ALGORYTMY

P. 2223/65/66

VOL. III • NO. 5 • 1965

INSTYTUT MASZYN MATEMATYCZNYCH

POLITECHNIKA ŚLĄSKA
BIBLIOTEKA GŁÓWNA
w Gliwicach

A L G O R Y T M Y

Vol. III N°5 1965

P. 2223/65/66

P R A C E

Instytutu Maszyn Matematycznych

Copyright © 1965 - by Instytut Maszyn Matematycznych, Warszawa
Poland
Wszelkie prawa zastrzeżone

K o m i t e t R e d a k c y j n y

Leon ŁUKASZEWICZ /redaktor/, Antoni MAZURKIEWICZ,
Tomasz PIETRZYKOWSKI /z-ca redaktora/, Dorota PRAWDZIC,
Zdzisław WRZESZCZ.
Redaktor działowy: Krzysztof MOSZYŃSKI.
Sekretarz redakcji: Romana NITKOWSKA.

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

P. 246/66 -vol 3/n.5 str. 51

TREŚĆ
CONTENTS

NUMERICAL ANALYSIS

Instytut Maszyn Matematycznych
Algorytmy N° 5
© 1965.12
517.91:518.61

THE NEWTON'S METHOD FOR FINDING AN APPROXIMATE SOLUTION TO AN EIGENVALUE PROBLEM OF ORDINARY LINEAR DIFFERENTIAL EQUATIONS

by Krzysztof MOSZYŃSKI,
Jerzy WRZOS

Received January 10th, 1965

The paper contains the definition of a kind of Newton's Method, applied for the solution of the eigenvalue problem for the system /1/ of linear ordinary differential equations with boundary conditions /2/. The second order convergence of defined process is proved in the case of a single eigenvalue. Some results obtained using this method are given in Appendix.

1. INTRODUCTION

Let us consider the eigenvalue problem of the form

$$\dot{u}(t) = \left[A(t) + \lambda B(t) \right] u(t) \qquad /1/ \ *)$$

$$M\, u(a) + N\, u(b) = 0 \qquad /2/$$

where

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \quad \text{- the column vector}$$

*) The point is used as a symbol of differentiation with respect to t.

$A(t)$, $B(t)$ - $m \times m$ matrices continuous *) on the interval $[a, b]$; $B(t) \not\equiv 0$

M, N - constant $m \times m$ matrices.

$-\infty < a, b < +\infty$.

Assume the boundary conditions /2/ to be independent, i.e. the rectangular matrix

$$[M \mid N]$$

to be of the rank m.

Let $P(t,\lambda)$ be the $m \times m$ matrix build up with m independent solutions of the system /1/, taken as columns.

Let's now form the matrix

$$S(\lambda) = M\,P(a,\lambda) + N\,P(b,\lambda) \qquad /3/$$

and the function

$$F(\lambda) = \det[S(\lambda)]$$

The matrix $S(\lambda)$ plays a very important role in the theory of the systems of the form /1/, /2/. The reason is that the eigenvalues of /1/, /2/ are the only roots of the equation [1], [2],

$$F(\lambda) = 0. \qquad /4/$$

It is clear that

$$\dot{P}(t,\lambda) = [A(t) + \lambda B(t)]\,P(t,\lambda) . \qquad /5/$$

On the other hand, it is a known fact, that the matrix $P(t,\lambda)$ can be represented by a uniformly convergent power series [4]

*) All such notions as continuity, differentiability and so on, of the matrices are to be understood as continuity resp. differentiability of their elements.

$$P(t,\lambda) = P(t,\lambda^*) + (\lambda - \lambda^*)P_1(t,\lambda^*) + (\lambda - \lambda^*)^2 P_2(t,\lambda^*) + \ldots \qquad /6/$$

where

$$P_k(t,\lambda^*) = \frac{1}{k!}\frac{\partial^k}{\partial\lambda^k} P(t,\lambda^*).$$

Let's denote

$$S_k(\lambda) = MP_k(a,\lambda) + NP_k(b,\lambda) \qquad /7/$$

then

$$S(\lambda) = S(\lambda^*) + (\lambda - \lambda^*)S_1(\lambda^*) + (\lambda - \lambda^*)^2 S_2(\lambda) + \ldots, \qquad /8/$$

the series /8/ being also uniformly convergent for any complex λ, λ^*.

It follows that the eigenvalues of /1/, /2/, being the roots of equation /4/, have no finite limit points.

Let's now substitute the uniformly convergent series /6/ to the equation /5/. This gives differential equations for the matrices $P_k(t,\lambda)$.

$$\dot{P}_k(t,\lambda) = \left[A(t) + \lambda B(t)\right] P_k(t,\lambda) + B(t) P_{k-1}(t,\lambda) \qquad /9/$$

for $k = 0, 1, 2, \ldots$

where

$$P_{-1}(t,\lambda) \equiv 0$$

$$P_0(t,\lambda) \equiv P(t,\lambda).$$

Equations /9/ with the initial conditions $P_k(a,\lambda)$ such that

$$\det\left[P_0(a,\lambda)\right] \neq 0,$$

and

$$k!\,\frac{\partial^k}{\partial\lambda^k}P_0(a,\lambda) = P_k(a,\lambda),$$

and the definition /7/ can be used for numerical determination of the matrices $S_k(\lambda)$ for $k = 0, 1, 2, \ldots$.

For example, any step by step prooess for integration of the initial value problems can be applied for system /9/.

In particular, it is possible to take as initial conditions for /9/

$$\begin{aligned} &P_0(a,\lambda) = C \quad \text{- constant matrix,} \quad \det(c) \neq 0 \\ &P_k(a,\lambda) \equiv 0 \quad \text{for any } k = 1, 2, 3, \ldots \end{aligned} \qquad /10/$$

Definition 1

The number r is oalled the multiplicity of the eigenvalue λ^* of the system /1/, /2/, if

$$\frac{d^{(r)}}{d\lambda^r}F(\lambda^*) \neq 0 \quad \text{and} \quad \frac{d^{(k)}}{d\lambda^k}F(\lambda^*) = 0,$$

for any k, $0 \leqslant k < r$

Definition 2

The eigenvalue λ^* of /1/, /2/ is oalled regular, if the matrix $S(\lambda^*)$ is of the rank $m-r$, where r is the multiplicity of the eigenvalue λ^*.

Note.

The problems of the form /1/, /2/ of a very important class, are the so called selfadjoint problems. The definition of selfadjointness, as well as the theory of the selfadjoint systems of the form /1/, /2/ are due to G.A. Bliss, and were published in [1], [2].

Definition 3

The system /1/, /2/ is called definitely selfadjoint if there exists a matrix $T(t)$, differentiable and invertible for any $t \in [a, b]$, such that:

1. $\dot{T}(t) + T(t)\,A(t) + A^T(t)\,T(t) = 0$

   $T(t)\,B(t) + B^T(t)\,T(t) = 0$

   $MT^{-1}(a)\,M^T - NT^{-1}(b)\,N^T = 0$

2. the matrix $T^T(t)\,B(t)$ is for every $t \in [a, b]$
   - symmetric
   - positively semidefined;

3. the function $u(t) \equiv 0$ is the only solution of the boundary value problem

$$\dot{u}(t) = A(t)u(t)$$
$$M\,u(a) + N\,u(b) = 0$$

   satisfying the additional condition

$$u^T(t)T^T(t)B(t)u(t) \equiv 0 .$$

It follows from the theory of G.A. Bliss that all eigenvalues of definitely selfadjoint problem are

- real and different from zero
- regular.

Example

The classic Sturm-Liouville problem

$$\begin{aligned} &\ddot{y}(t) + [\lambda - q(t)]\,y(t) = 0 \\ &\cos \alpha\, y(a) + \sin \alpha\, \dot{y}(a) = 0 \\ &\cos \beta\, y(b) + \sin \beta\, \dot{y}(b) = 0 \end{aligned} \qquad /11/$$

where $y(t)$, $q(t)$ are functions, α, β - given real numbers, is definitely selfadjoint. This problem can be presented in the form /1/, /2/ with

$$u(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}, \quad A(t) = \begin{bmatrix} 0, & 1 \\ q(t), & 0 \end{bmatrix}, \quad B(t) = \begin{bmatrix} 0, & 0 \\ -1, & 0 \end{bmatrix},$$

$$M = \begin{bmatrix} \cos\alpha, & \sin\alpha \\ 0, & 0 \end{bmatrix}, \quad N = \begin{bmatrix} 0, & 0 \\ \cos\beta, & \sin\beta \end{bmatrix}.$$

In this case $T(t) = \begin{bmatrix} 0, & 1 \\ -1, & 0 \end{bmatrix}$ and $T^T(t)\, B(t) = \begin{bmatrix} 1, & 0 \\ 0, & 0 \end{bmatrix}$.

It is also known that all eigenvalues of the problem /11/ are of multiplicity one /are single/.

Definition 4 [1], [2]

Let the matrix $S(\lambda^*)$ be of the rank $m - \varsigma$. If $\varsigma > 0$, the number ς is called the index of the eigenvalue λ^*.

Theorem [1], [2]

Let r be the multiplicity of the eigenvalue λ^* of /1/, /2/, and ς it's index. Then

$$r \geqslant \varsigma$$

Proof

Let the eigenvalue λ^* of /1/, /2/ be of the index ς, $0 < \varsigma \leqslant m$. It means that there exists a linear independent system of vectors

$$v_1, v_2, \ldots, v_\varsigma$$

such that

$$S(\lambda^*)v_i = 0 \qquad i = 1, 2, \ldots, \rho .$$

It is possible to find the vectors $v_{\rho+1}, v_{\rho+2}, \ldots, v_m$ so as to make the system $v_1, v_2, \ldots, v_m$ linear independent.

Let's form the matrix

$$V = \left[v_1, v_2, \ldots, v_m\right];$$

the vectors v_i being its columns.

Since $S(\lambda) = S(\lambda^*) + (\lambda - \lambda^*)S_1(\lambda^*) + \ldots$ and

$$S(\lambda)V = \left[S(\lambda)v_1, S(\lambda)v_2, \ldots, S(\lambda)v_\rho, S(\lambda)v_{\rho+1}, \ldots, S(\lambda)v_m\right],$$

we have

$$\det\left[S(\lambda)V\right] = (\lambda - \lambda^*)^\rho \det([K_1, K_2, \ldots, K_\rho, S(\lambda)v_{\rho+1}, \ldots, S(\lambda)v_m])$$

where

$$K_i = \left[S_i(\lambda) + (\lambda - \lambda^*)S_2(\lambda) + \ldots\right] v_i \qquad i = 1, 2, \ldots ;$$

K_i being vectors bounded in the neighbourhood of λ^*. But $\det(V) = d \neq 0$, hence

$$\det(S(\lambda)) = \frac{1}{d}(\lambda - \lambda^*)^\rho \det\left[S(\lambda)V\right] =$$

$$= \frac{1}{d}(\lambda - \lambda^*)^\rho \det\left[K_1, K_2, \ldots, K_\rho, S(\lambda)v_{\rho+1}\right].$$

Note

If the eigenvalue λ^* of /1/, /2/ is single, then its index is one; each single eigenvalue of /1/, /2/ is regular.

The purpose of this paper is to define and to prove the convergence of an iterative process, being a kind of Newton's Method, solving the problem /1/, /2/.

This process is defined by formulae /26/ and /29/ and further by formula /30/.

The second order convergence, in the case of a single eigenvalue, follows from Theorem 2.

In the case of the system of m equations the process /29/ or /30/ requires to determine the eigenvector corresponding to the eigenvalue of the minimal modulus of some symmetric positive definite matrix. For this purpose the simple iteration method can be used. As in the case of convergence, the obtained values do not change very roughly, only a few iterations should be needed on each step.

In the case $m = 2$ the process becomes much more simple /see Part 5/.

2. FUNCTION $\mu(\lambda)$ AND ITS PROPERTIES

Let's first prove a few simple lemmas.

Lemma 1

Let P be the $m \times m$ matrix. Then the matrices P^TP and PP^T have the same eigenvalues. If z is an eigenvector, corresponding to the non zero eigenvalue μ of P^TP, then Pz is an eigenvector of PP^T corresponding to the same eigenvalue μ.

Proof

It is known that the matrices P^TP and PP^T are of the same rank as P /see [3]/. It follows that if zero is an eigenvalue of P^TP of multiplicity $k \leq m$, it is the eigenvalue of the same multiplicity of PP^T. Let now $\mu_{k+1}, \mu_{k+2}, \ldots, \mu_m$ be the non zero eigenvalues, and $z_1, z_2, \ldots, z_m$ the orthogonal set of all eigenvectors of P^TP. Then

$$P^TP\, z_j = \mu_j\, z_j \qquad j = k+1,\ k+2,\ \ldots,\ m$$

and

$$PP^TP\, z_j = \mu_j\, Pz_j$$

Let us observe, that $Pz_j \neq 0$ for $j = k+1, k+2, \ldots, m$, henoe Pz_j, is the eigenveotor of PP^T, corresponding to the non zero eigenvalue μ_j. Moreover, $(Pz_j)^TPz_i = z_j^T P^T z_i = \mu_i\, z_j^T z_i = 0$, if $i \neq j$. It follows, that $\mu_j, j = k+1, k+2, \ldots, m$, are the eigenvalues of PP^T and of P^TP of the same multiplicity.

Lemma 2

Let P be the $m \times m$ matrix, and $P^TP = R$ be of the rank $m-1$. Let $Rx = 0$, where x is a so normalized veotor that $x^Tx = 1$. Then $Px = 0$.

Proof

Since $\det(P) = 0$, there exists such a vector y, $y^Ty = 1$ that $Py = 0$; henoe $P^TPy = 0$. Because of the rank of R, only $\pm x$ satisfy the relations $Rx = 0$, $x^Tx = 1$; henoe $y = \pm x$.

Lemma 3

Let C be a symmetrio $m \times m$ matrix of the rank $m-1$. Let x be suoh a vector that

$$Cx = 0 \qquad \text{and} \qquad x^Tx = 1$$

Let us form the symmetrio $(m+1) \times (m+1)$ matrix

$$W = \left[\begin{array}{c|c} x & C \\ \hline 0 & x^T \end{array}\right] \qquad /12/$$

Then

$$W^{-1} = \left[\begin{array}{c|c} x^T & 0 \\ \hline Q & x \end{array}\right] \qquad /13/$$

where

$$Q = \left[C + xx^T\right]^{-1} - xx^T$$

Proof - by simple multiplication.

Let now λ^* be a single eigenvalue of /1/, /2/. It follows that the matrix $S(\lambda^*)$ is of the rank m-1. Let's now denote

$$R(\lambda) = S^T(\lambda)\, S(\lambda).$$

The matrix $R(\lambda)$ is symmetric and positively semidefined; hence, for the roots μ of the polynominal equation

$$\det\left[R(\lambda) + \mu E\right] = 0 \qquad /14/$$

there is always $\mu \leq 0$.

As the rank of $S(\lambda^*)$ is m-1 the rank of $R(\lambda^*)$ is m-1 too. It follows then, that $\mu = 0$ is a single root of /14/ for $\lambda = \lambda^*$. We can consider the following relations

$$\begin{aligned} \left[R(\lambda) + \mu E\right] x &= 0 \\ x^T x &= 1 \end{aligned} \qquad /15/$$

where

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix},$$

as a system of $m+1$ equations with $m+2$ unknown $\lambda, \mu, x_1, x_2, \ldots, x_m$.

Let $x^{*T} = [x_1^*, x_2^*, \ldots, x_m^*]$ be a normalized eigenvector, corresponding to the single eigenvalue $\mu = 0$ of $R(\lambda^*)$.

Then, the following theorem holds:

Theorem 1

There exists a number $\delta > 0$, such that for $|\lambda - \lambda^*| < \delta$, the system /15/ defines exactly one system of $m+1$ functions $\mu(\lambda)\, x_1(\lambda), \ldots, x_m(\lambda)$,

$$\left.\begin{array}{l} [R(\lambda) + \mu(\lambda) E]\, x(\lambda) = 0 \\ x^T(\lambda)\, x(\lambda) = 1 \end{array}\right\} \qquad \text{for } |\lambda - \lambda^*| < \delta,$$

where

$$x^T(\lambda) = [x_1(\lambda), x_2(\lambda), \ldots, x_m(\lambda)]$$

and

$$\mu(\lambda^*) = 0$$
$$x(\lambda^*) = x^*.$$

Proof

It is quite easy to see that the Jacobi's matrix for the system /15/ is of the form

$$\left[\begin{array}{c|c} x & R(\lambda) + \mu E \\ \hline 0 & 2\, x\, T \end{array}\right]$$

As it follows from the Lemma 3, this matrix is invertible for $\lambda = \lambda^*$, and the classic Theorem on Implicit Functions can by applied, for the system /15/.

Note.

As the set of all eigenvalues of /1/, /2/ has no finite limit points, there exists a number δ', $0 < \delta' \leq \delta$, such that

$$\mu(\lambda) < 0, \qquad \text{for} \qquad 0 < |\lambda - \lambda^*| < \delta'.$$

Moreover, for $|\lambda - \lambda^*| < \delta'$, $-\mu(\lambda)$ is single eigenvalue of $R(\lambda)$.

Theorem 2

The value $\lambda = \lambda^*$ is a root of $\mu(\lambda) = 0$, of multiplicity 2. Exactly

$$\mu(\lambda^*) = 0$$

$$\frac{d\,\mu(\lambda^*)}{d\lambda} = 0$$

$$\frac{d^2\,\mu(\lambda^*)}{d\lambda^2} < 0$$

Proof

From the equations /15/ we get

$$\left[\frac{d\,R}{d\,\lambda} + \frac{d\,\mu}{d\,\lambda}\,E\right] x + \left[R + \mu\,E\right]\frac{dx}{d\lambda} = 0$$

$$\left(\frac{dx}{d\lambda}\right)^T x = 0 \qquad /16/$$

The formulae /16/ can be presented in the following matrix form:

$$\left[\begin{array}{c|c} x & R + \mu\,E \\ \hline 0 & x^T \end{array}\right] \left[\begin{array}{c} \frac{d\mu}{d\lambda} \\ \hline \frac{dx}{d\lambda} \end{array}\right] = - \left[\begin{array}{c} \frac{dR}{d\lambda}\,x \\ \hline 0 \end{array}\right]$$

In the neighbourhood of the point $(\lambda^*, 0, x_1^*, x_2^*, \ldots, x_m^*)$ the matrix

$$\left[\begin{array}{c|c} x & R + \mu E \\ \hline 0 & x^T \end{array}\right]$$

is invertible, hence applying Lemma 3 we can write

$$\frac{d\mu}{d\lambda}(\lambda) = - x^T \frac{dR}{d\lambda} x$$

$$\frac{dx}{d\lambda}(\lambda) = - \left\{ \frac{d\mu}{d\lambda} x + \left[(R + \mu E) + xx^T \right]^{-1} \frac{dR}{d\lambda} x \right\} \qquad /17/$$

Let's denote:

$$z(\lambda) = S(\lambda)^{-} x(\lambda)$$
$$w(\lambda) = S_1(\lambda)\, x(\lambda) \; . \qquad /18/$$

There is now

$$\frac{dR}{d\lambda}(\lambda)\, x(\lambda) = S_1^T(\lambda)\, z(\lambda) + S^T(\lambda) w(\lambda)$$

and

$$\frac{d\mu(\lambda)}{d\lambda} = -2\, z^T(\lambda)\, w(\lambda) \qquad /19/$$

According to Lemma 2, there is $z(\lambda^*) = 0$, and hence

$$\frac{d\mu}{d\lambda}(\lambda^*) = 0 \qquad /20/$$

From the first equation of /16/ we get

$$\left[\frac{d^2R}{d\lambda^2} + \frac{d^2\mu}{d\lambda^2} E \right] x + 2 \left[\frac{dR}{d\lambda} + \frac{d\mu}{d\lambda} E \right] \frac{dx}{d\lambda} + \left[R + \mu E \right] \frac{d^2x}{d\lambda^2} = 0 \qquad /21/$$

Multiplying the equation /21/ by x^T, and taking into account the relations

$$x^T \left[R + \mu E \right] = 0$$

$$x^T \frac{dx}{d\lambda} = 0$$

as well as the formulae /17/, we get

$$\frac{d^2\mu}{d\lambda^2}(\lambda) = -x^T \frac{d^2R}{d\lambda^2} x - 2\left(\frac{d\mu}{d\lambda}\right)^2 + 2\, x^T \frac{dR}{d\lambda} \left[R + \mu E + xx^T \right]^{-1} \frac{dR}{d\lambda} x. \qquad /22/$$

Let's observe that

$$\frac{dR}{d\lambda} = S^T S_1 + S_1^T S$$

$$\frac{d^2R}{d\lambda^2} = 2\left[S^T S_2 + S_1^T S_1 + S_2^T S \right];$$

since $z(\lambda^*) = 0$, and /18/, /20/, /22/, we get for $\lambda = \lambda^*$

$$\frac{d^2\mu}{d\lambda^2} = 2\, w^T \left\{ S \left[R + xx^T \right]^{-1} S^T - E \right\} w \qquad /23/$$

There exists an orthogonal matrix T such that /for $\lambda = \lambda^*$/

$$\left[R + xx^T \right]^{-1} = T\, D^{-1}\, T^T$$

where

$$D = \begin{bmatrix} 1 & & & 0 \\ & \mu_2 & & \\ & & \ddots & \\ 0 & & & \mu_m \end{bmatrix}; \quad \mu_j > 0 \quad \text{for} \quad j = 2, 3, \ldots, m \qquad /24/$$

μ_j are the non zero eigenvalues of $R(\lambda^*)$.

Hence, $S\left[R + xx^T\right]^{-1} S^T = STD^{-1} T^T S^T = STD^{-\frac{1}{2}}\left(STD^{-\frac{1}{2}}\right)^T$.

