P}(i)$$

is valid.

Theorem 1: For α_i determined by formulae /2c/ and for β_i determined by formulae /4/ $\alpha_i = \beta_i$ holds for every i.

Proof. Because

$$\alpha_i = a_i - \chi_J(i) - 2\chi_P(i) + \chi_N(i) ,$$

and $\beta_i = a_i + p_{i-1} - 2p_i$,

it suffices to prove that

$$p_{i-1} - 2p_i = -\chi_J(i) - 2\chi_P(i) + \chi_N(i)$$

for every i.
The dependence

$$(i-1) \in J \cup P$$

is equivalent to the dependence

$$i \in J \cup N ,$$

thus

$$\begin{aligned} p_{i-1} - 2p_i &= \chi_{J\cup P}(i-1) - 2\chi_{J\cup P}(i) = \\ &= \chi_{J\cup N}(i) - 2\chi_{J\cup P}(i) = \\ &= \chi_N(i) - 2\chi_P(i) - \chi_J(i) \end{aligned}$$

This completes the proof.

From theorem 1 results the dependence

$$a = \sum_{i=-\infty}^{+\infty} \alpha_i \, 2^{i-n} , \quad \text{where}$$

$$\alpha_i = \beta_i = a_i(1-2a_{i+1}) + p_{i-1}(1-2a_i)(1-2a_{i+1}) . \qquad /5/$$

If the described transformation is applied to the notation of the multiplier, the multiplication cycles in which the addition or subtraction of the multiplicand with shifting of the partial product, or only shifting of the partial product occurs, can be determined by the following Boolean expressions

$$\begin{aligned}
&\text{ADDITION} && = \overline{a_{i+1}} \wedge \left(a_i \dot{-} p_{i-1}\right),\\
&\text{SUBTRACTION} && = a_{i+1} \wedge \left(a_i \dot{-} p_{i-1}\right),\\
&\text{SHIFTING} && = a_i \equiv p_{i-1}\\
& && p_i = a_{i+1} \wedge a_i \vee p_{i+1} \wedge \left(a_{i+1} \dot{-} a_i\right), \qquad /6/
\end{aligned}$$

where

$$\left(a \equiv b\right) \equiv \left(\overline{a \dot{-} b}\right) \equiv a \wedge b \vee \bar{a} \wedge \bar{b}\,.$$

The execution of subtraction of the multiplicand in a sign and magnitude representation requires a lengthening of the adder, which simultaneously decreases the multiplication speed. These difficulties can be avoided if a complementary representation is applied for negative partial product during the execution of multiplication. The shifting of partial product is made according to the rules of complementary arithmetic. In order to distinguish the form of a partial product, an additional "sign" bit is not needed as the partial product is negative if and only if the last arithmetic operation performed is the subtraction of the multiplicand. The final product is always a positive number, as the most significant digit α_i being different from zero, is always positive.

The average time of multiplication by this method can be computed from the formula

$$T = I_n\, t_D + \left(n - I_n + 1\right) t_p = \left(n+1\right) t_p + I_n \left(t_D - t_p\right)$$

where t_D – is the time of addition or subtraction,
t_p – is the time of shifting by one position,
I_n – is the average number of additions and subtractions given in 2.1.

This method allows to speed up the multiplication without raising the costs of implementation.

2.3. Multiplication by shifting two positions from right to left

This method derives from the one given in the previous section because the following theorem is true

Theorem 2. For any number a with representation /5/

$$\alpha_{i+1} \cdot \alpha_i = 0$$

holds for every i.

Proof. For every i

$a_i^2 = a_i$, $p_i^2 = p_i$, and

$$\alpha_i = \left(1 - 2a_{i+1}\right)\left[a_i + p_{i-1}\left(1 - 2a_i\right)\right],$$

$$\alpha_{i+1} = \left(1 - 2a_{i+2}\right)\left\{a_{i+1} + \left(1-2a_{i+1}\right)\left[a_{i+1}\, a_i + p_{i-1}\left(a_{i+1}+a_i-2a_{i+1}a_i\right)\right]\right\}$$

$$= \left(1 - 2a_{i+2}\right)\left[a_{i+1}\left(1 - a_i\right) + p_{i-1}\left(a_i - a_{i+1}\right)\right]$$