Acoording to Lemma 1, the diagonal for of the matrix $S\left[R + xx^T\right]^{-1} S^T$ is the same as this of the matrix

$$\left(S\,T\,D^{-\frac{1}{2}}\right)^T S\,T\,D^{-\frac{1}{2}} = D^{-\frac{1}{2}} T^T S^T S\,T\,D^{-\frac{1}{2}} .$$

But since

$$T^T S^T S\,T = T^T R\,T = \begin{bmatrix} 0, & & 0 \\ & \mu_2 & \\ & & \ddots \\ 0 & & \mu_m \end{bmatrix},$$

then

$$\left(S\,T\,D^{-\frac{1}{2}}\right)^T S\,T\,D^{-\frac{1}{2}} = \begin{bmatrix} 0 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix}$$

It means that there exists an unitary matrix U, such that /always for $\lambda = \lambda^*$/

$$S\left[R + xx^T\right]^{-1} S^T = U \begin{bmatrix} 0 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix} U^T . \qquad /25/$$

From /23/ we get $\frac{d^2\mu}{d\lambda^2}(\lambda^*) = -2\, w^T(\lambda^*) U \begin{bmatrix} 1 & & 0 \\ & 0 & \\ 0 & & 0 \end{bmatrix} U^T w(\lambda^*)$ /26/

Let's denote $U = \left[u_1, u_2, \ldots, u_m\right]$ where u_j is a column of the orthogonal matrix U from the formula /25/.

There is $S\left[R + xx^T\right]^{-1} S^T u_1 = 0$ and we deduce that $S^T(\lambda^*)u_1 = 0$ because, acoording to /25/, zero is a single eigenvalue of the matrix $S\left[R + xx^T\right]S^T$ for $\lambda = \lambda^*$.

Now we can present the formula /26/ in the form

$$\frac{d^2\mu}{d\lambda^2}(\lambda^*) = -2\left[u_1^T S_1(\lambda^*)\, x(\lambda^*)\right]^2 \leqslant 0$$

It remains to show that

$$u_1^T S_1(\lambda^*)\, x(\lambda^*) \neq 0.$$

Let's observe that

$$S(\lambda) = S(\lambda^*) + (\lambda - \lambda^*) S_1(\lambda^*) + (\lambda - \lambda^*)^2 K$$

where K stands for a matrix of elements bounded in the neighbourhood of λ^*. If we denote $h = \lambda - \lambda^*$, we get

$$S(\lambda)\, x(\lambda^*) = h\, S_1(\lambda^*)\, x(\lambda^*) + h^2 K\, x(\lambda^*)$$

and

$$u_1^T S(\lambda) = h\, u_1^T S_1(\lambda^*) + h^2\, u_1^T K$$

$$u_1^T S(\lambda)\, x(\lambda^*) = h\, u_1^T S_1(\lambda^*)\, x(\lambda^*) + h^2\, u_1^T K\, x(\lambda^*)$$

On the other hand,

$$F(\lambda) = \det\left[S(\lambda)\right] = \pm \det\left[U^T S(\lambda) T\right]$$

where T and U are the orthogonal matrices, defined by /24/ and /25/. Since $T = \left[x(\lambda^*), \ldots\right]$, hence

$$U^T S(\lambda)\, T = \left[\begin{array}{c|c} h\, u_1^T S_1(\lambda^*)\, x(\lambda^*) + o(h^2) & o(h),\ o(h),\ \ldots \\ \hline \begin{array}{c} o(h) \\ o(h) \\ \vdots \end{array} & o(1) \end{array}\right]$$

If it were $u_1^T S_1(\lambda^*)x(\lambda^*) = 0$, then

$$F(\lambda) = O(h^2) \qquad \text{when} \quad \lambda \to \lambda^*.$$

But since λ^* is the single eigenvalue of the equation /4/, this is impossible; hence

$$u_1^T S(\lambda^*)x(\lambda^*) \neq 0. \qquad /27/$$

3. THE ALGORITHM

Let's observe that the relations

λ^* is an eigenvalue of /1/, /2/,

and

$\mu = 0$ is the root of the equation $\det\left[R(\lambda^*) + \mu E\right] = 0$,

are equivalent.

This note, as well as Theorems 1 and 2 allow to replace the problem of calculation of a single eigenvalue λ^* of /1/, /2/, by calculation of a zero of the function $\mu(\lambda)$, defined by the Theorem 1.

Clearly, because of the local character of the Theorem 1, this method of computing λ^* is of the 'local character' too.

Since λ^* is the root of the multiplicity 2 of the equation

$$\mu(\lambda) = 0,$$

we can apply the following variant of the Newton's method

$$\lambda_{k+1} = \lambda_k - 2\frac{\mu(\lambda_k)}{\frac{d\mu}{d\lambda}(\lambda_k)}. \qquad /28/$$

It is easy to see this process to be of the order 2, i.e.

$$\lambda_{k+1} - \lambda_k = O\left(\lambda_k - \lambda^*\right)^2,$$

provided λ_0 was chosen close enough to λ^*. Clearly, the function $\mu(\lambda)$ must be defined for $\lambda = \lambda_0$ /this is a 'local character' of this process/.

Let's denote

$$z_k = S(\lambda_k)\, x\,(\lambda_k) = z(\lambda_k)$$

$$w_k = S_1(\lambda_k)\, x(\lambda_k) = w(\lambda_k).$$

The first formula of /15/ gives

$$\mu(\lambda_k) = - z_k^T z_k$$

Using formula /19/ we can write /28/ in the form

$$\lambda_{k+1} = \lambda_k - \frac{z_k^T z_k}{z_k^T w_k} \qquad /29/$$

However, the formula /29/ may be still not very satisfactory from the numerical point of view, because for $\lambda = \lambda^*$ the denominator vanishes in the second member of /29/.

It is possible to omit this difficulty. It follows from Lemma 1 and the note to Theorem 1 that for $0 < |\lambda - \lambda^*| < \delta'$, $\delta' > 0$

$$z(\lambda) = \alpha\, v\,(\lambda)$$

where α is a constant, and $v(\lambda)$ is the eigenvector of the matrix $S(\lambda)\, S^T(\lambda)$, such that

$$\left[S(\lambda)S^T(\lambda) + \mu(\lambda)E\right] v(\lambda) = 0$$

$$v^T(\lambda)\, v(\lambda) = 1$$

Now the formula /28/ takes the form

$$\lambda_{k+1} = \lambda_k - \frac{v_k^T z_k}{v_k^T w_k} \qquad /30/$$

where

$$v_k = v(\lambda_k).$$

Applying Lemma 2 to the matrix $S(\lambda^*)S^T(\lambda^*)$ we get $S^T(\lambda^*)\, v(\lambda^*) = 0$. Since λ^* is a single eigenvalue of /1/, /2/, we get

$$v = \pm u_1$$

where u_1 is the first column of the matrix U, defined by /25/.

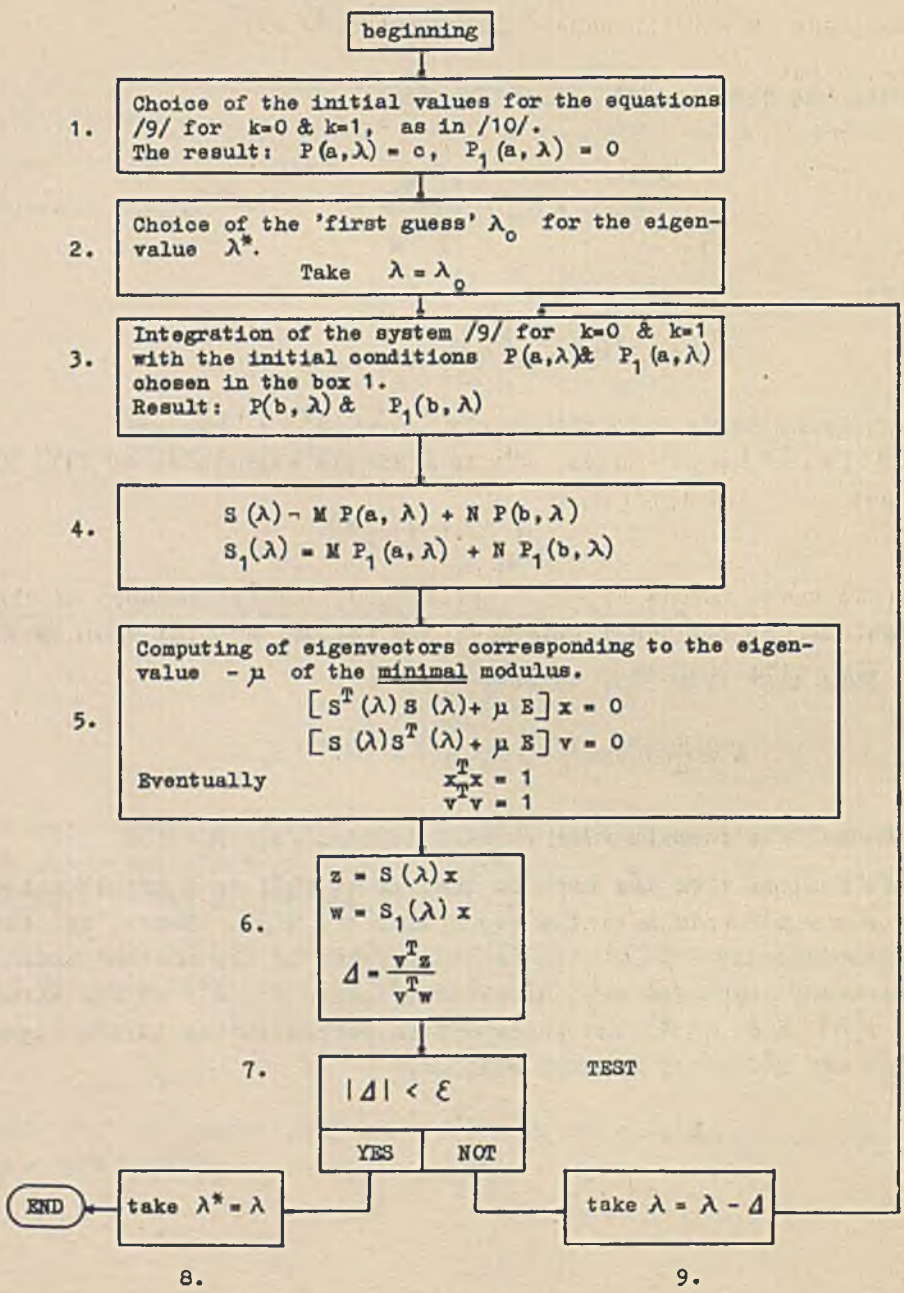
Note that from /27/ follows

$$v^T(\lambda^*) w(\lambda^*) = \pm u_1^T S_1(\lambda^*)\, x(\lambda^*) \neq 0.$$

It means, the formula /30/ is well defined for $\lambda = \lambda^*$.

It follows from the note to Theorem 1, that in a neighbourhood of λ^*, $-\mu(\lambda)$ is a single eigenvalue of $R(\lambda)$. Hence, in this neighbourhood, $-\mu(\lambda)$ is the eigenvalue of the minimal modulus. This means that for λ sufficiently close to λ^* we can choose as $x(\lambda)$ and $v(\lambda)$ the eigenvectors, corresponding to the eigenvalue of $R(\lambda)$, of minimal modulus.

4. THE FLOWDIAGRAM OF THE ALGORITHM



5. THE CASE m = 2

The case $m = 2$ is especially simple.

Let's put

$$S(\lambda) = \begin{bmatrix} a, & b \\ c, & d \end{bmatrix}$$

then

$$S^T(\lambda)S(\lambda) = \begin{bmatrix} a^2 + c^2, & ab + cd \\ ab + cd, & b^2 + d^2 \end{bmatrix}$$

and

$$S(\lambda)S^T(\lambda) = \begin{bmatrix} a^2 + b^2, & ac + bd \\ ac + bd, & c^2 + d^2 \end{bmatrix}$$

To avoid great values apearing while computing the squares of the elements /for ex. when we use the fixpoint arithmetic/ we introduce the dashed values

$$a' = \frac{a}{M}, \quad b' = \frac{b}{M}, \quad c' = \frac{c}{M}, \quad d' = \frac{d}{M}$$

where M is a normalizing factor.

As it is easy to see for the eigenvalue $-\mu_1$ of $S^T(\lambda)S(\lambda)$ of the minimal modulus we have

$$\mu_1' = \frac{\mu_1}{M^2} = \frac{1}{2}\left[\sqrt{B'^2 - 4C'} - B'\right] \qquad /31/$$

where

$$B' = a'^2 + b'^2 + c'^2 + d'^2$$

$$C' = (b'c' - a'd')^2$$

Hence, the eigenvectors x'^T and v'^T corresponding to μ_1 are of the form

$$x'^T: \left[a'b' + c'd', -(a'^2 + b'^2 + \mu_1'^2)\right], \text{or } \left[b'^2 + d'^2 + \mu_1'^2, -(a'b' + c'd')\right]$$
$$v'^T: \left[a'c' + b'd', -(a'^2 + b'^2 + \mu_1'^2)\right], \text{or } \left[c'^2 + d'^2 + \mu_1'^2, -(a'c' + b'd')\right] \quad /32/$$

The preferable choice is the one of the vectors of greater sum of moduli of coordinates. Because of the homogenity of the formula /30/ we can use vectors x' and v' in the place of unitary vectors x and v.

6. RESULTS OBTAINED

In the Computing Centre of IMM in Warsaw, the program for finding eigenvalues of /1/, /2/ for $m = 2$ have been completed.

This program deals with the formula /30/, where the vectors x_k and v_k are determined using /32/.

For integration of the system /9/ for $k = 0$ and $k = 1$, the Gill's Method was used [5]. For the integration purpose, the interval $[a, b]$ has been divided into n equal parts. Let's denote.

$$h = \frac{b - a}{n};$$

hence, the truncation error ε of the Gill's process is $\sim h^5$. This limits the accuracy of the program.

The program contains an independent subroutine for computing the matrices $A(t)$ and $B(t)$.

The data for the program should contain

- matrices M, N – /boundary conditions/,
- matrices $P(a\lambda)$, $P_1(a_1\lambda)$ / initial conditions for the system /9//,

- a, b, n - /the bounds of the interval and the number determining its decomposition/,
- λ_0 - /starting point for the iteration - the 'first guess' for the eigenvalue/.

Two simple examples of the problem /1/, /2/ were considered.

1. The equation $\ddot{u} + \lambda u = 0$ with boundary conditions $u(0) = u(\pi) = 0$. This problem determines the eigenvalues $\lambda = 1, 4, 9, 16, \ldots, n^2, \ldots$.
2. The so called 'Mathieu equation' $\ddot{u} + (\lambda - \beta \cos^2 t)u = 0$ with boundary conditions $u(0) = u(\pi) = 0$.

Following tables give some results obtained on ZAM-2 computer [6].

Example 1.

Table 1.

K Number of step	λ_k			
0 Starting point	0.81	3.89	8.81	1.75
1	0.974575	3.998238	9.003300	0.624684
2	0.999526	4.000626	9.006832	0.921598
3	1.000009	4.000627 *)	9.006833 *)	0.995466
4	1.000010 *)			0.999994 *)
Exact limit	1.0	4.0	9.0	1.0
n	20	20	20	20
h	≈0.157	≈0.157	≈0.157	≈0.157

*) Any more changes when n given as in the table.

Table 2.

K Number of step	λ_k	
0 Starting point	16.21	16.21
1	16.034	16.00070
2	16.036	16.0025
3	16.036 *)	16.0025 *)
Exact limit	16.0	16.0
n	20	40
h	≈0.157	≈0.075

Table 3.

K Number of step	λ_k		
0 Starting point	100.21	100.21	100.21
1	102.59	101.306	100.225
2	102.77 *)	101.321 *)	100.225 *)
Exact limit	100.0	100.0	100.0
n	20	30	50
h	≈0.157	≈0.104	≈0.063

*) Any more changes when n given as in the table

Example 2

Table 4.

$\beta = 0.5$

K Number of step	λ_k		
0 Starting point	1.06	4.18	9.18
1	1.119579	4.248333	9.257247
2	1.123077	4.249378	9.257784
3	1.123088 *)	4.249328 *)	9.257785 *)
Exact limit **)	1.123077	4.248698	9.250946
n	20	20	20
h	≈0.157	≈0.157	≈0.157

Table 5.

K Number of step	λ_k			
0 Starting point	100.21	100.21	100.21	100.21
1	102.59	101.306	100.225	100.113
2	102.77	101.321	100.225	100.113
Exact limit	100.0	100.0	100.0	100.0
n	20	30	50	60
h	≈0.157	≈0.104	≈0.063	≈0.052

*) Any more changes when n given as in the table.

**) See [7].

$\beta = 0.5$ Table 6.

K Number of step	λ_k		
0 Starting point	1.29	4.62	9.67
1	1.352625	4.735501	9.764013
2	1.358220	4.738938	9.764861
3	1.358248	4.738941	9.764861 *)
4	1.358248 *)	4.738941 *)	
Exact limit **)	1.358230	4.738288	9.757979
n	20	20	20
h	≈0.157	≈0.157	≈0.157

Relatively poor results for large values of λ are caused by errors of integration of the system /9/. It is possible, however, to get better results in those cases when increasing the number n. This will increase the computing time, but neither a greater memory space nor changes in the program are needed. The improvement of results by increasing the number n is illustrated in tables 2 and 3.

*) Any more changes.

**) See [7].

References

1. BLISS G.A.: A Boundary Value Problem for a System of Ordinary Linear Differential Equations of the First Order, Transactions Amer. Math. Soc., 1926:28, 561-584.

2. BLISS G.A.: Definitely Selfadjoint Boundary Value Problems, Transactions Amer. Math. Soc., 1938:44, 413-428.

3. MOSTOWSKI A., STARK M.: Algebra Wyższa, Warszawa 1953:1, 153.

4. CODDINGTON, LEVINSON: Theory of Ordinary Differential Equations, Mc Graw-Hill 1955, 37.

5. WILKES M.V., WHEEDER D.J., GILL S.: The Preparation of Programmes for an Electronic Digital Computer, Cambridge 1957.

6. Maszyna ZAM 2. Opis maszyny. Kompendium Programowania w Języku SAS. Oprac. K. Piałkowski, J. Swianiewicz, Prace ZAM PAN, Warszawa 1962:C3.

7. NBS. Tables Relating to Mathieu Functions. Columbia University Press, New York 1951.

S T A T I S T I C A L A P P L I C A T I O N

Instytut Maszyn Matematycznych
Algorytmy N° 5
© 1965.12
519.21:681.14-523.8

TECHNIKA STOSOWANIA METOD MONTE-CARLO NA ZAM-2

Elżbieta PLESZCZYŃSKA

Pracę złożono 16.10.1964 r.

W pracy opisano "potencjał losowy" opracowany dla maszyny ZAM-2, potrzebny przy przeprowadzaniu obliczeń metodami Monte-Carlo.

1. WSTĘP

Niniejszy artykuł przeznaczony jest dla tych użytkowników maszyn ZAM-2, którzy chcieliby przeprowadzić na niej obliczenia stosując metody Monte-Carlo, znają zaś jedynie programowanie w języku SAKO. Przy opracowywaniu obecnej wersji języka SAKO nie przygotowano gotowych schematów do obliczeń tego rodzaju, toteż potrzebne są dodatkowe objaśnienia.

Ogólne wiadomości o "warsztacie" metod Monte-Carlo podano w skrócie w artykule [4]. Będziemy korzystać z wprowadzonej tam terminologii. Obecny artykuł zawiera:

1. uwagi dotyczące symboliki i sposobu korzystania z generatorów liczb pseudolosowych, szczegółów technicznych ich budowy, oraz metod opisywania,
2. dokładne opisy z dokumentacją w SAS poszczególnych generatorów opracowanych dla ZAM-2,
3. przegląd dotychczasowych doświadczeń z przeprowadzaniem generacji na ZAM-2.

W ten sposób z jednej strony udostępnia się czytelnikowi istniejący "potencjał losowy" maszyny ZAM-2 /dotąd zresztą dosyć skromny/, z drugiej - podaje mu się wskazówki, jak ten potencjał uzupełniać i jak z niego korzystać.

2. SYMBOLIKA I SPOSÓB KORZYSTANIA Z GENERATORÓW LICZB PSEUDOLOSOWYCH

Programujący na ZAM-2 przyzwyczajeni są do posługiwania się językiem SAKO i do korzystania w tym języku z różnych funkcji języka /np. PWK(X), SIN(X)/ oraz podprogramów /np. transponowania macierzy/. Toteż korzystanie z generatorów liczb pseudolosowych byłoby najłatwiejsze, gdyby miały one postać funkcji języka SAKO lub podprogramów w SAKO. Byłoby to też wygodne przy dokumentacji, względnie ogłaszaniu drukiem programów, korzystających z generatorów. Stworzenie wersji SAKO z rozszerzeniem zbioru funkcji języka jest możliwe.

Na ogół jednak, zapisywanie generatorów liczb pseudolosowych w postaci podprogramów lub funkcji języka SAKO nie jest godne polecenia, gdyż w ten sposób wydłuża się czas potrzebny do otrzymania jednej liczby pseudolosowej. Naprzykład dla otrzymania jednej liczby pseudolosowej o rozkładzie równomiernym na odcinku (0,1) wykonuje się 3 lub 4 rozkazy w kodzie wewnętrznym maszyny, a co najmniej 11 takich rozkazów, gdy generator ma postać podprogramu. Czas generacji zaś odgrywa zasadniczą rolę, gdyż w obliczeniach metodami Monte-Carlo potrzeba nieraz wielkich ilości liczb losowych. Dlatego generatory opracowuje się w formie zespołu rozkazów, który programista przepisuje w odpowiednim miejscu programu. Rozkazy te są napisane w języku SAS [3], również dla skrócenia czasu generacji /np. program generowania liczb o rozkładzie wykładniczym ma 29 rozkazów w języku SAS i ... 24 rozkazy w języku SAKO/. Na ogół bowiem wszelkie generatory liczb pseudolosowych bardzo źle zapisuje się w autokodzie.