Thus

$$\alpha_i \alpha_{i+1} = \left(1-2a_{i+2}\right)\left(1-2a_{i+1}\right)\left\{a_{i+1}\, a_i\left(1-a_i\right) + p_{i-1}\left[a_i\left(a_i-a_{i+1}\right) + \right.\right.$$

$$\left.\left. + a_{i+1}\left(1-a_i\right)\left(1-2a_i\right) + \left(1-2a_i\right)\left(a_i - a_{i+1}\right)\right]\right\} =$$

$$= \left(1-2a_{i+2}\right)\left(1-2a_{i+1}\right) p_{i-1}\left(a_i-a_i a_{i+1}+a_{i+1}-a_i a_{i+1}-a_i-a_{i+1}+2a_i a_{i+1}\right)$$

Thus the proof is complete.

The notation of number a can be transformed as follows

$$a = \sum_{i=0}^{n} \alpha_i\, 2^{i-n} = \sum_{m=0}^{M} \left(\alpha_{2m}\, 2^{2m-n} + \alpha_{2m+1}\, 2^{2m+1-n}\right) =$$

$$= \sum_{m=0}^{M} 2^{2m-n}\left(\alpha_{2m} + 2\alpha_{2m+1}\right) = \sum_{m=0}^{M} o_m\, 4^{m-M},$$

where

$$M = E\left(\frac{n}{2}\right) + 1$$

$$o_m = \alpha_{2m} + 2\alpha_{2m+1}.$$

According to theorem 2

$$o_m \in \left\{-2, -1, 0, 1, 2\right\}.$$

Let us compute the value of digits c_m depending on digits a_i

$$c_m = a_{2m}\left(1-2a_{2m+1}\right) + p_{2m-1}\left(1-2a_{2m}\right)\left(1-2a_{2m+1}\right) +$$

$$+ \left(1-2a_{2m+2}\right)\left[a_{2m+1}\left(1-a_{2m}\right) + p_{2m-1}\left(a_{2m} - a_{2m+1}\right)\right],$$

$$p_{2m+1} = a_{2m+1}\, a_{2m+2} + p_{2m}\left(a_{2m+2} - a_{2m+1}\right)^2 .$$

Denote

$$p_{2m+1} = q_m .$$

After transforming we obtain

$$c_m = a_{2m}\left(1 - 4a_{2m+1} + 4a_{2m+1}\, a_{2m+2}\right) +$$

$$+ 2a_{2m+1}\left(1-2a_{2m+2}\right) + q_{m-1}\left(1-2a_{2m+2}\right)\left(1-2a_{2m+1}\right),$$

$$q_m = a_{2m+1}\left(a_{2m} + a_{2m+2} - a_{2m}\, a_{2m+2}\right) +$$

$$+ q_{m-1}\left(a_{2m} - a_{2m}\, a_{2m+1} + a_{2m}\, a_{2m+2} - a_{2m+1}\, a_{2m+2}\right) \quad \text{for} \quad m \geq 0,$$

$$q_m = 0 \qquad \text{for} \quad m < 0$$

Multiplication cycles in which the addition or subtraction of the multiplicand with shifting of the partial product occurs, addition or subtraction of twice the multiplicand with shifting of the partial product or only shifting of the partial product, can be determined by the following Boolean expressions

$$\text{ADDITION} = \overline{a_{2m+1}} \wedge \left(a_{2m} \dot{-} q_{m-1}\right),$$

$$\text{SUBTRACTION} = a_{2m+1} \wedge \left(a_{2m} \dot{-} q_{m-1}\right),$$

$$\text{TWICE MULTIPLICAND ADDITION} = \overline{a_{2m+2}} \wedge \left(a_{2m} \equiv q_{m-1}\right),$$

$$\text{TWICE MULTIPLICAND SUBTRACTION} = a_{2m+2} \wedge \left(a_{2m} \equiv q_{m-1}\right),$$

$$\text{SHIFTING} = a_{2m+1} \wedge a_{2m} \wedge q_{m-1} \vee \overline{a_{2m+1}} \wedge \overline{a_{2m}} \wedge \overline{q_{m-1}}$$

$$q_m = a_{2m+1} \wedge \left(a_{2m+2} \vee a_{2m}\right) \vee q_{m-1} \wedge \left(a_{2m+2} \wedge a_{2m} \vee a_{2m+1}\right).$$

The average time of multiplication by this method can be computed from the formula

$$T = I_n t_D + (M - I_n) t_p = I_n (t_D - t_p) + E\left(\frac{n}{2} + 1\right) t_p \ .$$

All given methods can be used to multiply numbers in sign and magnitude representation.