Przed bliższym opisem korzystania z generatorów nadajmy poszczególnym generatorom trzyliterowe nazwy, np. RNA - generator

liczb pseudolosowych o rozkładzie równomiernym na odcinku (0,1) sposobem A , WYA - generator liczb pseudolosowych o rozkładzie wykładniczym z parametrem $\lambda = 1$ sposobem A , itd. Pierwsze dwie litery służą do oznaczania rozkładu generowanych liczb, trzecia wyróżnia metodę generacji tych liczb, która może odbywać się różnymi sposobami, oznaczonymi przez A, B, C,... itd. Gdy chodzi nam o jakikolwiek generator liczb pseudolosowych o takim rozkładzie, na trzecim miejscu piszemy X. Gdy chodzi nam w ogóle o jakikolwiek generator liczb pseudolosowych, będziemy go oznaczać GEN.

Podobnie jak np. rozkaz SAKO Y = LOG(X) oznacza, że przy wykonywaniu programu zmienna Y przyjmie wartość będącą logarytmem wartości zmiennej X w ustalonej skali, można w sieci działań programów zapisywać w skrócie np.

$$R = RNA(R),$$

co oznacza, że przy wykonywaniu programu zmienna R przyjmie wartość będącą kolejną liczbą pseudolosową z generatora RNA, i że ta wartość będzie funkcją poprzedniej liczby pseudolosowej z tego generatora. Ogólnie zaś, dla generatorów używających liczb z generatora RNX, można w sieci działań zapisywać np.

$$LOS = GEN(R), \qquad /1/$$

co oznacza, że przy wykonywaniu programu zmienna LOS przyjmie wartość liczby pseudolosowej generatora GEN, obliczonej na podstawie liczb pseudolosowych generatora RNX poczynając od tej liczby, która będzie wartością zmiennej R na początku obliczania; po wykonaniu obliczenia wartością zmiennej R będzie ostatnia z liczb generatora RNX użytych do wyznaczenia wartości LOS.

Jeśli generowany rozkład nie zależy od żadnych parametrów, to napisanie programu według sieci działań zawierającej /1/ polega na przepisaniu w odpowiednim miejscu programu zespołu rozkazów generatora GEN, przy czym przy programach w SAKO należy poprzedzić je deklaracją JEZYK SAS, a zakończyć deklaracją JEZYK

SAKO. Jeśli natomiast /jak to ma miejsce np. przy otrzymywaniu liczb losowych o rozkładzie Rice'a/ konieczne jest podanie parametrów rozkładu, opis generatora musi zawierać dodatkowe informacje o sposobie korzystania z niego. Generowanie zapisujemy wtedy w postaci LOS = GEN(R; θ), gdzie θ jest wektorem parametrów. Istniejące na ZAM-2 generatory, opisane dalej, są skonstruowane dla rozkładów o stałych parametrach.

Wiadomo, że np. wykonanie rozkazu Y = LOG(X) ma sens tylko wtedy, gdy zmiennej X nadano poprzednio odpowiednią wartość. Podobnie przy generowaniu liczb pseudolosowych według schematu LOS = GEN(R) na początku całych obliczeń trzeba przypisać zmiennej R wartość pierwszej liczby pseudolosowej R_0 z generatora RNX. W generatorach RNX występują też zwykle różne stałe, które również należy wprowadzić do programu na początku obliczeń. Dokładne wskazówki na ten temat zawierają opisy poszczególnych generatorów.

Podkreślmy raz jeszcze, że wyrażenia takie, jak np. LOS = GEN(R) itp. n i e s ą w naszym rozumieniu rozkazami języka SAKO i mogą znaleźć się na listach rozkazów najwyżej w formie komentarza.

Przy korzystaniu z generatorów liczb pseudolosowych trzeba jeszcze zwrócić uwagę na to, żeby nie używać oznaczeń tych zmiennych, które są już wykorzystane w stosowanym generatorze, np. R, LOS. Listę tych zmiennych podaje się w opisach poszczególnych generatorów. Z drugiej strony, można oczywiście w razie potrzeby zmienić oznaczenia zmiennych w generatorze, np. może być

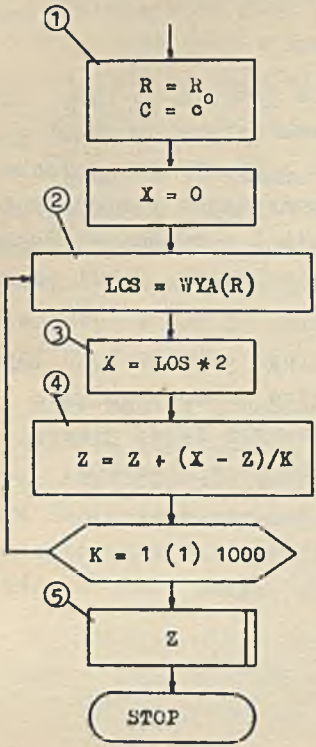
$$Y = GEN(X)$$

po konsekwentnej zmianie w przepisanych rozkazach generatora "LOS" na "Y" i "R" na "X".

Na zakończenie podajemy następujący przykład korzystania z generatorów liczb pseudolosowych.

Zadanie: Obliczyć i wydrukować średnią arytmetyczną w N- elementowej próbce prostej wartości zmiennej losowej, będącej kwadratem zmiennej losowej o rozkładzie wykładniczym ze średnią 1 przy N = 1000.

Sieć działań tego zadania jest następująca:



Zgodnie ze skrzynką 1 należy nadać zmiennym R i C wartość stałych R_0 i c z generatora RNA używanego do budowy WYA. W opisie tego generatora /w 4.1/ podano 2^{35} R_0 oraz c w postaci słów bulowskich:

2^{35} R_0: 234.642.457.001

c: 115.354.631.461.

W skrzynkach 2, 3 i 4 zapisano kolejno generowanie liczby pseudolosowej o rozkładzie wykładniczym ze średnią 1, obliczanie jej kwadratu i znajdowanie poszukiwanej średniej arytmetycznej na podstawie K próbek. Skrzynka 5 symbolizuje drukowanie wyniku Z.

Przekładając skrzynkę 2 na program w języku SAKO zaczynamy od deklaracji JEZYK SAS, następnie przepisujemy zespół rozkazów podany w opisie generatora WYA /patrz 4.2 e/, i kończymy deklaracją JEZYK SAKO.

3. BUDOWA, OPISYWANIE I DOKUMENTACJA GENERATORÓW LICZB PSEUDOLOSOWYCH.

Przy tworzeniu podanych dalej generatorów liczb pseudolosowych przyjęto następujące zasady:

a/ programy pisano w języku SAS,

b/ używano zmiennych roboczych zarówno długich jak i krótkich o nazwach XGEN, gdzie jako X wstawiono różne litery. Zmienną, której wartościami są generowane liczby, nazywano LOS /z wyjątkiem generatora RNX/, a zmienną, której wartościami są liczby pseudolosowe z generatora RNX, nazywano R.

c/ nie używano adresów zawierających numery symboliczne, aby uniknąć wystąpienia jednakowych adresów /np. przy stosowaniu w jednym programie dwóch różnych generatorów/.

Przy tworzeniu generatorów trzeba również pamiętać o tym, aby nie korzystać mechanicznie z różnych gotowych schematów nawet gdy zajdzie potrzeba np. wyciągania pierwiastka z liczby pseudolosowej, obliczania logarytmu itd., gdyż zwykle wykorzystanie posiadanych przez nas informacji /w szczególności o skali i o wymaganej dokładności/ pozwala na znaczne skrócenie zarówno programu, jak i czasu trwania obliczeń.

Sprawdzanie generatora i korzystanie z niego powinno być tak zaprojektowane, aby umożliwiło następnie łatwe przeprowadzenie losowania warstwowego /por. [4]/. Spośród opisanych dalej generatorów spełnia ten postulat i to tylko częściowo generator NOA.

Proponuje się następujący schemat opisu generatorów:

a/ nazwa generatora,
b/ algorytm generowania,
c/ sposób korzystania,
d/ średni czas generacji jednej liczby,
e/ lista rozkazów w SAS,
f/ lista pierwszych dwustu liczb,
g/ przekształcenia generowanych liczb,
h/ opis i wyniki przeprowadzonych testów,
i/ uwagi.

W a/ podaje się trzyliterową nazwę generatora, przy czym pierwsze dwie litery są przyporządkowane rozkładowi generowanych liczb, a trzecia - sposobowi generacji. W ten sposób w bibliotece programów można łatwo rozróżnić np. generatory WYA i WYB, będące generatorami liczb pseudolosowych o rozkładzie wykładniczym, otrzymanych dwoma różnymi sposobami A i B.

W punkcie b/ podaje się algorytm /niekiedy w postaci sieci działań/ i jego uzasadnienie teoretyczne oraz oczywiście wartości stałych potrzebnych do generacji.

W p. c/ podano jakie stałe należy wprowadzić do programu na początku obliczeń, jakie nazwy zmiennych /w szczególności roboczych/

są używane, w jakiej skali otrzymuje się liczbę pseudolosową i co się dzieje z zawartością rejestru B przy generacji.

W d/ mowa jest o średnim czasie, gdyż czas generacji jednej liczby nie jest na ogół wielkością stałą - tak jest np. w WYA.

Punkt e/ nie wymaga objaśnień.

W punkcie f/ podano listę pierwszych dwustu liczb. Liczby te zapisuje się w 50 wierszach po 4 wyrazy, a odczytywać je należy wierszami. Znajomość pierwszych liczb pseudolosowych jest konieczna do przeprowadzenia wstępnych obliczeń przy sprawdzaniu programu korzystającego z danego generatora - grają one wtedy podobną rolę, jak tablice funkcji.

W g/ należy podać podstawowe związki między zmiennymi losowymi o różnych rozkładach, które pozwalają na wykorzystanie generowanych liczb pseudolosowych do otrzymania liczb pseudolosowych o innym rozkładzie. Takich związków można oczywiście przytaczać wiele; w podanych w dalszym ciągu opisach generatorów przedstawiono niektóre z nich, uznane za najbardziej typowe i potrzebne.

O testach, które należy przeprowadzić przy badaniu generatorów pisano w [4]. W punkcie h/ opisów generatorów zawartych w 4.1, 4.2 i 4.3 podano częściowe wyniki testów dotąd przeprowadzonych.

4. OPISY ISTNIEJĄCYCH GENERATORÓW LICZB PSEUDOLOSOWYCH NA ZAM-2.

4.1. Generator liczb pseudolosowych o rozkładzie równomiernym na odcinku (0,1).

a/ nazwa: RNA
b/ algorytm generowania /według [1]/:

$$R = RNA(R),$$

gdzie: $RNA(R) = \left[(2^{35} R \cdot C) \bmod 2^{35}\right] \cdot 2^{-35}$,
C w zapisie ósemkowym /tj. jako słowo bulowskie/:
115.354.631.461.

Jako $2^{35} R_0$ przyjmujemy /w zapisie ósemkowym/:

234.642.457.001

lub

333.575.044.476

c/ Sposób korzystania: wg. ogólnej zasady podanej w 2.

Ponieważ algorytm generacji zależy od stałej c, na początku obliczeń należy nadać zmiennej R wartość R_0, a zmiennej C - wartość c. Nie należy używać w programie zmiennych R i C. Wartości zmiennej R są w skali binarnej 0. Zawartość rejestru B przy generowaniu nie ulega zmianie.

d/ czas generacji jednej liczby: ok. 0,004 sek.

e/ lista rozkazów w SAS:

```
UM. R
MN. C
PM. R
```

f/ lista pierwszych dwustu liczb:

+0.2299369662	+0.1834453149	+0.9599456978	+0.8883636054
+0.0450629588	+0.0620431314	+0.5489055016	+0.9645270370
+0.8993784092	+0.8544241404	+0.7275422816	+0.3434011210
+0.4877304244	+0.0319247069	+0.8589160354	+0.4262538627
+0.9273293927	+0.1366819768	+0.1503250418	+0.3570290488
+0.1014989834	+0.6253838772	+0.9970558607	+0.8660962470
+0.0034261472	+0.7130191168	+0.3261333336	+0.1140008066
+0.0799111161	+0.2166271853	+0.9400705830	+0.5352138579
+0.5751174032	+0.3990483536	+0.8606190128	+0.6818823126
+0.8743218030	+0.8126736740	+0.6725183078	+0.0675788634
+0.8476643912	+0.1431949800	+0.8672464346	+0.0110519845
+0.1938985247	+0.0533548864	+0.1867696406	+0.4799759314
+0.7841408416	+0.0266967891	+0.9672924546	+0.9347007405
+0.0056212615	+0.2113148654	+0.4830076867	+0.1720022299
+0.1054329849	+0.2636040738	+0.2898464282	+0.1688319985

+0.5342824939	+0.1920101540	+0.5692280517	+0.9260674268
+0.2902395516	+0.2007796271	+0.4718102110	+0.3122616764
+0.2624872024	+0.5337097952	+0.4612388411	+0.9073936902
+0.5750717721	+0.3178987420	+0.6578981583	+0.8466181923
+0.9306528676	+0.4074953320	+0.1826606603	+0.6640156880
+0.9542533770	+0.2274492113	+0.5006371257	+0.1363424640
+0.5370094697	+0.6172608072	+0.7649266147	+0.1267805882
+0.1799205966	+0.6747313282	+0.1603664686	+0.4286879096
+0.3375994894	+0.5997127639	+0.2472823100	+0.6093480587
+0.7620221614	+0.5378578855	+0.3052380425	+0.8537204471
+0.8462779071	+0.4243702451	+0.6767858514	+0.8081902676
+0.9683193020	+0.8277541763	+0.1112162029	+0.4243184943
+0.8347122032	+0.7935647937	+0.1083078445	+0.8025918827
+0.8243857487	+0.6881579934	+0.2620778051	+0.0361729693
+0.3323823582	+0.0424404526	+0.6045313948	+0.0546500720
+0.1136077321	+0.3956196299	+0.9494122048	+0.4677872900
+0.6265808526	+0.1389537650	+0.2359521077	+0.4092745036
+0.3771839827	+0.3595018788	+0.8726212574	+0.3804773744
+0.2624126668	+0.6838737736	+0.0808780887	+0.0941873454
+0.9141257308	+0.1219800329	+0.2389193182	+0.3183716405
+0.0427952814	+0.9107820215	+0.2254299433	+0.7199232653
+0.7812567069	+0.3580418853	+0.7633332429	+0.7084110063
+0.0284586768	+0.6860725517	+0.7635407769	+0.2798294314
+0.7932131416	+0.8754877290	+0.6687023868	+0.8603488896
+0.5379453334	+0.5089519071	+0.7969561951	+0.1500655115
+0.5224437653	+0.6149303569	+0.6856786055	+0.9667512086
+0.1476102318	+0.5114391307	+0.4352343031	+0.0896036737
+0.2992098085	+0.6497950619	+0.0527262544	+0.1029963810
+0.6916208524	+0.4583657654	+0.79574 7067	+0.2402285859
+0.2115850016	+0.1863196371	+0.5161221889	+0.2272753250
+0.2619571756	+0.7473758543	+0.0036735111	+0.1265374162
+0.1054555751	+0.5635543755	+0.3299557646	+0.1805914585
+0.2084116820	+0.4089259403	+0.1920133219	+0.6559398324
+0.5845202594	+0.2533620699	+0.2561288369	+0.6867606994
+0.1385893519	+0.1062850664	+0.5015732446	+0.1186580025

g/ przekształcenia generowanych liczb:

Niech R(a, b) oznacza zmienną losową o rozkładzie równomiernym w przedziale (a, b). Zachodzi związek:

$$R(a, b) = (b-a)\, R(0,1) + a,$$

na podstawie którego z liczb generatora RNA, pomnożonych przez (b-a) i powiększonych o a, otrzymuje się liczby pseudolosowe o rozkładzie równomiernym w przedziale (a,b).

h/ testy sprawdzające przeprowadzono według [1], str. 236, w nieco mniejszym zakresie. Badano więc przede wszystkim testem χ^2 zgodność z rozkładem równomiernym, przy czym odcinek (0,1) podzielono na 16 przedziałów, a badaniu poddano 16 tys. liczb, poczynając od pierwszej wartości R_0 podanej w a/. Testowano także wartość oczekiwaną owych 16 tysięcy liczb oraz średnią długość serii wyrazów malejących w ciągu. Na podstawie wyników testów nie odrzucono hipotezy, że badane 16 tysięcy liczb stanowi próbkę prostą z populacji o rozkładzie równomiernym na odcinku (0,1). Długość okresu aperiodyczności nie była badana.

i/ uwagi: często korzysta się z generatora RNB, będącego modyfikacją /ściślej: uogólnieniem/ generatora RNA, o algorytmie:

$$R = RNB(R)$$

gdzie

$$RNB(R) = \left[\left(C \cdot 2^{35} R\right) \bmod 2^{35+p}\right] \cdot 2^{35},$$

przy $p = 0,1,2,3,4,5.$

Lista rozkazów ma wtedy postać /przykład przy $p = 4$/:

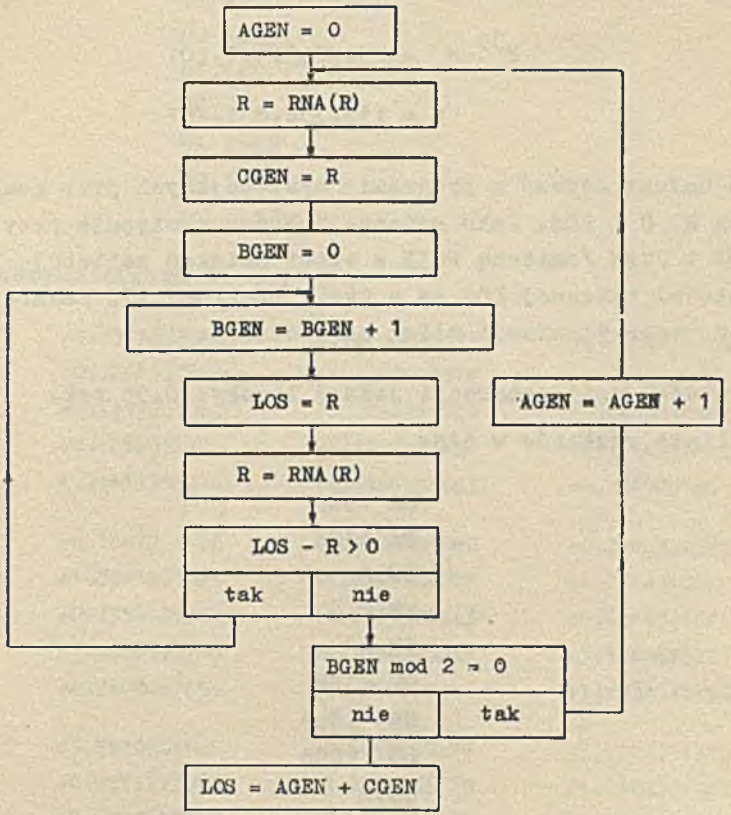
```
UM.R
MN.C
PW 4
PM.R
```

Badania tego generatora dotąd nie przeprowadzono.

4.2. Generator liczb pseudolosowych o rozkładzie wykładniczym z parametrem $\lambda = 1$.

a/ nazwa: WYA

b/ algorytm generowania: LOS = WYA(R), dany za pomocą sieci działań:



Uzasadnienie teoretyczne jest podane w [4], gdzie przytacza się dowód następującego twierdzenia:

Niech R_i będzie ciągiem zmiennych losowych o rozkładzie równomiernym na odcinku (0,1). W realizacjach tego ciągu badamy kolejne serie wyrazów malejących. Serie te numerujemy, poczynając od zera. Jeśli ilość wyrazów tworzących kolejno badaną serię jest parzysta, badamy następną serię; w przeciwnym razie tworzymy liczbę,

której część ułamkową stanowi pierwszy wyraz w serii, a część całkowitą - numer tej serii. Tak powstała liczba jest wartością zmiennej losowej o gęstości $f(x) = e^{-x}$.

c/ sposób korzystania: według ogólnej zasady podanej w 2, z tym że na początku obliczeń należy nadać wartości R_0 i o zmiennym R i C. Wartości te mogą być następujące /w zapisie ósemkowym/:

$$2^{35}\ R_0 = 234.642.457.001$$

$$o = 115.354.631.461$$

Nie należy używać w programie występujących przy generacji zmiennych R, C i LOS. Jako zmienne robocze występują przy generacji AGEN i CGEN /zmienną BGEN z sieci działań zastępuje rejestr B/. Wartości zmiennej LOS są w skali binarnej 17. Zawartość rejestru B podczas generacji ulega zniszczeniu.

d/ średni czas generacji jednej liczby: 0,06 sek.

e/ lista rozkazów w SAS:

```
UA 1009
PA AGEN
UM.R
MN.C
PM.R
PM.CGEN
UB  1009
DB  1008
UM.R
PM.LOS
MN.C
PM.R
UA.LOS
OD.R
SP - 7
PB LOS
```

```
UA LOS
PW  19
UA 1009
LW  1
SZ + 6
UA.CGEN
PW 17
DO AGEN
OK.LOS
SK + 4
UA AGEN
DO 1008
SK - 27
```

f/ lista pierwszych dwustu liczb:

+2.5489044189	+1.0319252014	+0.4262542725	+0.1366806030
+1.9970550537	+0.3261337280	+1.3990478516	+0.6818809509
+0.8126754761	+0.1431961060	+0.0110511780	+0.0533561707
+0.4799766541	+0.0266952515	+1.4830093384	+0.1920089722
+0.9260673523	+1.4612388611	+1.8466186523	+2.1363410950
+0.6172599792	+0.1267814636	+1.3376007080	+0.2472839355
+0.7620239258	+1.8081893921	+1.8347129822	+1.0424423218
+0.0546493530	+0.3956184387	+0.4677886963	+0.1389541626
+0.4092750549	+1.0808792114	+2.2254295349	+5.1500663757
+0.6149291992	+4.2402267456	+0.2272758484	+1.1054573059
+3.6867599487	+3.5020904541	+0.6695976257	+1.0027351379
+0.5380935669	+0.7511177063	+1.8720703125	+1.6408348083
+1.0394897461	+2.1747283936	+0.5252265930	+0.3735237122
+0.5311622620	+0.7160835266	+1.5745544434	+0.1269645691
+0.5424880981	+0.6755027771	+0.6611747742	+0.6058120728
+0.6380195618	+1.1416091919	+2.5896720886	+3.2533721924
+3.3471488953	+0.7353668213	+1.5153656006	+0.4823493958
+2.2916183472	+3.3794288635	+1.8783073425	+2.6385078430
+0.2758636475	+0.0425415039	+0.4151382446	+0.2203750610
+0.7395973206	+0.0945129395	+2.6193733215	+2.1536674500

+1.9404296875	+0.3180732727	+2.4406547546	+1.4982757568
+0.1877746582	+2.4209709167	+0.7629852295	+0.1802139282
+0.1814804077	+0.4694862366	+0.0452957153	+0.5941009521
+0.1521835327	+0.0551834106	+0.1675720215	+2.9864349365
+0.0528984070	+1.6060371399	+1.2461814880	+0.5545120239
+1.2085533142	+5.3084907532	+1.3285903931	+1.1223907471
+3.6390190125	+0.0192680359	+1.2279319763	+0.0386810303
+0.9645729065	+2.4426193237	+1.0737228394	+0.9495010376
+2.4348297119	+0.7484931946	+0.9636764526	+1.4185562134
+0.3044204712	+1.0979194641	+2.5325050354	+0.8474998474
+0.9098854065	+0.1916084290	+3.9863586426	+0.0769538879
+1.6156387329	+4.0561370850	+2.3311843872	+1.6199073792
+0.0664710999	+0.2097091675	+2.0741424561	+0.2562179565
+1.3263816833	+0.1729850769	+1.0039329529	+0.5713920593
+0.0597839355	+0.4254646301	+0.3372306824	+1.0495338440
+0.0940132141	+0.7642593384	+0.0315856934	+0.4477310181
+0.3637580872	+0.1192092896	+0.9574623108	+2.1563339233
+0.0083999634	+0.2942695618	+1.6032714844	+0.4375114441
+0.2293930054	+0.1235694885	+2.6681594849	+1.1233596802
+0.0169219971	+0.3236427307	+0.2900428772	+2.6600799561
+0.1877784729	+0.3437118530	+0.3480606079	+0.1251029968
+0.1278648376	+0.0031089783	+3.6445808411	+0.5750350952
+0.4527130127	+0.2541122437	+0.2950859070	+3.1124000549
+1.7910652161	+0.4408111572	+0.3374595642	+0.8891067505
+0.4141693115	+0.5760040283	+1.4683151245	+0.9290008545
+0.2594146729	+0.2435607910	+0.6886940002	+0.1450271606
+0.2355384827	+0.1355018616	+0.3331985474	+2.0492172241
+0.7049522400	+1.3666076660	+0.6270561218	+0.1923828125
+0.7194404602	+1.9856147766	+0.5940017700	+0.1479644775
+0.5110054016	+0.5700035095	+1.4216117859	+3.3311843872

g/ przekształcenia generowanych liczb:

Niech W_λ - zmienna losowa o rozkładzie wykładniczym o gęstości:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{dla } x > 0 \\ 0 & \text{poza tym.} \end{cases}$$

Zachodzi związek $W_\lambda = \frac{W_1}{\lambda}$. Zatem dzieląc liczby pseudolosowe z generatora WYA przez λ otrzymuje się liczby pseudolosowe o rozkładzie wykładniczym z parametrem λ.

Z kolei niech $\chi\sigma^2$ - zmienna losowa o rozkładzie Rayleigha z parametrem σ^2, o gęstości

$$f(x) = \begin{cases} \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) & \text{dla } x > 0 \\ 0 & \text{poza tym} \end{cases}$$

Zachodzi związek $\chi\sigma^2 = \sqrt{\frac{W_1}{2\sigma^2}} = \sigma\sqrt{2\,W_1}$. Zatem wyciągając pierwiastek kwadratowy z podwojonych liczb generatora WYA i mnożąc przez σ otrzymuje się liczby pseudolosowe o rozkładzie Rayleigha z parametrem σ^2.

h/ testy sprawdzające:

Zbadano sześć pierwszych setek generowanych liczb. Dla każdej setki obliczono /według [4]/ wartości pięciu statystyk, zdefiniowanych następująco:

Statystyka F: znajdujemy, ile spośród stu generowanych liczb należy do każdego z dziewięciu równoprawdopodobnych przedziałów, na które dzieli się zbiór wartości zmiennej losowej W_1; obliczamy wartość χ^2, oznaczamy ją przez χ^2_F i odczytaną z tablic rozkładu χ^2 wartość dystrybuanty χ^2 z 8 stopniami swobody w punkcie χ^2_F przyjmujemy jako wartość statystyki F.

Statystyka C: dzielimy zbiór wartości zmiennej losowej W_1 na trzy równoprawdopodobne przedziały I_1, I_2, I_3, i znajdujemy w

99 parach kolejnych liczb danej setki ilość par a_{jk}, w których pierwsza liczba należy do I_j, druga do I_k $(j,k = 1,2,3)$.

Znajdujemy $\chi_o^2 = \sum_j \sum_k \frac{(a_{jk} - 99\,\hat{p}_j\,\hat{p}_k)^2}{99\,\hat{p}_j\,\hat{p}_k}$, gdzie $\hat{p}_k = \frac{\sum_j a_{jk}}{99}$

dla $k = 1,2,3$ i odczytaną z tablic rozkładu χ^2 wartość dystrybuanty rozkładu χ^2 z 4 stopniami swobody w punkcie χ_o^2 przyjmujemy za wartość statystyki C.

Statystyka M: wyznaczamy średnią arytmetyczną $\bar{x}$ w setce generowanych liczb i obliczoną za pomocą rozwinięcia Edgeworth'a /patrz [2]/ wartość dystrybuanty średniej arytmetycznej w punkcie $\bar{x}$ przyjmujemy za wartość statystyki M.

Statystyka V: wyznaczamy średni kwadrat S^2 w setce generowanych liczb i obliczoną za pomocą rozwinięcia Edgeworth'a wartość dystrybuanty średniego kwadratu w punkcie S^2 przyjmujemy za wartość statystyki V.

Statystyka R: znajdujemy liczbę r serii wyrazów mniejszych i większych od mediany w setce generowanych liczb i odczytaną z tablicy 1 w [2] wartość dystrybuanty liczby serii w punkcie r przyjmujemy za wartość statystyki R.

Otrzymano wyniki:

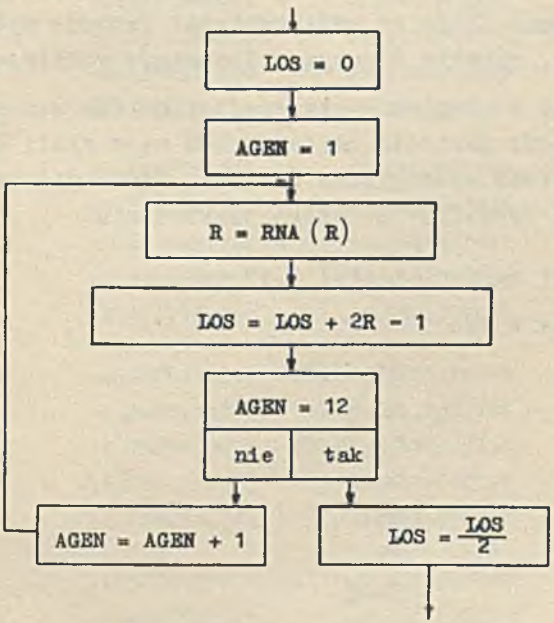
nazwa statystyki	numer setki						średnia arytmetyczna
	1	2	3	4	5	6	
F	.75	.32	.96	.73	.32	.32	.56
C	.43	.06	.70	.59	.61	.80	.51
M	.93	.50	.49	.13	.41	.17	.44
V	.84	.65	.18	.27	.48	.29	.45
R	.10	.07	.76	.76	.18	.18	.34

Jeśli liczby generatora WYA stanowią próbkę prostą z populacji o rozkładzie wykładniczym z parametrem $\lambda = 1$, to wartości każdej ze statystyk F, C, M, V i R stanowią próbkę prostą z populacji o rozkładzie równomiernym na odcinku $(0,1)$. W takim razie ich średnie arytmetyczne z kolejnych setek mają rozkład w przybliżeniu normalny z parametrami 0,5 i 0,12, a stąd prawdopodobieństwo otrzymania średniej arytmetycznej spoza przedziału $(0,26, 0.74)$ wynosi ok. 0,05. Jak widać z zamieszczonej wyżej tabelki, obliczone wartości arytmetyczne wszystkich statystyk zawierają się w tym przedziale, nie ma więc podstaw do dyskwalifikacji generatora WYA.

4.3. Generator liczb pseudolosowych o rozkładzie normalnym $N(0,1)$.

a/ nazwa: NOA

b/ algorytm generowania: LOS = NOA(R), dany flowdiagramem:



Uzasadnienie teoretyczne opiera się na centralnym twierdzeniu granicznym, według którego w szczególności suma n niezależnych zmiennych losowych o rozkładzie równomiernym na odcinku $(-1,1)$ ma rozkład asymptotycznie normalny ze średnią 0 i odchyleniem standartowym $\sqrt{\frac{n}{3}}$. Stąd jako liczbę pseudolosową z generatora NOA przyjmujemy połowę sumy dwunastu wartości z generatora RNA, pomnożonych przez dwa i zmniejszonych o jeden.

c/ sposób korzystania: według ogólnej zasady podanej w 2., z tym, że na początku obliczeń należy nadać wartości R_0 i c zmiennym R i C. Wartość c jest następująca /w zapisie ósemkowym/:

c : 115.354.631.461

Jeśli przyjąć jako 2^{35} R_0 liczbę : 234.642.457.001, to z tabeli 1 podanej dalej w h/ odczytujemy wartości kilku statystyk, obliczonych w stu kolejnych tysiącach generowanych liczb. W tabeli tej podane są także ósemkowo wartości początkowe zmiennej R w każdym generowanym tysiącu. Daje to użytkownikowi swobodę wybrania odpowiednich tysięcy, zgodnie z uwagą o losowaniu warstwowym w [4].

Nie należy używać w programie występujących przy generacji zmiennych R, C i LOS. Wartości zmiennej LOS są w skali binarnej /wiadomo, że wartość bezwzględna zmiennej LOS nie przekroczy liczby 6/. Zawartość rejestru B ulega zniszczeniu.

d/ czas generacji jednej liczby: 0,15 sek.

e/ lista rozkazów w SAS:

```
UA  1009
PA. LOS
UB  1009
UM.R
MN.C
PM.R
UA.R
LC  1
PW  4
```

```
DO.LOS
PA.LOS
SB + 4
DB  1008
SK - 10
SK + 2
SS 11
```

f/ lista pierwszych dwustu liczb:

-1.4791971808	-0.0833373684	-0.1667753439	-0.4385997998
+0.5402242811	+1.2201458057	-0.3573007714	+0.6776483199
-1.0157850133	-0.7119125146	+1.9297439493	+0.1562695261
-1.7340364382	-0.3225827822	+1.1254229490	+1.3506517829
+0.5380971003	-0.1047450006	-2.2801515032	-0.9314480396
-1.5208993470	-0.0340395523	-0.8109906213	+0.5355829680
+0.1893318538	+0.9941144194	+1.8101500971	-0.1634500856
-0.9199144049	+0.0972369574	+0.1322678858	-1.5084015382
-2.1815128904	-0.6352984700	+0.9048014879	+1.3336384268
-0.2139537586	+2.4996094164	+0.0121582616	-0.3246667879
-0.2494680723	+2.0842024032	+0.0308075789	-0.9380913330
-0.2678046459	+1.5840927223	-1.0486643035	+0.6989365770
+0.7221371578	-0.6279491708	-0.0148647511	+0.6967272256
-0.4958005184	-0.7383562876	-0.4682454085	+0.5533494912
-0.4374316121	+1.1777141327	-0.3484775005	-0.2265825476
+0.6741710957	+0.0806024689	+0.7453219909	+0.7868746584
+1.0341825029	-0.4401142513	+1.1272237720	+0.4761175979
-0.6791941915	-0.1699110279	+1.5614834176	-1.0660946239
+0.4265884561	+0.5874190712	-0.6801660527	+0.3227929054
+1.2931088153	-0.0657884050	-1.5278508216	+0.6903676968
-0.1346482821	-0.5834781788	+2.0395978699	+1.5306883985
+0.4423669018	-0.4226076435	-1.3804617049	+0.6402856605
-1.4410753977	-0.2878620857	+1.1049379073	-0.8196532791
-0.8670597365	+0.0210034288	+0.5860052267	-0.0447214860
-1.6193923121	+0.4488242976	+0.1193792466	+1.2510683117

+0.1296735303	-0.3912526686	+0.3145925645	-0.9987026062
-0.9824444326	+0.0064391466	-0.9975793092	-0.1151586249
+1.6697662380	+0.1029289467	-0.7384687746	-0.3759314418
-2.3088125279	+0.7764969999	-0.5413358295	+0.5762480432
+1.1613817029	-0.1831142046	+0.0194445830	+0.2253266471
+1.0163936466	+0.1322616441	-0.3713301048	+1.1496274667
+0.7115444718	-0.1641213531	+0.9387817578	-1.0395411178
+0.6379716402	-0.3549321210	+0.4252516497	-1.6205614209
-0.5514651891	+0.1088671451	+1.4155628448	-0.7904081969
-0.8710882999	-1.1583551187	+0.3690053970	-0.6725118877
+1.4776232895	-2.3220980829	-0.6498951800	+0.2522407882
-0.3211379061	-0.6528437911	+1.1351193823	-0.5001823222
+0.1738206325	+0.5199614353	-2.7714999188	-0.4435888408
+0.4077036986	-1.8012389401	+1.2451587114	-0.2226414615
+1.5199461812	-0.0404112376	+2.2974272510	+1.1263254350
-0.0557799311	+1.6626620321	+1.1546629863	+0.6205494888
-0.3951820433	-0.3124473430	-0.3692314140	-1.3774436032
+0.1808469463	-1.3242280837	+0.2608247576	-0.6833815975
-0.5025515724	+1.0775983194	-0.8286005808	+2.1620595930
-0.5534533374	-0.8134589465	-1.3241418153	+1.4341859911
-0.0070330100	+0.3869973058	-0.1437463183	+1.0210025460
-1.0669088922	-0.4029103108	+0.2873358363	-0.7569859065
-0.1292891139	-1.5245978748	-0.4907635273	-0.6887970436
+0.5866086660	+0.5166779840	-1.0527267577	-0.2521700365
+1.3366359295	-0.1894296361	+1.7350476906	-0.3944695666

g/ przekształcenia generowanych liczb:

Niech zmienna losowa X ma rozkład normalny $N(0,1)$. Wtedy zmienna losowa $\sigma X + m$ ma rozkład $N(m, \sigma)$. Liczby pseudolosowe o takim rozkładzie powstają z liczb generatora NOA po pomnożeniu ich przez σ i dodaniu m.

Niech teraz zmienna losowa (X,Y) ma dwuwymiarowy rozkład normalny $N(m_x, m_y, \sigma_x, \sigma_y, \rho)$. Wtedy warunkowy rozkład zmiennej Y

gdy $X = x$ jest rozkładem normalnym $N(m_y + \rho\frac{\sigma_y}{\sigma_x}(X-m_x), \sigma_y\sqrt{1-\rho^2})$. Zatem pary liczb pseudolosowych o rozkładzie $N(m_x, m_y, \sigma_x, \sigma_y, \rho)$ otrzymuje się z par liczb x_1, x_2 generatora NOA w postaci: $(\sigma_x x_1 + m_x, \sigma_y\sqrt{1-\rho^2}\,x_2 + \rho\sigma_y x_1 + m_y)$.

h/ w załączonej tabeli 1 w trzeciej i czwartej kolumnie podano wartość średnią i średni kwadrat w każdej spośród stu tysięcy kolejno generowanych liczb. W kolumnie piątej podano wartości współczynnika autokorelacji ρ_1, o którym jest mowa w [4], punkt 4. W kolumnie szóstej podano wartości χ^2, otrzymane przy podziale prostej na przedziały $(-\infty, -3)$, $(3, \infty)$ oraz na 24 równe przedziały, na które podzielono odcinek $(-3,3)$, przy czym jako rozkład teoretyczny wzięto oczywiście $N(0,1)$.

W drugiej kolumnie podano pomnożone przez 2^{35} wartości początkowe zmiennej R w każdym tysiącu.

Obliczono przedziały ufności przyjmując poziom ufności 0,05 :

dla średniej arytmetycznej:	(-0,062; 0,062)
dla średniego kwadratu:	(0,914; 1,089)
dla współczynnika ρ_1:	(-0,051; 0,053)
dla χ^2 przy 25 stopniach swobody:	(13,1; 40,6)

W tabeli zakreskowano te wartości statystyk, które nie zawierają się odpowiednio w podanych przedziałach.

Sprawdzono testem χ^2 zgodność stu wartości χ^2 w kolumnie szóstej z rozkładem χ^2 o 25 stopniach swobody, przy czym zbiór wartości podzielono na 6 części. Otrzymany wynik (9.16) nie pozwala na odrzucenie hipotezy, że ciąg 100.000 liczb z NOA jest próbką prostą z populacji o rozkładzie $N(0,1)$, gdyż przy tej hipotezie prawdopodobieństwo otrzymania wartości nie mniejszej od 9.16 przy 5 stopniach swobody wynosi ok. 10%.

Inne statystyki ze 100.000 liczb nie zostały obliczone.

Tabela 1

Sprawdzanie generatora NOA

Lp.	Wartości początkowe zmiennej R /ósemkowo/	Wartość średnia tysiąca liczb	Wartość średnia kwadratów tysiąca liczb	Wartość średnia wsp. ρ_1 w tysiącu liczb	χ^2 w tysiącu liczb
1	2	3	4	5	6
1.	234.642.457.001	-0.0158	0.9724	-0.0158	30.3
2.	367.313.154.001	-0.0034	0.9907	-0.0178	16.8
3.	377.174.651.001	-0.0478	0.9756	-0.0233	26.9
4.	131.267.346.001	0.0087	0.9768	0.0142	20.0
5.	252.573.043.001	-0.0102	0.9513	-0.0423	36.9
6.	230.307.540.001	0.0163	1.0459	-0.0274	19.7
7.	307.235.235.001	0.0026	0.9923	0.0029	17.8
8.	334.373.732.001	-0.0071	1.0373	0.0125	38.3
9.	174.743.427.001	-0.0184	1.0351	0.0197	20.1
10.	115.524.124.001	0.0295	1.0556	0.0429	30.7
11.	363.515.621.001	0.0557	1.0480	-0.0312	36.8
12.	023.720.316.001	-0.0002	1.0258	0.0139	21.6
13.	123.334.013.001	0.0261	1.0088	-0.0122	16.1
14.	127.160.510.001	-0.0135	0.9351	0.0143	19.3
15.	304.216.205.001	-0.0170	1.0012	-0.0063	20.9
16.	077.464.702.001	-0.0185	0.9924	-0.0016	22.7
17.	156.144.377.001	0.0136	1.0695	-0.0080	25.4
18.	365.035.074.001	0.0193	0.9712	-0.0106	28.4
19.	171.136.571.001	-0.0232	0.9475	0.0050	18.5
20.	237.451.266.001	-0.0215	1.1036	0.0581	35.6
21.	015.174.763.001	-0.0448	1.0075	0.0119	27.8
22.	147.131.460.001	-0.0441	1.0772	-0.0271	32.8
23.	122.277.155.001	0.0404	0.9474	0.0191	27.2
24.	163.655.652.001	-0.0281	1.0338	0.0142	28.7
25.	160.445.347.001	0.0141	0.9903	-0.0004	25.1
26.	355.446.044.001	0.0347	0.9918	-0.0206	29.7
27.	217.657.541.001	0.0180	1.0161	-0.0193	24.4
28.	374.302.236.001	-0.0493	0.9150	-0.0199	31.8
29.	330.135.733.001	-0.0221	1.0848	-0.0117	20.4
30.	310.202.430.001	0.0084	1.0302	-0.0048	16.4

1	2	3	4	5	6
31.	161.460.125.001	0.0059	0.9799	-0.0160	14.9
32.	171.146.622.001	-0.0185	0.9284	-0.0053	18.2
33.	204.046.317.001	0.0394	0.9367	-0.0178	22.0
34.	067.157.014.001	-0.0390	1.0156	-0.0324	27.8
35.	067.500.511.001	0.0514	1.0683	0.0087	37.1
36.	052.233.206.001	0.0238	1.0506	-0.0401	20.4
37.	264.176.703.001	0.0024	1.0646	-0.0189	23.4
38.	172.353.400.001	0.0244	0.9301	0.0194	18.5
39.	041.741.075.001	-0.0610	1.0215	0.0018	35.8
40.	117.537.572.001	0.0127	1.0480	0.0236	46.0
41.	250.547.267.001	-0.0420	1.0142	-0.0174	27.5
42.	321.767.764.001	-0.0379	0.9812	0.0209	26.9
43.	160.421.461.001	0.0149	1.0686	0.0052	24.6
44.	051.264.156.001	-0.0069	0.9554	-0.0042	22.1
45.	241.337.653.001	-0.0186	1.0050	-0.0169	20.6
46.	175.624.350.001	0.0530	1.0414	0.0032	24.6
47.	143.322.045.001	-0.0000	0.9740	-0.0062	17.4
48.	367.230.542.001	0.0125	0.9909	-0.0062	28.8
49.	336.350.237.001	-0.0115	1.0031	-0.0133	15.7
50.	275.700.734.001	0.0141	0.9719	-0.0170	17.7
51.	072.442.431.001	-0.0362	0.9402	0.0146	29.3
52.	371.415.126.001	-0.0031	0.9534	-0.0134	28.4
53.	237.600.623.001	0.0353	1.0582	0.0172	39.0
54.	322.175.320.001	-0.0085	0.9562	-0.0022	21.0
55.	066.003.015.001	-0.0386	1.0174	-0.0235	32.8
56.	160.021.512.001	0.0032	0.9473	-0.0041	19.5
57.	045.251.207.001	-0.0204	1.0335	-0.0221	22.1
58.	372.711.704.001	0.0070	1.0180	-0.0111	22.9
59.	225.563.401.001	0.0272	0.9662	0.0003	17.2
60.	232.646.076.001	-0.0018	0.9965	0.0150	20.6
61.	257.141.573.001	0.0120	1.0314	-0.0272	16.8
62.	167.646.270.001	0.0018	1.0301	-0.0263	20.0
63.	231.563.765.001	0.0286	0.9075	0.0071	26.0
64.	271.712.462.001	0.0046	1.0167	-0.0021	22.9
65.	175.252.157.001	-0.0245	0.9948	0.0006	30.0