Multiplication of absolute values is executed by one of the above-mentioned methods, and the sign of the result is determined according to the sign algebra.

3. MULTIPLICATION OF BINARY NUMBERS IN TWO's COMPLEMENT REPRESENTATION

In many computers the two's complement number representation is used.

The presented methods can also be applied for multiplication in this number representation.

We shall prove that the application of these methods does not require any product correction. Two's complement representation of numbers is the mapping of the number A onto the number $\varphi(A)$ determined as follows

$$\varphi(A) = \begin{cases} |A| & \text{when} \quad A \geqslant 0 \\ 2 - |A| & \text{when} \quad A < 0 \ , \end{cases}$$

φ is a one to one function which maps the interval $\langle -1, 1 \rangle$ onto the union of intervals $\langle 0, 1) \cup (1, 2)$. To every arithmetic operation corresponds its image in the set of numbers in two's complement representation, i.e. an operation that satisfies the condition

$$\varphi(A \square B) = \varphi(A) \triangle \varphi(B) \ , \qquad /8/$$

where "$\square$" is an operation on numbers in sign and magnitude representation and "$\triangle$" an operation on numbers in two's complement representation.

For further consideration let us assume that

$$|A| = a,$$

number $a \in \langle 0,1)$ has the form

$$a = \sum_{i=0}^{n} a_i^+ \, 2^{i-n} \qquad \text{where} \qquad a_i^+ \in \{0,1\} \ ,$$

number $(2-a) \in (1,2)$ has the form /9/

$$2-a = \sum_{i=0}^{n} a_i^- \, 2^{i-n} \qquad \text{where} \qquad a_i^- \in \{0,1\} \ ,$$

number $x \in (-1,1) \cup (1,2)$ has the form

$$x = \sum_{i=0}^{n} x_i \, 2^{i-n} \qquad \text{where} \qquad x_i \in \{-2,-1,0,1,2\}$$

In the product ab, a is the multiplier, and b is the multiplicand.

Definition 1. Let "$*$" denote an operation on numbers defined by equalities /9/ that satisfies the conditions

$$2^{-k} * a = 2^{-k} a \qquad \text{for} \quad 0 \leqslant k \leqslant n \ ,$$

$$2^{-k} * (2-a) = 2 - 2^{-k} a$$

$$x * a = \Big(\sum_{i=0}^{n} x_i \, 2^{i-n} \Big) * a = \sum_{i=0}^{n} x_i \, 2^{i-n} * a = \sum_{i=0}^{n} x_i \, 2^{i-n} \, a = xa$$

$$x * (2-a) = \Big(\sum_{i=0}^{n} x_i \, 2^{i-n} \Big) * (2-a) = \sum_{i=0}^{n} x_i \, 2^{i-n} * (2-a) =$$

$$= \sum_{i=0}^{n} x_i \left(2 - 2^{i-n} a\right) \equiv (2 - x a) \pmod 2 .$$

From this definition we immediately obtain

Theorem 3.

For numbers $a \in \langle 0,1)$, $b \in \langle 0,1)$, $(2-a) \in (1,2)$, $(2-b) \in (1,2)$ the following equalities are true

$$\begin{aligned} a * b &\equiv (ab) \pmod 2 , \\ a * (2-b) &\equiv (2 - ab) \pmod 2 , \\ (2 - a) * b &\equiv (2b - ab) \pmod 2 , \\ (2 - a) * (2 - b) &\equiv (2 - 2b + ab) \pmod 2 . \end{aligned}$$

Therefore, the operation "*" is not the image /in the sense /8// of multiplication in the set of numbers in two's complement representation.