1	2	3	4	5	6
66.	211.022.654.001	-0.0761	0.9687	0.0142	23.5
67.	202.004.351.001	0.0034	0.9685	0.0672	19.4
68.	015.177.046.001	-0.0031	0.9869	0.0503	14.5
69.	317.602.543.001	0.0087	1.0364	0.0343	28.0
70.	156.417.240.001	-0.0414	1.0323	0.0281	25.6
71.	216.444.735.001	-0.0055	0.9477	0.0185	18.1
72.	324.703.432.001	0.0308	0.9611	0.0128	17.8
73.	346.353.127.001	-0.0111	0.8912	-0.0061	24.2
74.	150.233.624.001	-0.0340	0.9929	-0.0116	20.6
75.	377.325.321.001	-0.0211	1.0518	-0.0233	51.6
76.	120.630.016.001	-0.0427	1.0000	-0.0454	19.7
77.	001.343.513.001	-0.0032	0.9997	0.0480	28.3
78.	266.270.210.001	-0.0002	1.0050	-0.0444	25.8
79.	024.425.705.001	-0.0453	0.9171	0.0248	24.2
80.	300.774.402.001	-0.0031	0.9992	0.0259	34.5
81.	140.554.077.001	0.0360	1.0136	-0.0142	26.5
82.	230.544.574.001	0.0136	1.0039	0.0288	41.3
83.	015.746.271.001	0.0123	1.0194	0.0095	34.5
84.	145.360.766.001	-0.0248	0.9739	0.0167	21.7
85.	104.204.463.001	0.0494	1.0356	0.0472	35.3
86.	117.241.160.001	-0.0035	1.0531	-0.0089	25.4
87.	053.506.655.001	-0.0094	1.0177	-0.0175	34.9
88.	176.165.352.001	0.0135	0.9974	0.0241	33.7
89.	354.055.047.001	-0.0181	1.0105	0.0055	17.8
90.	032.155.544.001	-0.0490	1.0139	-0.0343	24.3
91.	255.467.241.001	-0.0515	1.0534	0.0481	31,0
92.	113.211.736.001	-0.0081	1.0043	-0.0234	19.5
93.	230.145.433.001	0.0355	0.9104	0.0118	29.5
94.	071.312.130.001	-0.0091	1.0358	-0.0349	23.1
95.	323.667.625.001	0.0022	1.0006	-0.0289	26.0
96.	014.456.322.001	-0.1165	1.0130	0.0053	47.8
97.	234.642.457.001	0.0158	0.9724	-0.0303	30.3
98.	367.313.154.001	-0.0034	0.9907	-0.0178	16.8
99.	377.174.651.001	-0.0478	0.9756	-0.0235	26.9
100.	131.267.346.001	0.0087	0.9768	0.0143	20.0

5. O STOSOWANIU METOD MONTE-CARLO NA ZAM-2.

5.1. Charakterystyka ZAM-2 pod względem przystosowania do metod Monte-Carlo.

Charakterystyczną cechą obliczeń metodami Monte-Carlo jest wielokrotne powtarzanie stosunkowo prostych algorytmów obliczeń, stąd wielka przydatność maszyn cyfrowych przy stosowaniu tych metod, przy czym zasadniczą rolę gra tu szybkość operacji logicznych i arytmetycznych, natomiast mniej ważna jest sprawność urządzeń wejścia i wyjścia, a także sama pojemność maszyny.

Maszyna ZAM-2 ma względnie dużą szybkość operacji logicznych i rachunkowych w stosunku do czasu trwania czytania i drukowania, dlatego można twierdzić, że jest ona dobrze wykorzystywana przy obliczeniach Monte-Carlo, szczególnie wtedy, gdy rzadko trzeba korzystać z pamięci bębnowej. Kod wewnętrzny maszyny ZAM-2 jest dobrze przystosowany do budowy różnych generatorów liczb losowych i procesów losowych /np. rozkaz LC - lewo-cyklicznie - rzadko kiedy używany poza metodami Monte-Carlo/.

Praktyka wykazała, że szybkość maszyny ZAM-2 jest wystarczająca do przeprowadzenia wielu obliczeń metodami Monte-Carlo, np. w badaniach operacyjnych.

ZAM-2 nie ma żadnych gotowych urządzeń do generowania liczb losowych i dlatego trzeba dla niej konstruować generatory liczb pseudolosowych.

5.2. O dotychczasowym zastosowaniu generatorów istniejących na ZAM-2.

W oparciu o generator R opracowano testy sprawdzające jakość maszyny ZAM-2 /losowe wybieranie np. ścieżek bębna itd./. Generatorem NOA posłużono się np. do rozwiązania w 1962 r. pewnego zadania dla przemysłu elektronicznego. Badano układ elektryczny, którego elementami były lampy elektronowe, pochodzące z produkcji ma-

sowej, o oporach odchylających się od wartości nominalnej. Przyjęto, że odchylenia te mają rozkład normalny i generowano je za pomocą NOA.

Generator WYA zastosowano między innymi do estymowania prawdopodobieństw ergodycznych w procesie urodzin i śmierci, opisującym stan kolejki w pewnym modelu teorii kolejek ze sprzężeniem zwrotnym.

A oto tematyka niektórych innych prac z dziedziny modelowania, wykonanych na maszynie ZAM-2 przy użyciu istniejących generatorów:
a/ modelowanie i badanie działania węzła kolejowego /R. Zieliński/,
b/ rozwiązanie pewnego zagadnienia z zakresu statystycznej teorii odbioru sygnałów /A. Glińska, L. Łukaszewska i W. Rudzki/,
c/ modelowanie i badanie pewnego systemu obsługi masowej /J. Winkowski/.

5.3. Wnioski

Przedstawiony wyżej dotychczasowy "losowy potencjał" maszyny ZAM-2 nie jest duży. Uruchomionych generatorów liczb pseudolosowych jest niewiele, a przy tym nie zostały one sprawdzone w dostateczny sposób. Warto tu zauważyć, że uruchamianie i sprawdzanie generatorów nie było nigdy oficjalnie postulowane i odbywało się w miarę potrzeb /można powiedzieć: prywatnie/, przy czym dużą trudność stanowił brak przydziału czasu maszyny na ten cel. Nikt też dotychczas nie opracował polskiego słownictwa, ani nie ustalił zasad opisywania i korzystania z generatorów.

Konieczne jest podjęcie w przyszłości badań nad generowaniem liczb pseudolosowych o rozkładzie równomiernym, stanowiącym punkt wyjścia do realizowania zjawisk losowych na maszynie cyfrowej. Trzeba również zbadać wszechstronniej inne generatory /w szczególności zbadać autokorelogramy generowanych liczb/, a przez odpowiednią dokumentację wyników tego badania umożliwić łatwe stosowanie losowania warstwowego. Należałoby też uruchomić generatory liczb losowych o rozkładzie Rayleigh'a, Rice'a, Poissona itp.

Ważną sprawą byłoby też uruchomienie i sprawdzenie generatora losowych jednakowo prawdopodobnych bitów.

Zupełnie osobnym zagadnieniem, na które warto zwrócić uwagę, jest generowanie i badanie na maszynie cyfrowej realizacji procesów stochastycznych, dotąd niemal całkiem nie podjęte.

Literatura

1. MEYER H.A.: Symposium on Monte Carlo Method, John Wiley and Sons, New York, London, 1954.
2. CLARK Ch.E., HOLZ B.W.: Exponentially Distributed Random Numbers, Published for Operations Research Office, 1960.
3. Kompendium programowania w języku SAS, Oprac. K. Piałkowski, J. Swianiewicz, Prace ZAM PAN, 1962:C 3.
4. PLESZCZYŃSKA E.: Technika stosowania metod Monte-Carlo na maszynach cyfrowych, Prace IMM, Algorytmy, 1965:2, 4.
5. PLESZCZYŃSKA E.: Niektóre metody generowania realizacji procesu Poissona, Prace IMM, Algorytmy 1963:1, 2.

THE TECHNIQUE OF MONTE-CARLO METHODS FOR ZAM-2.

Summary

The paper contains:

1. Notes concerning the symbolism and way of using pseudo-random number generators, their technical details and the method of their description.
2. Detailed description and documentation of some generators, developed for ZAM-2 in the SAS language.
3. Survey of experience gained when generating random numbers on ZAM-2.

On the one hand, "random potential" of the ZAM-2 computer is being rendered available for the reader, on the other hand indications are given as to how supplement this potential and make use of it.

The paper is destined for those ZAM-2 users, who would like to carry out computations by the Monte-Carlo method, being, however, familiar with programming in the SAKO language only.

The terminology used in this paper is the one of Monte-Carlo methods, presented in paper [4].

Instytut Maszyn Matematycznych
Algorytmy N° 5
© 1965.12
519.281.2:517.27

UWAGI O POPRAWNOŚCI SFORMUŁOWANIA ZADANIA APROKSYMACJI METODĄ NAJMNIEJSZYCH KWADRATÓW

Ryszard ZIELIŃSKI

Pracę złożono 1.07.1964 r.

W pracy zdefiniowano zadanie aproksymacji metodą najmniejszych kwadratów jako poprawne, gdy spełnione są pewne warunki. Warunki te dobrano tak, aby w przypadku, gdy liczba zadanych punktów nie przekracza liczby parametrów funkcji aproksymującej, zadanie aproksymacji metodą najmniejszych kwadratów było równoważne zadaniu interpolacji. Podano dwa twierdzenia ułatwiające weryfikację poprawności zadania.

I. Dany jest ciąg punktów (x_i, y_i), $i = 1, 2, \ldots, N$, przy czym $x_i = x_j$ wtedy i tylko wtedy, gdy $i = j$. Należy tak określić parametry $\alpha = [\alpha_1, \ldots, \alpha_n]$ funkcji $y = f(x, \alpha)$, aby wielkość

$$S(\alpha) = \sum_{i=1}^{N} \left[y_i - f(x_i, \alpha)\right]^2 \qquad /1/$$

osiągała najmniejszą wartość.

Minimum $S(\alpha)$ poszukuje się zwykle przez rozwiązanie układu równań normalnych

$$\sum_{i=1}^{N} \left[y_i - f(x_i, \alpha)\right] \frac{\partial f(x_i, \alpha)}{\partial \alpha_j} = 0 \qquad (j = 1, 2, \ldots, n) \qquad /2/$$

W zastosowaniach problem wyboru funkcji aproksymującej jest często rozpatrywany niezależnie od konkretnego ciągu punktów (x_i, y_i). Ma to miejsce na przykład w fizyce teoretycznej, technice, ekonomii itp., gdzie często, w wyniku rozważań teoretycznych, ustala się postać zależności zmiennej y od zmiennej x i dopiero wtedy planuje się eksperyment mający na celu uzyskanie pewnego ciągu (x_i, y_i), na podstawie którego wyznaczone zostają parametry funkcji $f(x,\alpha)$.

Zdarza się, że minimum funkcji $S(\alpha)$ można znaleźć przez rozwiązanie, zamiast układu równań normalnych, układu równań:

$$y_i - f(x_i,\alpha) = 0 \qquad (i = 1, 2, \ldots, N) \qquad /3/$$

W dalszym ciągu zajmiemy się zagadnieniem wyboru funkcji $f(x,\alpha)$ z punktu widzenia relacji między rozwiązaniami układów /2/ i /3/.

II. Wszędzie dalej będziemy zakładali, że przestrzeń P parametrów $\alpha = [\alpha_1 \ldots, \alpha_n]$ jest n-wymiarową przestrzenią liczb rzeczywistych oraz że $f(x,\alpha)$ nie jest funkcją stałą argumentu x.

Zauważmy, że gdy istnieje takie $\alpha = \alpha^0$, iż $S(\alpha^0) = 0$, to α^0 jest rozwiązaniem układu równań /3/.

W związku z tym naturalne wydaje się wymaganie, aby funkcja $f(x,\alpha)$ spełniała następujący warunek:

(W_1): istnieje taki ciąg (x_i, y_i), $i = 1, 2, \ldots, N$, że układ równań /3/ ma dokładnie jedno rozwiązanie.

Zauważmy, że funkcje nie spełniające (W_1) mają tę właściwość, że jeżeli nawet wszystkie punkty ciągu (x_i, y_i) leżą na pewnej krzywej $y = f(x,\alpha^1)$, to istnieje wartość α spełniająca układ /3/, różna od wartości α^1. Wyłączenie takich funkcji z dalszych rozważań jest więc w pełni uzasadnione.

Jeżeli układ równań /3/ ma rozwiązania, to rozwiązania te powinniśmy uzyskać na drodze standartowego postępowania, to znaczy

na drodze rozwiązania układu /2/ równań normalnych. Wymaganie to sformułujemy w postaci warunku:

(W_2): jeżeli $\alpha = \alpha^0$ jest dla pewnego ciągu (x_i, y_j) rozwiązaniem układu /2/ równań normalnych oraz jeżeli dla tego ciągu (x_i, y_i) istnieje rozwiązanie układu /3/, to α^0 jest rozwiązaniem układu /3/.

Spełnienie warunku (W_2) gwarantuje wyznaczenie tych i tylko tych wartości α^0, dla których $S(\alpha^0) = 0$, gdy wartości takie istnieją, na drodze rozwiązania układu równań normalnych.

O funkcjach, które spełniają (W_1) oraz (W_2) będziemy mówili, że są funkcjami poprawnymi.

Przykładem funkcji niepoprawnych są: $f_1(x,\alpha) = (\alpha_1 + \alpha_2)x$ oraz $f_2(x,\alpha) = (\alpha_1^2 + \alpha_2^2)x$. Funkcja $f_1(x,\alpha)$ nie spełnia warunku (W_1), natomiast funkcja $f_2(x,\alpha)$ spełnia warunek (W_1), bo np. dla jednoelementowego ciągu $(x_1 \neq 0,\ y_1 = 0)$ istnieje rozwiązanie $\alpha_1 = \alpha_2 = 0$, natomiast nie spełnia warunku (W_2), bo np. dla jednoelementowego ciągu $(x_1 = 1,\ y_1 = 1)$ jednym z rozwiązań układu równań normalnych:

$$\left[1 - (\alpha_1^2 + \alpha_2^2)\right]\alpha_1 = 0$$

$$\left[1 - (\alpha_1^2 + \alpha_2^2)\right]\alpha_2 = 0$$

jest $\alpha_1 = \alpha_2 = 0$, podczas gdy wszystkie rozwiązania układu /3/:

$$1 - (\alpha_1^2 + \alpha_2^2) = 0$$

spełniają warunek $\alpha_1^2 + \alpha_2^2 = 1$.

III. Do tej pory mówiliśmy o wyborze funkcji niezależnie od ciągów par wartości (x_i, y_i).

Oznaczmy przez Ω zbiór par wartości (x_i, y_i) dopuszczalnych w danym zadaniu. Jeżeli np. ciąg (x_i, y_i) otrzymujemy na drodze

eksperymentu, to zbiór Ω jest zbiorem wszystkich możliwych wyników tego eksperymentu.

Założymy, że Ω jest pewnym obszarem w dwuwymiarowej przestrzeni liczb rzeczywistych.

Jeżeli Ω jest ustalone, można sformułować mocniejsze warunki dla funkcji $f(x,\alpha)$. Zauważmy, że każdy punkt (x_i, y_i) ustala pewien związek między wielkościami $\alpha_1 \ldots, \alpha_n$, natomiast n punktów (x_i, y_i) ustala n takich związków. Od funkcji $f(x,\alpha)$ będziemy w związku z tym wymagali aby spełniała warunek:

(W_3): dla każdego ciągu par $(x_i, y_i) \in \Omega$, $i = 1, 2, \ldots, N$, $N \leqslant n$ istnieje rozwiązanie układu /3/, przy czym dla $N = n$ rozwiązanie takie jest dokładnie jedno.

O funkcjach, które spełniają warunki (W_2) i (W_3) będziemy mówili, że są poprawne względem Ω.

Łatwo zauważyć, że funkcje te spełniają również warunek (W_1) oraz warunek:

(W_4): dla każdego ciągu par $(x_i, y_i) \in \Omega$, $i = 1, 2, \ldots, N$, $N \leqslant n$, każde rozwiązanie układu /2/ równań normalnych jest rozwiązaniem układu /3/.

Pewne kryterium poprawności funkcji względem Ω dają następujące twierdzenia:

Twierdzenie 1

Jeżeli funkcje $\frac{\partial f(x,\alpha)}{\partial \alpha_j}$, $j = 1, 2, \ldots, n$ są liniowo zależne, to funkcja $f(x,\alpha)$ nie jest poprawna względem Ω.

Dowód

Przepiszmy dla $N = n$ układ /2/ w rozwiniętej postaci:

$$\left.\begin{aligned}
&\left[y_1 - f(x_1,\alpha)\right]\frac{\partial f(x_1,\alpha)}{\partial \alpha_1} + \left[y_2 - f(x_2,\alpha)\right]\frac{\partial f(x_2,\alpha)}{\partial \alpha_1} + \\
&\quad \ldots\ldots + \left[y_n - f(x_n,\alpha)\right]\frac{\partial f(x_n,\alpha)}{\partial \alpha_1} = 0 \\
&\left[y_1 - f(x_1,\alpha)\right]\frac{\partial f(x_1,\alpha)}{\partial \alpha_2} + \left[y_2 - f(x_2,\alpha)\right]\frac{\partial f(x_2,\alpha)}{\partial \alpha_2} + \\
&\quad \ldots\ldots + \left[y_n - f(x_n,\alpha)\right]\frac{\partial f(x_n,\alpha)}{\partial \alpha_2} = 0 \\
&\left[y_1 - f(x_1,\alpha)\right]\frac{\partial f(x_1,\alpha)}{\partial \alpha_n} + \left[y_2 - f(x_2,\alpha)\right]\frac{\partial f(x_2,\alpha)}{\partial \alpha_n} + \\
&\quad \ldots\ldots + \left[y_n - f(x_n,\alpha)\right]\frac{\partial f(x_n,\alpha)}{\partial \alpha_n} = 0
\end{aligned}\right\} \quad /4/$$

Oznaczmy: $z_i = y_i = f(x_i,\alpha)$ oraz $c_{ij} = \frac{\partial f(x_j,\alpha)}{\partial \alpha_i}$. Gdy rząd macierzy $[c_{ij}]$ jest równy n, układ /4/, jako jednorodny układ równań liniowych względem z_i, ma dokładnie jedno rozwiązanie $z_i = 0$. Układ równań normalnych ma więc wtedy tylko takie rozwiązanie jak układ /3/. Jeżeli funkcje $\frac{\partial f(x,\alpha)}{\partial \alpha_j}$ są liniowo zależne, to rząd macierzy $[c_{ij}]$ jest mniejszy od n i układ /4/ ma nie tylko rozwiązanie $z_j = 0$.

Niech rząd macierzy $[c_{ij}]$ będzie równy n-1. Wtedy układ /4/ ma nieskończenie wiele rozwiązań spełniających:

$$z_1 : z_2 : \ldots : z_N = D_1 : D_2 : \ldots : D_N,$$

gdzie D_i $(i = 1, 2, \ldots, N)$ są wyznacznikami odpowiednich podmacierzy macierzy $[c_{ij}]$. Rozpatrzmy rozwiązanie:

$$z_i = \varepsilon D_i \qquad (i = 1, 2, \ldots, N)$$

Zapiszemy to rozwiązanie wyraźnie w postaci:

$$y_i - f(x_i, \alpha) = \varepsilon D_i \qquad (i = 1, 2, \ldots, N)$$

Jeżeli wybierzemy ε tak, aby ciąg $(x_i, y_i - \varepsilon D_i) \in \Omega$, to na mocy (W_3) istnieje rozwiązanie tego układu względem α. To rozwiązanie układu równań /4/ nie jest oczywiście rozwiązaniem układu /3/, zatem warunek (W_4) nie jest spełniony.

Analogicznie można wykazać, że warunek (W_4) nie jest spełniony, gdy rząd macierzy $[c_{ij}]$ jest mniejszy od $n - 1$, co kończy dowód twierdzenia.

Twierdzenie 2

Jeżeli przynajmniej jedna z funkcji $\frac{\partial f(x, \alpha)}{\partial \alpha_j}$ spełnia warunki:

1^0 dla każdego x istnieje takie α, że $\frac{\partial f(x, \alpha)}{\partial \alpha_j} = 0$,

2^0 $\frac{\partial f(cx, \alpha)}{\partial \alpha_j} = c \frac{\partial f(x, \alpha)}{\partial \alpha_j}$,

to funkcja $f(x, \alpha)$ nie jest poprawna względem Ω.

Dowód

Niech $N = n$ oraz niech $\frac{\partial f(x, \alpha)}{\partial \alpha_j}$ będzie funkcją spełniającą 1^0 i 2^0. W układzie równań normalnych równanie zawierające tę funkcję ma postać:

$$\sum_{i=1}^{n} \left[y_i - f(x_i, \alpha) \right] \frac{\partial f(x_i, \alpha)}{\partial \alpha_j} = 0. \qquad /5/$$

Niech $x_n \neq 0$. Na mocy 2° mamy:

$$\frac{\partial f(x_i, \alpha)}{\partial \alpha_j} = \frac{x_i}{x_n} \cdot \frac{\partial f(x_n, \alpha)}{\partial \alpha_j}$$

i równanie /5/ przyjmuje postać:

$$\frac{1}{x_n} \cdot \frac{\partial f(x_n, \alpha)}{\partial \alpha_j} \sum_{i=1}^{n} \left[y_i - f(x_i, \alpha) \right] x_i = 0.$$

Z założenia 1° równanie $\frac{\partial f(x_n, \alpha)}{\partial \alpha_j} = 0$ ma rozwiązanie $\alpha = \alpha_n$. Rozwiązanie to nie jest rozwiązaniem układu /3/ dla każdego ciągu (x_i, y_i). Jeżeli bowiem $\alpha = \alpha_n$ jest rozwiązaniem układu /3/ dla ciągu (x_i', y_i'), to nie może być rozwiązaniem dla ciągu $(x_i', y_i' + \xi_i)$, gdzie ξ_i - odpowiednio dobrane stałe. Zatem warunek (W_4) nie jest spełniony, obdo.

NOTES ON CORRECTNESS OF LEAST SQUARE APPROXIMATION

Summary

The correctness of least squares approximation is defined so as to make the correct approximation problem equivalent to the interpolation problem, if the number of parameters, appearing in approximating functions, is not less than the number of the given points. The paper contains two theorems that help to check the correctness of least squares approximation problems.

Instytut Maszyn Matematycznych
Algorytmy N° 5
© 1965.12
519.27

GENERATORY LICZB LOSOWYCH O ROZKŁADACH RAYLEIGH'A I RICE'A

Halina ŻOŁNOWSKA

Pracę złożono 27.10.1964 r.

Praca zawiera opis i podaje efektywne algorytmy generowania zmiennych losowych o rozkładach Rice'a i Rayleigh'a. Zmienne te mają duże zastosowanie w zagadnieniach technicznych szczególnie w radiokomunikacji.

1. ROZKŁADY PRAWDOPODOBIEŃSTW RICE'A I RAYLEIGH'A

Zmienna losowa P ma rozkład Rice'a jeżeli jej funkcja gęstości ma postać:

$$f(\rho) = \frac{\rho}{\sigma^2} e^{-\frac{\rho^2 + m^2}{2\sigma^2}} I_0\left(\frac{m\rho}{\sigma^2}\right) \qquad \text{dla} \qquad \rho > 0$$

$$f(\rho) = 0 \qquad \text{dla} \qquad \rho < 0 \qquad /1/$$

Funkcja $I_0\left(\frac{m\rho}{\sigma^2}\right)$ jest zmodyfikowaną funkcją Bessela, rodzaju pierwszego, stopnia zerowego i określona jest równoważnymi równościami:

$$I_0(x) = J_0(ix) \qquad /2a/$$

$$I_0(x) = \frac{1}{2\pi} \int_0^{2\pi} e^{x\cos Q}\, dQ \qquad /2b/$$

$$I_0(x) = \sum_{m=0}^{\infty} \frac{x^{2m}}{(m!)^2 \, 2^{2m}} \qquad /2o/$$

Mając na uwadze cel pracy, jakim jest zbudowanie algorytmu generowania liczb pseudolosowych o tym właśnie rozkładzie, związżemy zmienną losową o rozkładzie Rice'a prostą zależnością ze zmiennymi losowymi, dla których potrafimy w oparciu o pracę [1] zbudować generator liczb pseudolosowych. W tym celu rozpatrzmy dwie niezależne zmienne losowe X i Y o rozkładach normalnych $N(m_1, \sigma)$ i $N(m_2, \sigma)$. Łączna zmienna losowa (X, Y) ma dwuwymiarowy rozkład normalny dany funkcją gęstości

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left\{\frac{(x-m_1)^2}{\sigma^2} + \frac{(y-m_2)^2}{\sigma^2}\right\}} \qquad /3/$$

Dokonajmy zamiany zmiennych

$$\begin{aligned} x &= \rho \cos\varphi \\ y &= \rho \sin\varphi \end{aligned} \qquad /4/$$

czyli

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2} \\ \varphi &= \operatorname{arc\,tg} \frac{x}{y} \end{aligned} \qquad /5/$$

Funkcja gęstości /3/ od nowych zmiennych losowych wyrazi się wzorem:

$$g(x(\rho, \varphi), y(\rho, \varphi)) = \frac{\rho}{2\pi\sigma^2} e^{-\frac{1}{2}\left\{\frac{(\rho\cos\varphi - m_1)^2}{\sigma^2} + \frac{(\rho\sin\varphi - m_2)^2}{\sigma^2}\right\}} \qquad /6/$$

$$\text{dla} \quad \rho > 0 \quad \text{i} \quad 0 \leq \varphi \leq 2\pi$$

Z /6/ możemy znaleźć jednowymiarową funkcję rozkładu zmiennej P, co w języku technicznym oznacza uśrednianie względem fazy.

$$g_1(\rho) = \frac{\rho}{2\pi\sigma^2} \int_0^{2\pi} e^{-\frac{1}{2}\left\{\frac{(\rho\cos\varphi - m_1)^2}{\sigma^2} + \frac{(\rho\sin\varphi - m_2)^2}{\sigma^2}\right\}} d\varphi \qquad /7/$$

dla $\rho > 0$

Podstawiając $Q = \operatorname{arc\,tg}\frac{m_2}{m_1}$, stosując elementarne przekształcenie i korzystając z /2b/ otrzymamy:

$$g_1(\rho) = \frac{\rho}{\sigma^2} e^{-\frac{\rho^2 + m^2}{2\sigma^2}} I_0\left(\frac{\rho m}{\sigma^2}\right) \qquad \text{dla} \quad \rho > 0 \qquad /8/$$

$$g_1(\rho) = 0 \qquad \text{dla} \quad \rho < 0$$

Widzimy więc, że $g_1(\rho) = f(\rho)$ dla $m = \sqrt{m_1^2 + m_2^2}$ /por. /1//.

Mając na uwadze związki /5/ i /8/ stwierdzamy, że zmienna losowa będąca długością wektora, na płaszczyźnie którego składowe są niezależnymi zmiennymi losowymi o rozkładzie normalnym z jednakową wariancją σ, przy czym jedna ze składowych ma wartość oczekiwaną $\neq 0$, ma rozkład Rice'a.

Związek /5/ był punktem wyjścia do budowy generatora liczb pseudolosowych o tym właśnie rozkładzie.

Połóżmy we wzorze /1/ $m = 0$, otrzymamy:

$$f_1 z = \frac{z}{\sigma^2} e^{-\frac{z^2}{2\sigma^2}} \qquad \text{dla} \quad z > 0 \qquad /9/$$

$$z = 0 \qquad \text{dla} \quad z < 0$$

Jest to funkcja gęstości zmiennej losowej Z o rozkładzie Rayleigh'a.

Czytelnik zechce sprawdzić, że dla rozkładu Rayleigh'a pozostają w mocy wszystkie przekształcenia od /3/ do /8/ dla $m_1 = m_2 = 0$.

Stąd wniosek, że zmienna losowa Z dana wzorem

$$Z = \sqrt{X^2 + Y^2} \quad , \qquad /10/$$

gdzie X i Y są niezależnymi zmiennymi losowymi o jednakowym rozkładzie $N(0, \sigma)$, ma rozkład prawdopodobieństwa Rayleigh'a z parametrem σ. Oznaczając zmienną losową P, o rozkładzie Rice'a z parametrami m i σ przez $P_{m,\sigma}$ na podstawie wzorów /1/ i /9/ można napisać:

$$P_{0,\sigma} = Z_\sigma \quad , \qquad /11/$$

w którym zmienna losowa Z_σ ma rozkład Rayleigh'a z parametrem σ.

2. SPOSOBY GENEROWANIA LICZB PSEUDOLOSOWYCH O ROZKŁADACH RICE'A I RAYLEIGH'A.

W pierwszej kolejności omówimy sposoby generacji liczb o rozkładzie Rayleigh'a, ponieważ liczby losowe o tym rozkładzie mogą być - jak to później pokażemy - wykorzystane do generowania liczb pseudolosowych o rozkładzie Rice'a.

Po pierwsze, ponieważ funkcja gęstości /9/ da się łatwo scałkować, można skorzystać z podstawowego twierdzenia wiążącego zmienną losową R o rozkładzie równomiernym z przedziału $(0,1)$ z dowolną zmienną losową o rozkładzie ciągłym - a więc przyporządkować każdej wartości r_i zmiennej losowej R, wartość z_i zmiennej losowej Z o rozkładzie Rayleigh'a.

Mamy więc:

$$r_i = \int_0^{z_i} \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} dx = 1 - e^{-\frac{z_i^2}{2\sigma^2}}, \quad \text{stąd} \quad z_i = \sigma\sqrt{-2\lg(r_i - 1)} \qquad /12/$$

Dokonując na wartościach zmiennej losowej R, operacji wyrażonej powyższym związkiem otrzymamy wartości zmiennej losowej Z. Niech x_1, y_1 będą odpowiednio wartościami niezależnych zmiennych losowych X, Y o rozkładach N $(0,G)$. Na mocy /10/ zachodzi

$$z_1 = \sqrt{x_1^2 + y_1^2}$$

a więc mamy drugi związek, na podstawie którego można generować wartości zmiennej losowej Z.

Porównując funkcje gęstości Rayleigh'a i wykładniczą /[1], 4.2 g/ otrzymamy trzecią zależność dla zmiennych losowych o tych rozkładach:

$$Z_G = G\sqrt{2W_1} \quad , \qquad /13/$$

gdzie W_1 jest zmienną losową o rozkładzie wykładniczym z parametrem $\lambda = 1$.

Ta ostatnia zależność jest godna uwagi z tego względu, że istnieje bardzo prosty i szybki algorytm generowania liczb o rozkładzie wykładniczym zaproponowany przez von Neumana [3], który zaprogramowano na maszynę ZAM-2 [1]. Dla zmiennej losowej $P_{m,G}$ o rozkładzie Rice'a podamy 2 algorytmy.

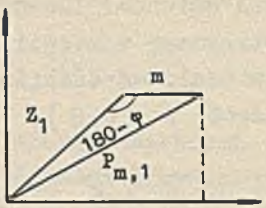
Pierwszy z nich oparty jest na zależności /5/, którą tu jeszcze raz napiszemy

$$P_{m,G} = \sqrt{X^2 + Y^2} \qquad /14/$$

Druga metoda generacji oparta jest na związku jaki zachodzi pomiędzy zmienną losową Z_1 o rozkładzie Rayleigh'a a zmienną $P_{m,1}$ o rozkładzie Rice'a /[2] str. 293/.

Z przesłanek fizycznych i rozważań geometrycznych otrzymujemy zależność /patrz rys. 1/:

$$P_{m,1} = \sqrt{z_1^2 + m^2 - 2z_1 m \cos\phi} \qquad /15/$$



Rys. 1.

Analizując wyrażenie podpierwiastkowe stwierdzamy, że m - jest wielkością stałą, Z_1 - zmienną losową o rozkładzie Rayleigh'a.

Znajdziemy rozkład zmiennej losowej Φ wstawiając do /6/ $m_1 = m_2 = 0$ i całkując względem ρ otrzymamy

$$f_1(\varphi) = \frac{1}{2\pi}\int_0^{\infty} \frac{\rho}{\sigma^2} e^{-\frac{\rho^2}{2\sigma^2}} d\rho = \frac{1}{2\pi} \qquad /16/$$

Zmienna losowa Φ ma więc rozkład równomierny na odcinku $(0,2\pi)$.

Związek /15/ ustala nam jednocześnie zależność pomiędzy wartościami zmiennej losowej $P_{m,1}$ i wartościami zmiennych losowych Z_1 i Φ.

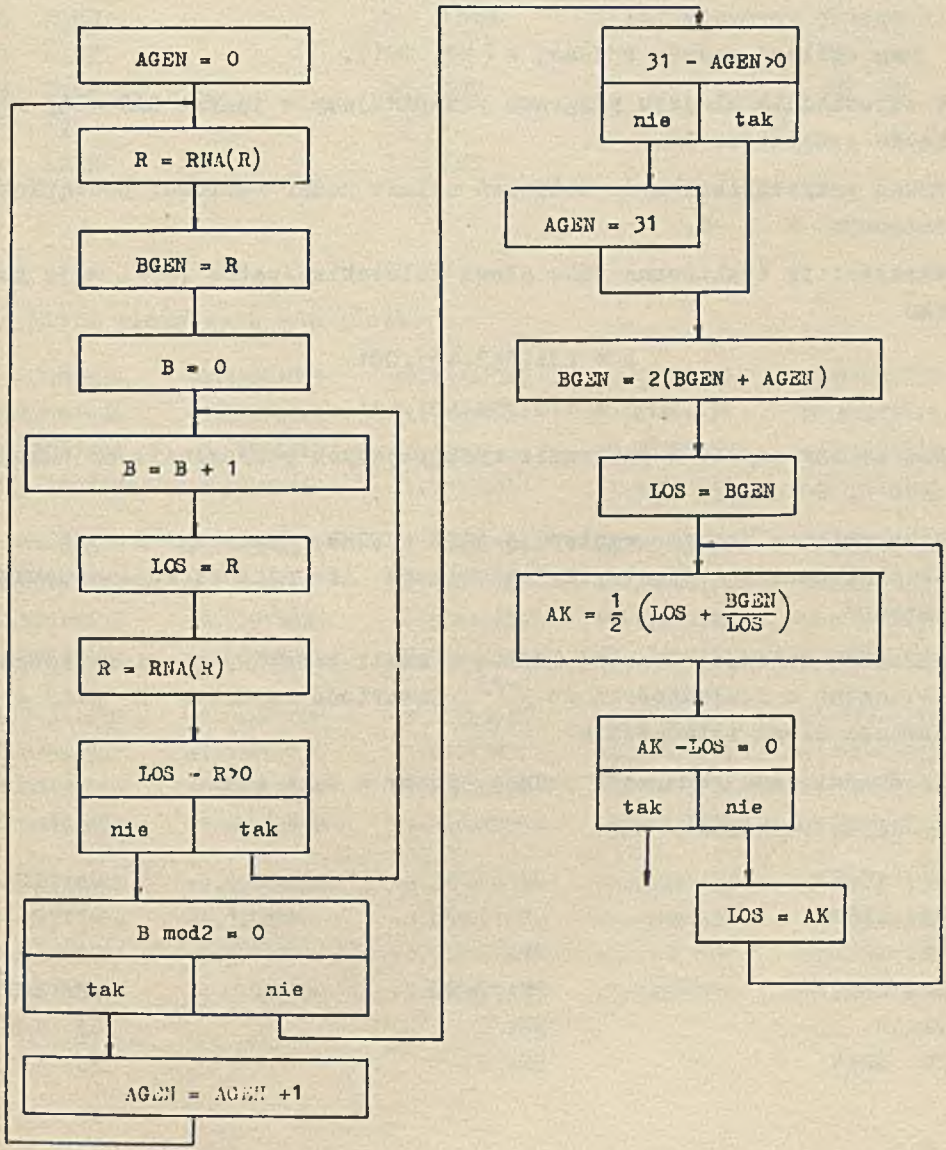
3. OPIS I DOKUMENTACJA GENERATORÓW LICZB PSEUDOLOSOWYCH O ROZKŁADACH RAYLEIGH'A I RICE'

Przy budowaniu algorytmów generujących liczby losowe wspomnianych rozkładów przyjęto zasady z pracy [1]. Zgodnie z zaproponowaną tam symboliką dla generatorów, występujące w sieciach działań /p. 3.1, 3.2, 3.3/ wyrażenie R = RNA(R) oznacza wygenerowanie wartości zmiennej losowej o rozkładzie równomiernym z przedziału (0,1) metodą A.

3.1. Opis generatora liczb pseudolosowych o rozkładzie Rayleigh'a z parametrem $\sigma = 1$.

a/ nazwa: RAA;

b/ algorytm generatora: LOS = RAA(R) dany siecią działań.



Algorytm realizuje /13/ dla $G = 1$, $Z_1 = \sqrt{2W_1}$.

Sposób otrzymywania liczb o rozkładzie wykładniczym podany jest w pracach [3] i [1], a jego teoretyczne uzasadnienie w pracy [3].

Pierwiastek liczono za pomocą iteracyjnej formuły Newtona (patrz [5]).

c/ sposób korzystania:

wg ogólnej zasady podanej w ([1] 2.1).

W odpowiednim miejscu programu przepisujemy w języku SAS ciąg rozkazów generatora RAA.

Przed przystąpieniem do obliczeń należy nadać wartości początkowe zmiennym R i C.

Wartości te traktowane jako słowa bulowskie /patrz [1]/, mają postać

$$R \equiv 234.642.457.001$$

$$C \equiv 115.354.631.461$$

Nie należy używać w programie występujących przy generacji zmiennych R, C i LOS.

Jako zmienne robocze występują AGEN i BGEN, /zmienna B z sieci działań oznacza rejestr B, a zmienna AK oznacza rejestr akumulatora/.

Wartości zmiennej losowej LOS są w skali binarnej 6. Pierwiastek obliczono z dokładnością do 2^{-25}. Zawartość rejestru B przy generacji ulega zniszczeniu.

d/ średni czas generacji jednej liczby = 0,14 sek.

e/ lista rozkazów w SAS:

UA 1009	UB 1009	UA. LOS
PA AGEN	DB 1008	OD. R
UM. R	UM. R	SP - 7
MN. C	PM. LOS	PB LOS
PM. R	MN. C	UA LOS
PM. BGEN	PM. R	PW 19

```
UA  1009
LW  1
SZ + 25
UA + 27
OD  AGEN
SP + 3
DO  AGEN
PA  AGEN
UA. BGEN
PW  17
DO  AGEN
LW  12
PA. BGEN
PA. LOS
UA. BGEN
PW  6
DZ. LOS
UA  1009
LW  35
DO. LOS
PW  1
OD. LOS
PW  2
SZ + 8
LW  2
DO. LOS
SK - 13
UA  AGEN
DO  1008
SK - 46
SS  31
```

f/ lista pierwszych 200 liczb:

+2.25783325	+1.43661039	+0.92331345	+0.52284219	+1.99852739
+0.80763028	+1.67275124	+1.16780334	+1.27489111	+0.53515415
+0.14867404	+0.32666462	+0.97977133	+0.23107050	+1.72221235
+1.54402730	+1.36093161	+1.70952557	+1.92177948	+2.06704739
+1.11109028	+0.50354858	+1.63560355	+0.70325288	+1.23452190
+1.90167835	+1.91557417	+1.44391167	+0.33060572	+0.88951631
+0.96725104	+0.52716935	+0.90473698	+1.47029119	+2.10970611
+3.20938172	+1.10899085	+2.91212245	+0.67420371	+1.48691330
+2.71542288	+2.64654090	+1.15723516	+1.41614529	+1.03739541
+1.22565592	+1.93497740	+1.81153838	+1.44186587	+2.08553423
+1.02491669	+0.86432052	+1.03069073	+1.19673134	+1.77457356
+0.50391008	+1.04162149	+1.16232835	+1.14993537	+1.10073927
+1.12961996	+1.51103238	+2.27581806	+2.55083140	+2.58733346
+1.21273934	+1.74089864	+0.98219168	+2.14084874	+2.59978056
+1.93819909	+2.29717531	+0.74278281	+0.29168801	+0.91119313
+0.66388817	+1.21622138	+0.43477317	+2.28883031	+2.07541175

+1.96999061	+0.79758873	+2.20936923	+1.73105593	+0.61282216
+2.2004414	+1.23530241	+0.60035923	+0.60246462	+0.96900490
+0.30098805	+1.09004613	+0.55169497	+0.33221146	+0.57891324
+2.44394539	+0.32526122	+1.79222517	+1.57872263	+1.05310203
+1.55470542	+3.25837045	+1.63008608	+1.49825916	+2.69778325
+0.19631174	+1.56711975	+0.27813640	+1.38893781	+2.21025723
+1.46541740	+1.37804235	+2.20672979	+1.22351478	+1.38829222
+1.68437234	+0.78028193	+1.48183552	+2.25055787	+1.30192296
+1.34899022	+0.61904384	+2.82360057	+0.39231001	+1.79757646
+2.84820597	+2.15925210	+1.79994926	+0.36461262	+0.64762458
+2.03673460	+0.71584890	+1.62873020	+0.58819210	+1.41699109
+1.06901219	+0.34578315	+0.92245647	+0.82125397	+1.44881708
+1.47919734	+1.23633186	+0.25133395	+0.94628924	+0.85294728
+0.48828151	+1.38380769	+2.07669696	+0.12961712	+0.76716455
+1.79068234	+0.93542541	+0.67733907	+0.49712831	+2.31004815
+1.49890582	+0.18396962	+0.80454056	+0.76163535	+2.30654747
+0.61282695	+0.82910906	+0.83433934	+0.50020763	+0.50569922
+0.07885435	+2.69984458	+1.07241210	+0.95154048	+0.71290076
+0.76822694	+2.49495530	+1.89265239	+0.93894926	+0.82153513
+1.33349700	+0.91013124	+1.07331809	+1.71365891	+1.36308495
+0.72029957	+0.69794047	+1.17362333	+0.53856522	+0.68635312
+0.52058304	+0.81633265	+2.02445930	+1.18739389	+1.65324361
+1.11987026	+0.62029680	+1.19953294	+1.99279353	+1.08995695
+0.54399116	+1.01094523	+1.06771071	+1.68618709	+2.58115588

g/ przekształcenie generowanych liczb:

Dla zmiennych losowych o rozkładzie Rayleigh'a zachodzi związek:

$$Z_G = G Z_1$$

wobec tego chcąc otrzymać liczby losowe z parametrem G należy liczby z opisanego generatora pomnożyć przez G.

h/ testy sprawdzające:
Generowane liczby są funkcyjnie związane z liczbami z generatora WYA, który był uruchomiony i częściowo sprawdzony przez E. Pleszczyńską [1].