Representation /9/ can be treated as infinite representation:

$$a = \sum_{i=0}^{n} a_i^+ 2^{i-n} \equiv \Big(\sum_{i=-\infty}^{\infty} a_i^+ 2^{i-n} \Big) \pmod 2 ,$$

where $a_i^+ \in \{0,1\}$ for $0 \leq i < n$; $a_i^+ = 0$ for $i < 0$, and $i \geq n$,

$$2 - a = \sum_{i=0}^{n} a_i^- 2^{i-n} \equiv \Big(\sum_{i=-\infty}^{\infty} a_i^- 2^{i-n} \Big) \pmod 2$$

where $a_i^- \in \{0,1\}$ for $0 \leq i < n$; $a_i^- = 0$ for $i < 0$, $a_i^- = 1$ for $i \geq n$

We shall prove the following property of the notation described by formulae /5/.

Theorem 4

If α_i^+ and α_i^- denote the i-th digits of infinite representations of numbers a and 2-a given in the notation described by formulae /5/ then

$$\alpha_i^- = -\alpha_i^+ \qquad \text{for every} \quad i .$$

Proof. Let

$$i_M = \min \left\{ i : a_i = 1 \right\} ,$$

then

$$a_i^- = \begin{cases} a_i^+ = 0 & \text{for} \quad i < i_M \\ a_i^+ = 1 & \text{for} \quad i = i_M \\ 1 - a_i^+ & \text{for} \quad i > i_M . \end{cases}$$

For $i < i_M$ the theorem is evident.
For $i \geqslant i_M$ the proof will be made by induction

$$\alpha_{i_M}^+ = 1 - 2\, a_{i_M+1}^+ , \qquad p_{i_M}^+ = a_{i_M+1}^+ ,$$

$$\alpha_{i_M}^- = 1 - 2\, a_{i_M+1}^- = 1 - 2 \left(1 - a_{i_M+1}^+\right) =$$

$$= - \left(1 - 2\, a_{i_M+1}^+\right) = - \alpha_{i_M}^+ ,$$

$$p_{i_M+1}^- = a_{i_M+1}^- = 1 - a_{i_M+1}^+ = 1 - p_{i_M}^+ .$$

Assume that

$$\alpha_i^- = - \alpha_i^+ ,$$

and

$$p_i^- = 1 - p_i^+ ,$$

for

$$i_M \leqslant i \leqslant k .$$

We shall prove that

$$\alpha_{k+1}^- = - \alpha_{k+1}^+ ,$$

$$p_{k+1}^- = 1 - p_{k+1}^+ .$$

From formulae /5/

$$\alpha^+_{k+1} = \left(1 - 2a^+_{k+2}\right)\left[a^+_{k+1} + p^+_k\left(1 - 2a^+_{k+1}\right)\right],$$

$$p^+_{k+1} = a^+_{k+2}\, a^+_{k+1} + p^+_k\left(a^+_{k+1} - a^+_{k+2}\right)^2$$

$$\alpha^-_{k+1} = \left(1 - 2a^-_{k+2}\right)\left[a^-_{k+1} + p^-_k\left(1 - 2a^-_{k+1}\right)\right] =$$

$$= -\left(1 - 2a^+_{k+2}\right)\left[1 - a^+_{k+1} - 1 + 2a^+_{k+1} + p^+_k\left(1 - 2a^+_{k+1}\right)\right] =$$

$$= -\left(1 - 2a^+_{k+2}\right)\left[a^+_{k+1} + p^+_k\left(1 - 2a^+_{k+1}\right)\right] = -\alpha^+_{k+1},$$

$$p^-_{k+1} = a^-_{k+2}\, a^-_{k+1} + p^-_k\left(a^-_{k+1} - a^-_{k+2}\right)^2 =$$

$$= \left(1 - a^+_{k+2}\right)\left(1 - a^+_{k+1}\right) + \left(1 - p^+_k\right)\left[\left(1 - a^+_{k+1}\right) - \left(1 - a^+_{k+2}\right)\right]^2 =$$

$$= 1 - \left[a^+_{k+2}\, a^+_{k+1} + p^+_k\left(a^+_{k+1} - a^+_{k+2}\right)\right] = 1 - p^+_{k+1}.$$

This completes the proof.