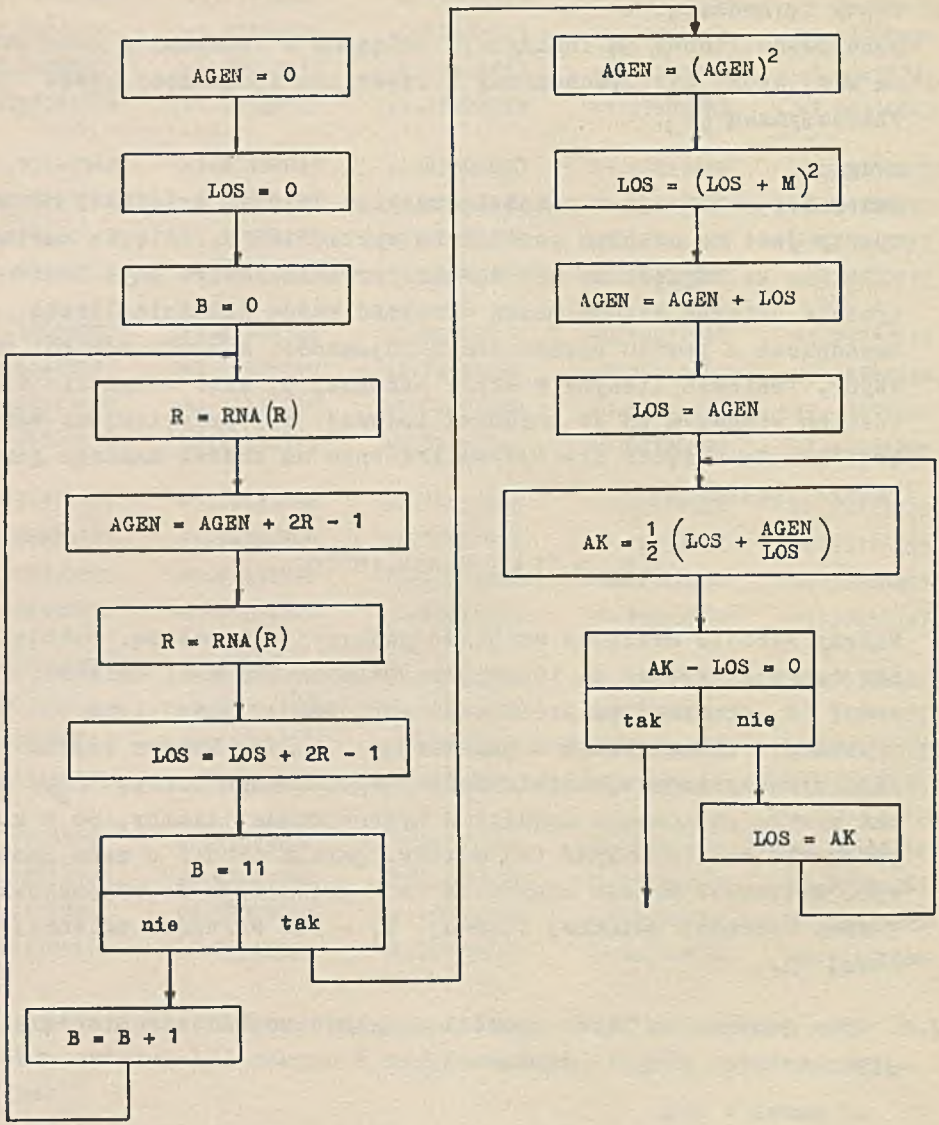
i/ uwagi:
Generator realizuje w zasadzie rozkład ucięty, a ściślej mówiąc oparty jest na uciętym rozkładzie wykładniczym. Ucięcie nastąpiło nie ze względu na metodę otrzymywania liczb, gdyż teoretycznie opisaną metodą można otrzymać każdą dodatnią liczbę, a zasadniczo z powodu ograniczonej pojemności komórki pamięci maszyny. Ponieważ liczymy w skali binarnej 5, więc wszystkim wartościom większym od 31 zmiennej losowej W_1 przypisujemy wartość 31. To ucięcie nie wpływa istotnie na budowę naszego generatora, ponieważ

$$P(W_1 > 31) = 0.344.10^{-13},$$

Należy zwrócić uwagę na szybkość generacji tą metodą. Dzieje się tak dlatego, że ze 100 wygenerowanych wartości zmiennej losowej R otrzymujemy średnio 19 wartości zmiennej losowej W_1 o rozkładzie wykładniczym z parametrem $\lambda = 1$. Metoda oparta na /12/ wykorzystuje wprawdzie każdą wygenerowaną liczbę r_1, jednak wymaga policzenia logarytmu wygenerowanej liczby, co z kolei przemawia na niekorzyść tej metody, jeżeli chodzi o czas generacji. Natomiast metoda oparta na /10/ potrzebuje do wygenerowania jednej wartości zmiennej losowej W_1 - 24 wartości zmiennej losowej R.

3.2. Opis generatora liczb pseudolosowych o rozkładzie Rice'a z parametrem $G = 1$ (wariant I)

a/ nazwa = RIA
b/ algorytm generowania: LOS = RIA (R,M) dany siecią działań.


AGEN = 0
LOS = 0
B = 0
R = RNA(R)
AGEN = AGEN + 2R - 1
R = RNA(R)
LOS = LOS + 2R - 1
B = 11
nie
tak
B = B + 1
AGEN = (AGEN)^2
LOS = (LOS + M)^2
AGEN = AGEN + LOS
LOS = AGEN
AK = 1/2 (LOS + AGEN/LOS)
AK - LOS = 0
tak
nie
LOS = AK

Algorytm generowania oparty jest na równości /14/. Sposób generowania wartości zmiennych losowych o rozkładzie normalnym czerpie swe teoretyczne uzasadnienie z centralnego twierdzenia granicznego, dla sum niezależnych zmiennych losowych o jednakowym rozkładzie i o skończonych pierwszych i drugich momentach.

Pierwiastek liczono za pomocą iteracyjnej formuły Newtona /patrz [5] /.

c/ sposób korzystania:

wg ogólnej zasady /patrz [1] 2, 3/.

Ponieważ program realizuje wartości zmiennej losowej $P_{m,1}$, przy czym parametr generowanego rozkładu $m \equiv M$ w programie, należy przed przystąpieniem do obliczeń oprócz nadania wartości początkowych zmiennym R i C nadać wartość parametru m zmiennej M. Wartość zmiennej M winna być podana jako liczba długa w skali binarnej 4.

Program nakłada ograniczenia na m : $|m| \leq 8$.

Nie należy używać w programie występujących przy generacji zmiennych R, C, M, LOS.

Jako zmienna robocza występuje w programie AGEN, /zmienna B z sieci działań oznacza rejestr B, a zmienna AK oznacza rejestr akumulatora/.
Wartości zmiennej losowej LOS są w skali binarnej 8. Pierwiastek policzono z dokładnością do 2^{-23}. Zawartość rejestru B podczas generacji ulega zniszczeniu.

d/ średni czas generacji jednej liczby: 0,37 sek.

e/ lista rozkazów w języku SAS:

```
UA 1009
PA. AGEN
PA. LOS
UB 1009
UM. R
MN. C
PM. R
UA. R
```

```
LC  1
PW  4
DO. AGEN
PA. AGEN
UM. R
MN. C
PM. R
UA. R
LC  1
PW  4
DO. LOS
PA. LOS
SB+ 29
DB  1008
SK - 18
UM. AGEN
MN. AGEN
PW  2
PA. AGEN
UA. LOS
PW  1
DO. M
PA. LOS
UM. LOS
MN. LOS
DO. AGEN
PA. AGEN
PA. LOS
UA. AGEN
PW  8
DZ. LOS
UA  1009
LW  35
DO. LOS
PW  1
OD. LOS
```

PW 2
SZ + 5
LW 2
DO. LOS
SK - 13
SS 11

f/ Lista pierwszych 200 liczb:

+1.6375624	+2.2023296	+1.9611302	+1.6626225	+0.9137866
+2.8795700	+1.2105006	+2.5896602	+1.5744209	+2.8134097
+1.7260741	+2.5837935	+1.5451219	+2.4181503	+0.9700458
+0.9431581	+1.6050495	+2.3902740	+3.1788711	+0.9611953
+2.0930420	+1.6051129	+2.7557855	+0.4605326	+0.7977520
+1.3549537	+0.7756200	+0.8900100	+2.5852504	+2.2997302
+1.3680019	+2.7264764	+1.2608241	+1.9318365	+0.1721104
+1.8302788	+1.6203486	+3.0364227	+2.4230871	+1.3743864
+0.7374958	+3.3428652	+2.2674899	+0.2266335	+0.5232202
+2.1901488	+1.8921359	+1.4782733	+0.3949858	+1.6893394
+2.0079275	+1.4691047	+0.2889381	+1.3130672	+1.9792016
+0.4856375	+0.4402635	+0.8715855	+2.0240891	+2.4686354
+1.8978381	+1.4253960	+1.8675843	+1.3844461	+2.2409858
+1.6082937	+1.3493692	+1.5548725	+1.0046109	+1.9800889
+0.2147247	+1.8334769	+0.0297279	+1.8758854	+1.2574423
+1.9044351	+2.1837332	+1.6122080	+2.0943000	+3.9414430
+1.8542635	+2.2695027	+1.7642899	+1.7146887	+1.2504414
+1.9107895	+1.8174926	+1.7388683	+3.2758058	+0.9858373
+1.0017164	+2.1703788	+0.3375882	+1.3963466	+0.6708510
+1.7946082	+1.5007679	+0.2394332	+1.7933867	+3.1842978
+3.0057613	+1.5209156	+2.4395636	+2.1503494	+0.9604841
+3.2768481	+1.8700970	+1.4405710	+2.2381065	+1.6315927
+1.1947745	+1.6877682	+0.5095367	+2.5761285	+2.7547651
+1.4943253	+2.0449595	+1.9013990	+1.6092499	+0.6876890

+2.7591678	+1.3131282	+0.5809765	+0.6727923	+0.8326484
+2.9906952	+1.3649562	+1.6471295	+1.7144944	+0.3993588
+1.4957319	+2.6360548	+1.6177517	+1.8050389	+0.9597516
+1.4507410	+0.2989012	+2.9363829	+2.5868172	+0.3123403
+1.6220793	+1.6818686	+1.3650874	+0.3000617	+1.9533413
+1.8382395	+1.7959785	+4.0093591	+1.6926544	+1.8451457
+1.7226006	+3.0191878	+1.8251202	+0.2117425	+1.3168873
+1.1467815	+3.0775850	+1.8853674	+2.4831492	+2.8422574
+1.6452376	+0.6179625	+1.2817544	+1.1592839	+0.4194064
+1.5757876	+1.4556107	+0.8784010	+1.8362126	+0.7472564
+1.5474216	+2.5842548	+2.0545344	+0.5118060	+0.9973814
+1.2565546	+1.3967868	+0.3295129	+3.0685684	+2.5344256
+1.0076572	+1.3685033	+1.1289790	+1.3645620	+2.2740759
+0.4917983	+2.0438068	+2.2763925	+1.3703985	+1.2958487
+1.8876402	+1.0884959	+1.2952361	+0.8219342	+0.3378188
+2.0079776	+0.9585812	+1.3898098	+0.4396238	+2.6232079

g/ przekształcenie generowanych liczb

Zmienna losowa P o rozkładzie Rice'a zależy od parametrów m i σ. Parametr m w najogólniejszym przypadku jest wyrażony przez $m = \sqrt{m_1^2 + m_2^2}$ /patrz wzór /8//, gdzie m_1 i m_2 są wartościami średnimi zmiennych losowych normalnych związanych ze zmienną losową o rozkładzie Rice'a zależnością /14/.

Analizując zależność /14/ stwierdzamy, że dla zmiennych losowych o rozkładzie Rice'a zachodzi związek

$$P_{m,\sigma} = \sigma P_{m,1} \qquad /18/$$

Wobec tego chcąc otrzymać liczby losowe o rozkładzie Rice'a z parametrami m i σ należy wartość parametru m zgodnie z opisem nadać zmiennej M występującej w programie przed rozpoczęciem obliczeń, a następnie otrzymane liczby z opisanego generatora pomnożyć przez σ.

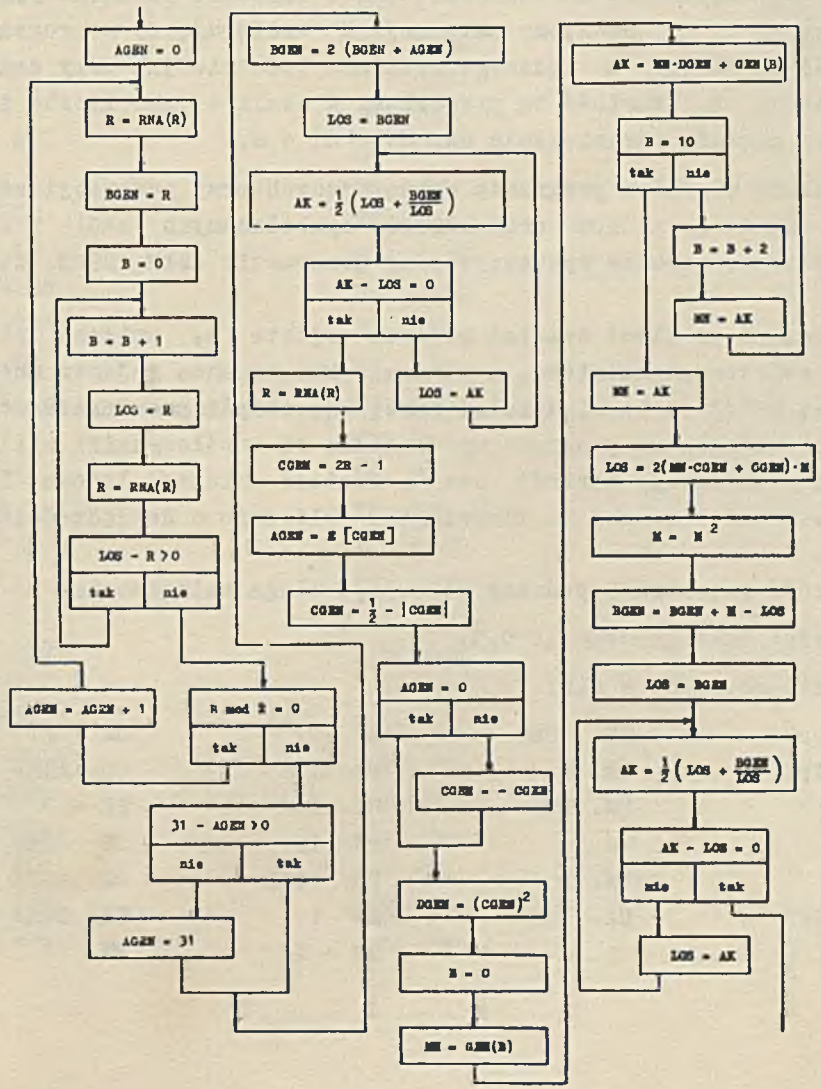
h/ testy sprawdzające

Generator oparty jest na liczbach pseudolosowych z generatorem o rozkładzie normalnym, który również był uruchomiony i sprawdzony przez E. Pleszczyńską [1].

3.3. Opis generatora liczb pseudolosowych o rozkładzie Rice'a z parametrem $\sigma = 1$ /wariant II/

a/ nazwa: RIB

b/ algorytm generowania: LOS = RIB /R,M/ dany siecią działań.



Algorytm generowania oparto na wzorze /15/.
Wartość występujących pod pierwiastkiem zmiennych losowych Z_1, o rozkładzie Rayleigh'a i Z_1^2 otrzymano w analogiczny sposób jak w generatorze RAA.
Wartość φ_1 zmiennej losowej Φ otrzymano z generatora RNA /patrz [1]/ po pomnożeniu przez 2π, to znaczy $\varphi_1 = 2\pi r_1$. Funkcję $\cos \varphi_1$ liczono za pomocą wielomianu aproksymacyjnego [5].

c/ sposób korzystania wg ogólnej zasady /patrz [1], 3/.

Przed przystąpieniem do obliczeń oprócz wartości początkowych na zmienne R i C nadajemy zmiennej M występującej w programie wartość parametru m żądanego rozkładu podobnie jak przy generacji metodą A. Wartość tę przesyłamy w skali 4 jako liczbę długą. Program nakłada ograniczenia na m: $|m| \leq 8$.

Nie należy używać w programie występujących przy generacji zmiennych: R, C, M i LOS oraz adresów symbolicznych oRO) i 1RO). Jako zmienne robocze występują przy generacji: AGEN, BGEN, CGEN i DGEN.
/Zmienna B z sieci działań oznacza rejestr B, zmienna AK oznacza rejestr akumulatora, a zmienna MN oznacza rejestr mnożnika/.
Zmienną GEN(B) z sieci działań zastępuje zespół pseudorozkazów, w postaci których do programu wprowadzone są współczynniki wielomianu aproksymującego wartość $\cos \varphi$. Wartość zmiennej losowej LOS jest w skali binarnej 8. Pierwiastek obliczono z dokładnością do 2^{-23}.
Zawartość rejestru B podczas generacji ulega zniszczeniu.

d/ średni czas generacji: 0,35

e/ lista rozkazów w SAS:

UA 1009	DB 1008	SP - 7	UA + 27
PA AGEN	UM. R	PB LOS	OD AGEN
UM. R	PM. LOS	UA LOS	SP + 3
MN. C	MN. C	PW 19	DO AGEN
PM. R	PM. R	UA 1009	PA AGEN
PM. BGEN	UA. LOS	LW 1	UA. BGEN
UB 1009	OD. R	SZ + 25	PW 17

```
DO  AGEN
LW  12
PA. BGEN
PA. LOS
UA. BGEN
PW  6
DZ. LOS
UA  1009
LW  35
DO. LOS
PW  1
OD. LOS
PW  2
SZ + 8
LW  2
DO. LOS
SK - 13
UA  AGEN
DO  1008
SK - 46
SS  31
UA. R
MN. C
PM. R
UA. R
LW  1
SN + 1
PA. CGEN
PW  17
PA  AGEN
UA + 31
OB. CGEN
PA. CGEN
UA  AGEN
SZ + 4
UA  1009
```

```
OD. CGEN
PA. CGEN
UM. CGEN
MN. CGEN
LW  1
PA. DGEN
UB  1009
UM. oRO)
MN. DGEN
LW  1
DB  1012
DO. oRO)+
PW  35
SB + 2
SK - 6
SS  10
MN. CGEN
DO. CGEN
PW  35
MN. LOS
PW  35
MN. M
LW  4
PA. LOS
SK  1RO)
PP  000
oRO)) DZ 780 +
   . SS 000
     SS 274
     SS  21
   . SB.514
   . SS 613
     KO.762 +
  PG  205 +
   . PB 286
   . KO 699 +
```

```
OB 544 +
PM. 63
1RO)UM. M
MN. M
PA. M
UA. BGEN
PW 2
DO. M
OD. LOS
PA. BGEN
PA. LOS
UA. BGEN
PW  8
DZ. LOS
UA  1009
LW  35
DO. LOS
PW  1
OD. LOS
PW  2
SZ + 4
LW  2
DO. LOS
SK - 13
```

f/ lista pierwszych 200 liczb:

+1.6510253	+1.2582253	+1.5079146	+0.4505529	+1.6285781
+1.0346148	+1.9040345	+1.4514509	+2.2284558	+0.5810886
+1.0669846	+0.6336870	+0.7245512	+1.3471404	+1.3642968
+0.6744616	+1.0363488	+0.5035486	+0.5663329	+1.8772438
+0.9212711	+1.3916316	+2.0211088	+0.4766712	+0.9672510
+0.6869528	+0.8479409	+0.7244483	+0.4340215	+0.6912587
+3.5071543	+1.1710496	+1.0113744	+0.3247345	+0.9574610
+1.5667006	+0.0857147	+1.0616538	+0.6196988	+0.7118456
+3.0006964	+2.0013664	+1.1686250	+1.9349774	+0.6372874
+1.3102477	+0.4794408	+1.3327383	+1.7465550	+1.8526106
+0.8884747	+1.7564098	+1.7921013	+1.1875963	+0.7495210
+0.5269488	+1.7830726	+0.4032912	+0.9259839	+1.2927213
+0.7804460	+0.8742785	+0.6567659	+2.2429285	+2.5657812
+0.5930039	+0.8137453	+0.6038371	+1.3345406	+1.9381991
+0.9964102	+0.6012707	+1.4439813	+1.8652599	+1.2113793
+2.2888303	+0.3422416	+0.0047108	+1.2751141	+1.6236218
+0.7515905	+0.5904213	+1.1881082	+2.6157105	+1.5371934
+1.7855533	+1.5281167	+0.3739330	+0.2558782	+1.4511357
+0.2960552	+1.3895348	+2.1016919	+0.9284874	+1.2817489
+0.7194532	+0.0732105	+0.1333000	+2.1580502	+0.8780265
+1.1902347	+0.4081697	+0.5079803	+2.1108918	+0.8965122
+0.6128247	+0.4997526	+0.7261014	+2.0365285	+2.6210029
+1.9817556	+0.3531539	+2.0483741	+0.5701515	+0.9390967
+1.9544244	+1.3030885	+0.6477274	+1.3224362	+1.4676195
+0.3144027	+1.3232705	+1.0116338	+1.3781403	+0.9656471
+1.1251570	+1.3121602	+0.8621340	+1.4604596	+0.7508043
+1.3194173	+0.6690983	+1.6287302	+1.0225480	+1.4558729
+2.0246162	+0.6823177	+0.9392264	+0.9488131	+1.6515202

+1.3793221	+1.0358366	+0.7557621	+0.4838174	+1.6088945
+0.9354079	+1.5680524	+1.4010845	+0.5482355	+1.3617374
+1.6270481	+0.9358719	+0.8740551	+1.5412842	+1.6419872
+1.4164103	+0.9599182	+1.0890661	+1.1667511	+1.7748430
+1.5837385	+1.1091589	+0.4993371	+0.3367398	+1.2417171
+1.6355183	+1.6817666	+2.2218521	+1.5870827	+0.7174676
+0.5385652	+1.6329112	+0.6010082	+0.5317811	+1.1474941
+0.6333744	+0.1649022	+1.5442694	+2.4436092	+1.7383930
+1.1167642	+1.0330847	+1.3037635	+0.2723062	+2.5648560
+2.2715384	+1.2442835	+1.8325286	+0.3013802	+1.1284091
+1.9848527	+2.1651288	+2.3816430	+0.5726639	+2.1749874
+1.5753921	+1.4166230	+0.5519522	+2.5179274	+0.7260075

g/ liczby z tego generatora są wartościami zmiennej losowej $P_{m,1}$, odnoszą się więc do nich uwagi z punktu g/ wcześniej opisanego generatora tej zmiennej losowej.

h/ opis i wyniki przeprowadzonych testów:

liczby z opisanego generatora oparte są na liczbach z generatorów uruchomionych i częściowo sprawdzonych przez E. Pleszczyńską [1].