Conclusion 1.

$$\sum_{i=0}^{n} \alpha^-_i\, 2^{i-n} = \sum_{i=0}^{n} \left(-\alpha^+_i\right) 2^{i-n} = -\sum_{i=0}^{n} \alpha^+_i\, 2^{i-n} = -a$$

Let α denote the transformation which maps the sum

$$\sum_{i=0}^{n} a^+_i\, 2^{i-n},$$

onto the sum

$$\sum_{i=0}^{n} \alpha^+_i\, 2^{i-n},$$

and the sum

$$\sum_{i=0}^{n} a^-_i\, 2^{i-n},$$

onto the sum

$$\sum_{i=0}^{n} \alpha_i^- \, 2^{i-n} ,$$

where α_i^+ and α_i^- denote i-th digits of infinite representations of numbers a and 2-a, and where $a \in \langle 0,1)$.
On the basis of conslusion 1

$$\alpha(a) = a ,$$
$$\alpha(2-a) = -a ,$$

thus $$\alpha(x) = \begin{cases} x & \text{for} \quad x = a \\ x-2 & \text{for} \quad x = 2 - a \end{cases}$$

where $x \in \langle 0,1) \cup (1,2)$.

Hence, it results

Conclusion 2. The transformation α is the inverse transformation of the transformation φ, otherwise $\alpha(x) = \varphi^{-1}(x)$ for all $x \in \langle 0,1) \cup (1,2)$.
Similarly, let γ denote the transformation which maps the sum $\sum_{i=0}^{n} a_i^+ \, 2^{i-n}$ onto the sum $\sum_{m=0}^{M} c_m^+ \, 2^{2(m-M)}$, and the sum $\sum_{i=0}^{n} a_i^- \, 2^{i-n}$ onto the sum $\sum_{m=0}^{M} c_m^- \, 2^{2(m-M)}$.

Because $$c_m = \alpha_{2m} + 2\alpha_{2m+1}$$

thus $$c_m^- = \alpha_{2m}^- + 2\alpha_{2m+1}^- = -\left(\alpha_{2m}^+ + 2\alpha_{2m+1}^+\right) = -c_m^+ .$$

Hence $$\gamma(x) = \begin{cases} x & \text{for} \quad x = a \\ x-2 & \text{for} \quad x = 2-a , \end{cases}$$

which signifies that the transformation γ is the inverse transformation of φ.

Conolusion 3. The transformation $\gamma(x) = \varphi^{-1}(x)$ for all $x \in \langle 0,1) \cup (1,2)$.

Definition 2. Let "$\odot$" denote an operation on numbers in two's complement representation satisfying the condition
$x \odot y = \varphi^{-1}(x) * y$.

Theorem 5. The operation "$\odot$" is the image /in the sense /8// of the multiplication in the set of numbers in two's complement representation.

Proof.

$$a \odot b = \varphi^{-1}(a) * b = a * b \equiv (a\ b) \pmod 2 = \varphi(a\ b) ,$$

$$a \odot (2-b) = \varphi^{-1}(a) * (2-b) = a * (2-b) \equiv (2-ab) \pmod 2 =$$
$$= \varphi\left[a\ (-b)\right] ,$$

$$(2-a) \odot b = \varphi^{-1}(2-a) * b = (-a) * b \equiv (2-ab) \pmod 2 =$$
$$= \varphi\left[(-a)\ b\right] ,$$

$$(2-a) \odot (2-b) = \varphi^{-1}(2-a) * (2-b) = (-a) * (2-b) \equiv$$
$$= 2 - (-a)\ b \pmod 2 \equiv (a\ b) \pmod 2 =$$
$$= \varphi\left[(-a)\ (-b)\right] .$$

From theorem 5 and conclusions 2 and 3 it immediately follows

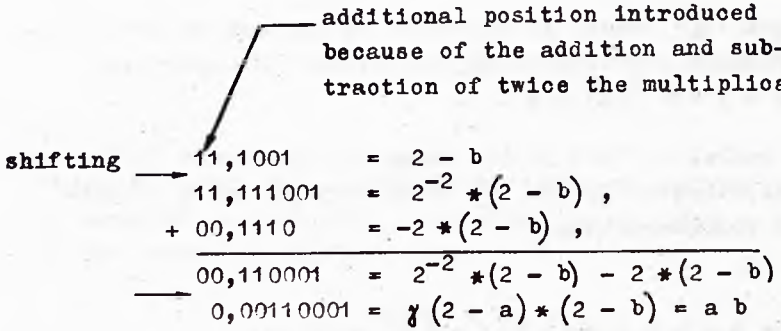
Conclusion 4. Operations

$$x \odot y = \alpha(x) * y ,$$
$$x \odot y = \gamma(x) * y ,$$

are images /in sense /8// of multiplication in the set of numbers in two's complement representation.

Example.

$$2 - a = 1,1001$$
$$2 - b = 1,1001$$
$$\gamma(2-a) = 0,\bar{2}1 , \quad \text{where } \bar{2} \text{ means } -2,$$



4. FINAL NOTES

The method of number multiplication given in section 2.3 has been applied in the design of the arithmetic unit, implemented on silicon basic circuits [6].

The authors thank docent A. W. Mostowski and docent K. Fiałkowski for their valuable remarks about the paper manuscript.

References

[1] AKUSHKY A.I. ... ed al.: Methods of Speeding-up the Operation of Digital Computers. UNESCO /NS/ICIP/E2.5/1959/.

[2] MAC SORLEY O.L.: High-speed Arithmetic in Binary Computers. Proc. IRE vol. 49 /January 1961/, pp. 67-91.

[3] FLORES I: The Logic of Computer Arithmetic . Prentice Hall /Englewood Cliffs - 1963/ pp. 151 - 245.

[4] BOOTH A.D., BOOTH K.V.H.: Automatic Digital Calculators. Buttenworth /London - 1956/ pp. 45 - 48.

[5] GŁOWACKI B., WALIGÓRSKA P., ZIEMKIEWICZ A.: Metody przyśpieszonego mnożenia w binarnych systemach liczbowych. Prace IMM No 2, 1968.

[6] KOJEMSKI A., KOWALEWSKA M., KULIŃSKA E., ŚWIATKOWSKI Z.: Szybkie tranzystorowe układy logiczne S-50 - opis ogólny. Prace IMM, Sprawozdania T1, Nr. 4, 1965.

Instytut Maszyn Matematycznych
Algorytmy Nº 9

O PEWNEJ METODZIE WYZNACZANIA ILORAZU W URZĄDZENIACH CYFROWO - ANALOGOWYCH

Jacek BAŃKOWSKI
Konrad PIAŁKOWSKI

Pracę złożono 19.12.1966 r.

W pracy przedstawiono metodę normalizacji zakresu dzielnej i dzielnika, umożliwiającą zrealizowanie stosunkowo prostego, a równocześnie dostatecznie dokładnego układu dzielenia w urządzeniach cyfrowo-analogowych oraz oszacowano dokładność ilorazu, jaką można uzyskać w takim układzie.

1. WSTĘP

W urządzeniach analogowych realizacja operacji dzielenia napotyka na poważne trudności. Trudności te wynikają głównie z faktu, że nie jest znany element fizyczny o dwóch wejściach, który na wyjściu dawałby dostatecznie dokładny sygnał, proporcjonalny do ilorazu sygnałów wejściowych. Poza tym, gdy wartość dzielnika jest mała, dokładność takiego urządzenia jest na ogół nie do przyjęcia.

Mała dokładność urządzeń analogowych spowodowała, że często łączy się je z urządzeniami cyfrowymi, które pozwalają wykonywać działania arytmetyczne z dowolną w zasadzie precyzją. W ten sposób doszło do powstania całego szeregu urządzeń cyfrowo-analogowych, w których operacje wymagające dużej dokładności są realizowane w części cyfrowej, a pozostałe - w analogowej. Jeżeli wynik ma być uzyskany w postaci analogowej to możliwe jest wykonanie, jednocześnie z konwersją cyfrowo-analogową, operacji dzielenia w układzie o mniej więcej dwukrotnej liczbie elementów w porównaniu z liczbą elementów konwertera. Układ taki działa jednak dostatecznie dokładnie jedynie w przypadku, gdy dzielnik jest liczbą z przedziału

$[1,2]$ - poza tym przedziałem błędy szybko rosną. Gdy przedział zmienności wartości dzielnika jest szerszy, wówczas alternatywą jest dzielenie cyfrowe. Prowadzi to jednak do rozwiązania kosztownego, o czym świadczy m.in. fakt, że w wielu małych uniwersalnych maszynach cyfrowych rezygnuje się z operacji automatycznego dzielenia właśnie ze względu na wysokie koszty takiego rozwiązania.

2. NORMALIZACJA ZAKRESU ZMIENNOŚCI DZIELNEJ I DZIELNIKA

Zauważmy, że wymieniony we wstępie przedział wartości dzielnika ogranicza do liczby mniejszej od dwu jedynie stosunek największej do najmniejszej przewidywanej wartości dzielnika. Liczby z każdego innego przedziału, którego krańce spełniają powyższy warunek, po pomnożeniu przez odpowiedni stały współczynnik mogą stać się liczbami z przedziału $[1,2]$. Postępowanie takie zmienia jedynie współczynnik proporcjonalności pomiędzy wartością ilorazu, a wyjściową wielkością analogową i nie wpływa na dokładność wykonywanej operacji. Istotną zatem wielkością, która decyduje o dokładności dzielenia jest stosunek największej do najmniejszej przewidywanej wartości dzielnika.

Załóżmy, że dzielna jest zapisana w postaci

$$A = \sum_{i=0}^{k-1} a_i \cdot 2^i , \qquad /1/$$

a zatem przyjmuje wartość z przedziału $[0,\ 2^k - 1]$, zaś dzielnik B

$$B = \sum_{i=0}^{p-1} b_i \cdot 2^i , \qquad /2/$$

a zatem przyjmuje wartości z przedziału $[1,\ 2^p - 1]$ przy czym z przedziału wykluczamy wstępnie liczbę zero. Stosunek największej do najmniejszej przewidywanej wartości dzielnika wynosi więc w tym przypadku $2^p - 1$.

W przypadku, gdy dzielnik przyjmuje wartość z przedziału $[2^r, 2^{r+1} - 1]$ należy go doprowadzić do żądanego zakresu dzieląc przez 2^r. Aby w wyniku tej operacji nie uległ zmianie współczynnik proporcjonalności pomiędzy wartością ilorazu a wyjściową wielkością analogową, dzielna powinna również zostać podzielona przez 2^r. Zauważmy jednak, że w przypadku zapisu dwójkowego, w którym reprezentowane są obydwa argumenty, dzielenie przez dwa do potęgi naturalnej r liczby dwa jest równoważne przesunięciu obu tych zapisów o r pozycji w prawo.

Przy wyznaczaniu zapisów przesuniętych obowiązują następujące zależności:

1. Należy wyznaczyć wektor boolowski $Q = (q_0, q_1, \ldots, q_{p-1})$ przy czym

$$q_i = b_i \cdot \prod_{k=1}^{p-i-1} \bar{b}_{p-k} \qquad (i = 0, 1 \ldots, p-1). \qquad /3/$$

Cechą charakterystyczną tego wektora jest fakt, że tylko jedna z jego składowych może być różna od zera. Ponieważ wykluczyliśmy zerową wartość dzielnika, nie istnieje również wektor Q o wszystkich składowych zerowych. Na podstawie wzoru /3/ łatwo sprawdzić, że niezerowa jest składowa wektora o numerze r.

2. Należy utworzyć zespoły wektorów boolowskich

$$B_i = (b_i, b_{i+1}, \ldots, b_{i+p-2}, b_{i+p-1}) \qquad (i = 0,-1,-2\ldots)^* \qquad /4/$$

oraz

$$A_i = (a_i, a_{i+1}, \ldots, a_{i+p-2}, a_{i+p-1}) \qquad (i = 0,1,2\ldots)^*. \qquad /5/$$

3. Bity zapisów przesuniętych c_i oraz d_i są wtedy dane równaniami:

---

* por. wzory /8/ i /9/. Jeżeli $u < 0$ to b_u z definicji należy uważać za zerowe.

$$d_i = Q \cdot B_i, \qquad /6/$$

$$c_i = Q \cdot A_i. \qquad /7/$$

Po prawych stronach równań /6/ i /7/ znajdują się iloczyny skalarne dwóch wektorów boolowskich. Definicja takiego iloczynu jest analogiczna do definicji zwykłego iloczynu skalarnego wektorów, w którym należy jedynie zastąpić sumowanie sumowaniem logicznym. Wartości, reprezentowane przez te zapisy są następujące:

$$C = \sum_{i=0}^{m-1} c_i \cdot 2^i. \qquad /8/$$

Warto zwrócić uwagę, że wartość liczby C będzie w przybliżeniu równa 2^{-r}-tej części liczby A, jeżeli liczba ta będzie niewiększa od $2^{m+r}-1$, gdyż w przeciwnym przypadku najbardziej znaczące bity zapisu A nie znalazłyby się w zapisie C. Spełnienie tego warunku nakłada ograniczenie na wartość ilorazu. Odpowiednio

$$D = \sum_{i=-v}^{0} d_i \cdot 2^i , \qquad /9/$$

przy czym, jak łatwo sprawdzić, wartość bitu d_0 jest zawsze równa jedności przy niezerowym dzielniku. Pomiędzy wielkościami A i C oraz B i D zachodzą następujące zależności:

$$A = 2^r \cdot C + \sum_{i=0}^{r-1} a_i 2^i = 2^r \cdot C + \delta_1 , \qquad /10/$$

$$B = 2^r D + \sum_{i=0}^{r-v-1} b_i \cdot 2^i = 2^r \cdot D + \delta_2 . \qquad /11/$$

Ostatecznie

$$\frac{C}{D} = \frac{2^{-r}/A - \delta_1/}{2^{-r}/B - \delta_2/} = \frac{A}{B} \cdot \frac{1 - \delta_1/A}{1 - \delta_2/B}, \qquad /12/$$

a zatem, iloraz liczb C i D jest w przybliżeniu równy ilorazowi liczb A i B, lecz dzielnik D jest w tym przypadku zawsze liczbą z przedziału $[1,2]$, co jak wspominaliśmy we wstępie pozwala na przeprowadzenie operacji dzielenia jednocześnie z konwersją.

3. DOKŁADNOŚĆ METODY

Maksymalne wartości błędów bezwzględnych licznika i mianownika wynoszą:

$$\delta_{1\ max} = 2^r, \qquad /13/$$

$$\delta_{2\ max} = 2^{r-v},$$

/jeżeli $r > v$, w przeciwnym przypadku jest on zawsze równy zeru/.

Maksymalne odchylenie wyniku w górę wynosi

$$\Delta_{+max} \approx \frac{A}{B}\left(1 + \frac{\delta_{2\ max}}{B}\right) - \frac{A}{B} = \frac{A}{B^2}\,\delta_{2\ max} < A \cdot 2^{-r-v} \qquad /14/$$

odpowiednio w dół

$$\Delta_{-max} = \frac{A}{B} - \frac{A - \delta_{1\ max}}{B} = \frac{\delta_{1\ max}}{B} < 1. \qquad /15/$$

Maksymalny błąd względny w górę wynosi

$$\varepsilon_g = B \cdot 2^{-r-v} < 2^{1-v}, \qquad /16/$$

a w dół

$$\varepsilon_d = \delta_{1\ max}/A. \qquad /17/$$

Jeżeli dzielna i dzielnik przyjmują wartości bliskie dopuszczalnemu maksimum dla przedziału znormalizowanego /odpowiednio $2^{m+r}-1$ oraz $2^{r+1}-1$/ wówczas maksymalny błąd względny w dół wynosi 2^{1-m}. Opisana metoda jest więc korzystna, gdy wartości dzielnej i dzielnika są ze sobą związane w ten sposób, że wartość ilorazu jest w przybliżeniu stała. Ponieważ błędy, wynikające z odrzucania w niektórych przypadkach części bitów dzielnej i dzielnika kompensują się, należy oczekiwać, że błąd średni będzie w tym przypadku znacznie mniejszy od maksymalnego.

ON A CERTAIN METHOD OF QUOTIENT DETERMINATION IN DIGITAL-ANALOGUE EQUIPMENT

Summary

The paper presents a method of normalizing the range of divident and divisor. This method permits to implement a relatively simple and sufficiently precise system of dividing in digital-analogue equipment. The exactness of the quotient available in the above system is estimated.

W.D.N. Zam. 965/0/68. Nakład 500+25 egz.

Cena zł 60,—

Druk „Znak" Zakł. 3. Zam. 282. 500.