Literatura

1. PLESZCZYŃSKA E.: Technika stosowania metod Monte Carlo na ZAM-2, Prace IMM, Algorytmy, Warszawa 1965:3, 5.

2. Metod statisticeskich ispytanij, Oprac. Buslenko N.P., Golenko D.I., Sobol I.M., Sragowicz W.G., Srejder J.A., Moskva 1962.

3. CLARK Ch. E., HOLZ B.W.: Exponentially Distributed Random Numbers, Published for Operations Research Office, 1960.

4. LEWIN B.T.: Teorija słučajnych processov i jejo primienienie w radiotechnike, Sovieckoje Radio, Moskva 1960.

5. Vyčislitelnaja Matematika, 1957:2.

6. PLESZCZYŃSKA E.: Technika stosowania metod Monte Carlo na maszynach cyfrowych, Prace IMM, Algorytmy, Warszawa 1965:2, 4.

GENERATORS OF RANDOM NUMBERS OF RAYLEIGH AND RICE'S DISTRIBUTION

Summary

The paper contains a description and presents algorithms of generation of random variables of Rayleigh and Rice's distribution.

It also gives the documentation of generation methods discussed.

THEORY OF PROGRAMMING

Instytut Maszyn Matematycznych
Algorytmy N° 5
© 1965.12
518.6:681.14-523.8

O MOŻLIWOŚCI AUTOMATYCZNEGO TŁUMACZENIA PROGRAMÓW Z AUTOKODU MARK-2 NA AUTOKOD MOST-1

Wojciech JAWORSKI
Andrzej PASIKOWSKI

Pracę złożono 4.10.1965 r.

Artykuł, przeznaczony dla pracowników i użytkowników ośrodków obliczeniowych, podaje porównanie autokodu Mark-2 dla maszyny cyfrowej E 803 z autokodem Most-1 maszyny ODRA 1003 oraz udowadnia możliwość opracowania stosunkowo prostego translatora tłumaczącego programy napisane w autokodzie Mark-2 na autokod Most-1. W artykule zamieszczono schemat blokowy takiego translatora. Zainteresowany Czytelnik winien znać przynajmniej jeden z opisywanych autokodów.

1. WSTĘP

Celem niniejszej pracy było zbadanie możliwości i wskazanie techniki tłumaczenia programów napisanych w autokodzie Mark-2 dla elektronicznej maszyny cyfrowej E 803 na autokod Most-1 dla elektronicznej maszyny cyfrowej krajowej produkcji ODRA 1003.

Biorąc dla przykładu fakt, że Centralny Resortowy Ośrodek Przetwarzania Informacji przy Instytucie Elektrotechniki wyposażony w maszynę cyfrową E 803 posiada bibliotekę zawierającą przeszło 250 własnych programów w autokodzie Mark-2, wydaje się uzasadniona podjęta przez autorów praca zmierzająca do udostępnienia bibliotek programów maszyn E 803 ośrodkom obliczeniowym wyposażonym w maszynę cyfrową ODRA 1003.

Seryjna produkcja maszyn ODRA 1003 i konieczność wyposażenia tych maszyn w bogate biblioteki programów oraz popularność na świecie maszyn E 803 /w Polsce znajdują się 3 egzemplarze/ i ich dobre oprogramowanie, wskazują na potrzebę opracowania translatora umożliwiającego w pierwszym rzędzie tłumaczenie z autokodu Mark-2 na autokod Most-1.

Ponadto, jak wiadomo, możliwość tłumaczenia programów ułatwia współpracę między określonymi ośrodkami obliczeniowymi i zapobiega dublowaniu prac nad zbliżonymi tematami.

Koncepcja opracowania translatora i jego ideowy schemat blokowy wynikają z przeprowadzonego w dalszej części artykułu porównania autokodów Mark-2 i Most-1. Niniejszą pracę podjęto dzięki inicjatywie i na zlecenie Katedry Budowy Mostów Politechniki Warszawskiej.

2. PORÓWNANIE AUTOKODÓW MARK-2 i MOST-1.

Autokody Mark-2 i Most-1 są bardzo zbliżone w swojej strukturze i stąd rozkazy tych autokodów można podzielić na dziewięć wspólnych podstawowych grup:

1. deklaracje /opisy/ i rozkaz START,
2. rozkazy arytmetyczne,
3. rozkazy funkcyjne,
4. rozkazy skokowe,
5. rozkazy organizujące pętle,
6. rozkazy korzystania z podprogramów,
7. rozkazy wejścia,
8. rozkazy wyjścia,
9. stop i oczekiwanie.

Przeprowadzona niżej analiza uwidacznia podobieństwa i różnice zapisów stosowanych w poszczególnych grupach.

2.1. Deklaracje i rozkaz START

Programowanie w autokodach Mark-2 i Most-1 wymaga umieszczenia na początku programu deklaracji określających: nazwy i rodzaje /całkowite, zmiennoprzecinkowe/ zmiennych, oraz użyty maksymalny numer referencyjny /etykieta/. Ponadto wyłącznie w autokodzie Mark-2 deklaruje się używane funkcje standartowe. Wielkości zmienne dzielimy na całkowite i zmiennoprzecinkowe. Postać zapisów jest następująca:

Zmienne	Mark-2	Most-1
całkowite np. K, K1, K2, L	SETS K(2)L	INTEGER K2L
zmiennoprzecinkowe np. A,A1,A2,A3,X,B,B1	SETV A(3)XB(1)	REAL A3XB1
Pozostałe deklaracje:		
	Mark-2	Most-1
nr referencyjny, np.23	SETR 23	LABEL 23
funkcje, np. pierwiastek kwadratowy, logarytm	SETF SQRT LOG	/nie deklaruje się/

W programach pisanych w autokodzie Most-1 po ostatniej deklaracji umieszcza się zawsze zwrot BEGIN.

Pomijając wszelkie wyjaśnienia, przykładowa postać zapisu deklaracji jest następująca:

Mark-2	Most-1
SETS K(2)L	INTEGER K2L
SETV A(3)XB(1)	REAL A3XB1
SETF SQRT LOG	LABEL 23
SETR 23	BEGIN

Rozkaz START ma identyczną postać i to samo znaczenie w obu autokodach.

W procesie tłumaczenia programów pisanych w autokodzie Mark-2 na zapis w autokodzie Most-1 należy więc przy deklaracjach zastąpić nazwy SETS, SETV, SETR odpowiednio przez INTEGER, REAL, LABEL, przekopiować listy zmiennych z pominięciem nawiasów, zignorować deklarację SETF oraz dopisać BEGIN.

2.2. Rozkazy arytmetyczne

Znaki działań: dodawania "+", odejmowania "-" i dzielenia "/", używane w zapisach rozkazów, są identyczne w obu autokodach, natomiast znak mnożenia w Most-1 jest postaci "✱", a w Marku-2 - postaci "*".

W rozkazach arytmetycznych obu autokodów argumentami mogą być także wielkości stałe.

Przykład

Mark-2	Most-1	
A=A(J+I)-B2	A=A(J+I)-B2	
X=A/Y(K-2)	X=A/Y(K-2)	
A=A*B	A=A✱B	
ZL=X(2N)	ZL=X(2N)	
A=BI+3.14	A=BI+3.14	/lub A=BI+.314'1/

W autokodzie Mark-2 dopuszczalne jest używanie zmiennych o następujących wskaźnikach złożonych: $(n^{\pm}I)$, $(K^{\pm}nI)$, $(m^{\pm}nI)$, które nie są dostępne w autokodzie Most-1.

Poza tymi osobliwościami daje się zauważyć identyczność rozkazów tej grupy i wobec tego proces tłumaczenia sprowadza się do przekopiowywania rozpatrywanego wiersza programu. Należy dodać, że przy tłumaczeniu zapisu stałych wielkości z autokodu Mark-2 na Most-1, nie pojawiają się żadne trudności, ponieważ w rozkazach Mark-2 nie jest dopuszczalne zapisywanie liczb w postaci cecha; mantysa /porównaj ostatni wiersz przykładu/.

2.3. Rozkazy funkcyjne

Standartowe funkcje używane w rozpatrywanych autokodach można zestawić w następującą listę:

Funkcja	Mark-2	Most-1
wartość bezwzględ.	MOD	ABS
pierw.kwadr.	SQRT	SQRT
funkcja wykładnicza o podstawie e	EXP	EXP
logarytm naturalny	LOG	LN
sinus *)	SIN	SIN
cosinus	COS	COS
tangens	TAN	TAN
arcus sinus	-	ARCSIN
arcus tangens	ARCTAN	ARCTAN
część ułamkowa liczby	FRAC	FRAC
część całkowita argumentu	INT	ENTIRR
standaryzacja argumentu	STAND	STAND

Z zestawienia wynika, że każdemu rozkazowi funkcyjnemu autokodu Mark-2 odpowiada rozkaz Most-1.

Proces tłumaczenia rozkazów funkcyjnych jest dość prosty, gdyż argumenty funkcji są tej samej postaci.

Należy jednak przy tłumaczeniu rozkazów funkcji trygonometrycznych SIN, COS, TAN, poprzedzić je rozkazem przygotowującym argument dla tych funkcji.

Wartości funkcji ARCTAN nie wymagają takich zabiegów, gdyż:

*) Argumenty funkcji trygonometrycznych oraz wartości funkcji kołowych w Most-1 wyrażone są w radianach, a w Marku-2 w jednostkach umownych. Jednostką umowną jest tu kąt półpełny /tj. 180°/ albo π radianów.

1/ jeżeli wartość funkcji ARCTAN w programie Mark-2 jest argumentem funkcji trygonometrycznej, to ten argument zgodnie z poprzednią uwagą musi być odpowiednio przygotowany i wobec tego nie potrzeba go w toku tłumaczenia przeliczać na radiany,

2/ jeżeli wartość funkcji ARCTAN spełnia inną rolę niż w punkcie 1., to za tym rozkazem będą rozkazy odpowiednio przeliczające tę wartość i tym samym nie ma potrzeby wyrażania wartości ARCTAN w radianach.

W tej grupie rozkazów należy zwrócić jeszcze uwagę na różnicę między funkcjami ENTIER oraz INT. Funkcja ENTIER w Most-1 określa największą liczbę całkowitą nie większą od argumentu, np. ENTIER (-5,1) = - 6, a funkcja INT w Mark-2 określa część całkowitą argumentu, np. INT(-5,1) = -5.

W obu porównywanych autokodach różnią się także funkcje FRAC. Np. w autokodzie Most-1 mamy FRAC (-5.1) = -5.1 - ENTIER (-5.1)= = 0.9, natomiast w Mark-2: FRAC (-5.1) = -5.1 - INT (-5.1) = = - 0.1.

2.4. Rozkazy skokowe

Tłumaczenie rozkazów skoku bezwarunkowego sprowadza się do zastąpienia zapisu "JUMP @" w Mark-2 przez "GO TO" w Most-1 oraz przekopiowanie numeru referencyjnego.

W Mark-2 nie ma odpowiednika rozkazu skoku zewnętrznego autokodu Most-1 o postaci "GO TO I J IF BUTTON n".

Rozkaz skoku warunkowego w autokodzie Most-1 postaci "GO TO I J K IF L=N" względnie "GO TO I J K IF A=B" jest równoważny:

GO TO I jeżeli L - N < 0
GO TO J jeżeli L - N = 0
GO TO K jeżeli L - N > 0

Stąd mamy następującą odpowiedniość *):

Mark-2	Most-1
JUMP IF L < N@K	GO TO K 0 0 IF L=N
JUMP IF L=N@K	GO TO 0 K 0 IF L=N
JUMP IF L > N@K	GO TO 0 0 K IF L=N

W przypadku rozkazów typu "JUMP UNLESS" nie trudno ustalić podobną tablicę:

Mark-2	Most-1
JUMP UNLESS L < N@K	GO TO 0 K K IF L=N
JUMP UNLESS L=N@K	GO TO K 0 K IF L=N
JUMP UNLESS L > N@K	GO TO K K 0 IF L=N

W toku tłumaczenia z autokodu Mark-2 rozkazu skoku warunkowego należy go zastąpić odpowiednikiem z autokodu Most-1, zgodnie z podanymi tablicami.

2.5. Rozkazy organizujące pętle

W tej grupie ma miejsce wzajemnie jednoznaczna odpowiedniość rozkazów z wyjątkiem CYCLE typ 1 dla zmiennych całkowitych.

Rozkazowi CYCLE I=J:K:L spełniającemu warunek konieczny $\frac{L-J}{K} \geqslant 0$ i całkowite, odpowiada w autokodzie Most-1 rozkaz "FOR I=J STEP K UNTIL L" ze słabszym ograniczeniem $\frac{L-J}{K} \geqslant 0$. Z tego wynika, że w powyższym przypadku tłumaczenie z Mark-2 na Most-1 jest jednoznaczne, lecz nie odwrotnie.

Poniższa tablica zestawia wszystkie typy rozkazów tej grupy w obu autokodach:

*) W autokodzie Most-1 rozkaz "GO TO 0" oznacza przejście do wykonywania następnego rozkazu.

Mark-2	Most-1
VARY I=J:K:L /pętla/ REPEAT I	FOR I=J STEP K REPEAT L /pętla/ END I
VARY A=B:C:L /pętla/ REPEAT A	FOR A=B STEP C REPEAT L /pętla/ END A
CYCLE I=J:K:L /pętla/ REPEAT I	FOR I=J STEP K UNTIL L /pętla/ END I
CYCLE A=B:C:D /pętla/ REPEAT A	FOR A=B STEP C UNTIL D /pętla/ END A
CYCLE I=m,n, ... /pętla/ REPEAT I	FOR I=m,n, ... /pętla/ END I
CYCLE A=x,y, ... /pętla/ REPEAT A	FOR A=x,y, ... /pętla/ END A

2.6. Rozkazy korzystania z podprogramów

W autokodzie Mark-2 program wywołuje się rozkazem SUBR a powrót do programu uzyskuje się przy pomocy EXIT.

W Most-1 odpowiadająca para rozkazów ma postać PROCEDURE n oraz END. W autokodzie Most-1 stosowany jest jeszcze rozkaz PROCEDURE I, który w Mark-2 nie ma odpowiednika /taki odpowiednik istnieje w Mark-3/. Jak z tego wynika w procesie tłumaczenia wystarczy zastąpić SUBR przez PROCEDURE, przekopiować numer referencyjny oraz zastąpić EXIT przez END.

2.7. Rozkazy wejścia

Do tej grupy zalicza się rozkazy czytania liczb i ozytania znaków:

Czynność	Mark-2	Most-1
czytanie liczb całkowitych	READ I	READ I
czytanie liczb zmiennoprzecinkowych	READ A	READ A
czytanie znaku	INPUT I	INPUT I

Identyczność postaci i funkcji rozkazów sprowadza tłumaczenie jedynie do kopiowania.

Należy tu dodać, że rozkazy READ w obu autokodach umożliwiają sterowanie biegiem programu z taśmy danych zawierającej następujące znaki:

Czynność	Mark-2	Most-1
stop	)	::
skok	n(	n:

/w Marku-2 istnieje jeszcze dodatkowo możliwość przekopiowywania tzw. szyldu *) z taśmy danych. Udogodnienie to może być jednak pominięte bez szkody dla obliczeń/.
Rozkaz INPUT I powoduje przeczytanie znaku z taśmy i ustawienie zmiennej I na wartość dziesiętną kodu dalekopisowego tego znaku.

Ponieważ kody dalekopisowe w obu autokodach są różne, to znaki, które mają być odczytane rozkazem INPUT muszą być odpowiednio przygotowane na taśmie danych, tak by uzyskany efekt działania przetłumaczonego programu Most-1 był zgodny z działaniem programu Mark-2.

*) Szyld - dowolna grupa znaków alfanumerycznych zawartych między "=" a "bl".

2.8. Rozkazy wyjścia

Rozkazy wyjścia dzielą się na trzy podgrupy:

Druk liczb
Druk znaku dalekopisowego
Rozkazy organizujące druk wyników obliczeń.

1. W autokodzie Mark-2 dwa rozkazy umożliwiają wyprowadzenie liczb całkowitych i cztery rozkazy liczb zmiennoprzecinkowych /w Most-1 łącznie 5 rozkazów/. W autokodzie Most-1 nie istnieje rozkaz drukujący liczbę zmiennoprzecinkową n-cyfrową z kropką dziesiętną umieszczoną we właściwym miejscu.

Należy zaznaczyć, że jedynym sterowanym przez Mark-2 wyjściem maszyny Elliott 803 jest urządzenie dziurkujące 5-cio ścieżkową taśmę papierową. Maszyna ODRA 1003 wyposażona jest w takie samo urządzenie; ale oprócz tego możliwy jest programowany druk na dalekopisie włączonym on-line. W autokodzie Mark-2 dziurkowanie taśmy realizuje się rozkazami PRINT, zaś w Most-1 dziurkowanie rozkazami PUNCH, natomiast druk na dalekopisie rozkazami PRINT. Zamieszczone niżej tablice porównują tylko rozkazy dziurkowania /pomijają wyjście na dalekopis/.

Liczba	Mark-2	Most-1
całkowita	PRINT I,n	PUNCH I,n
całkowita	PRINT I	PUNCH I
zmiennoprzecinkowa	PRINT A,m:n	PUNCH A,m.n
zmiennoprzecinkowa	PRINT A,n/	PUNCH A,n'
zmiennoprzecinkowa	PRINT A,n	/nie występuje/
zmiennoprzecinkowa	PRINT A	PUNCH A

Ze względu na to samo znaczenie rozkazów w każdej parze, należy przy tłumaczeniu zastępować rozkazy typu PRINT odpowiadającymi PUNCH. Ponadto w obu autokodach istnieje jeszcze możliwość warunkowego wyprowadzania wyników, z tym, że w Mark-2 drukowana jest tylko w odpowiedniej postaci wartość określonej zmiennej, a w Most-1 drukowany jest symbol tej zmiennej oraz jej wartość.

Postać rozkazów warunkowego druku:

Liczba	Mark-2	Most-1
stałoprzecinkowa	CHECK I	TEST I
zmiennoprzecinkowa	CHECK A	TEST A

2. Druk znaku z alfabetu dalekopisowego osiąga się rozkazami

Mark-2	Most-1
OUTPUT n	PUNCHOUT n
OUTPUT I	PUNCHOUT I

które spełniają tę samą rolę. Należy jednak przy korzystaniu z programu mieć na uwadze, że Mark-2 i Most-1 /a konkretnie maszyny E803 i ODRA 1003/ korzystają z kodów dalekopisowych o innych wartościach dziesiętnych poszczególnych znaków.

3. Organizację druku wyników osiąga się przez zaopatrywanie wyników w nagłówki, oddzielanie liczb lub znaków odstępami i umieszczanie liczb /lub znaków/ w nowym wierszu.

Tablica porównawcza:

Czynność	Mark-2	Most-1
druk nagłówka /klamra pokazuje miejsce na tekst/	TITLE ␣␣␣␣ bl	PUNCH ␣␣␣␣
odstępy	SPACES n	PUNCHSPACE n
odstępy	SPACES I	PUNCHSPACE I
zmiana wiersza	LINE	PUNCHLINE 1
zmiana wiersza	LINES n	PUNCHLINE n
zmiana wiersza	LINES I	PUNCHLINE I

2.9. STOP i oczekiwanie

Zatrzymanie maszyny uzyskuje się działaniem rozkazów:

Czynność	Mark-2	Most-1
stop	STOP	STOP
stop z oczekiwaniem	WAIT /nie występuje/	STOP 0 STOP n

Dla tych rozkazów tłumaczenie z autokodu Mark-2 na Most-1 jest jednoznaczne.

3. AUTOMATYZACJA PROCESU TŁUMACZENIA

Wykazane podobieństwo a niejednokrotnie identyczność rozkazów autokodu Mark-2 i rozkazów autokodu Most-1 umożliwiają automatyczne przekodowywanie programów napisanych w tych autokodach.

Zamieszczony niżej przykład podaje prosty program napisany w autokodzie Mark-2 i jego odpowiednik w Most-1.

Przykład

Program przeznaczony jest do obliczania iloczynu dwóch macierzy kwadratowych

$$\| c_{ij} \| = \| a_{ij} \| \cdot \| b_{ij} \| \qquad i,j \leqslant 5$$

Jako dane wprowadza się: rząd macierzy n oraz wierszami elementy a_{ij} i b_{ij} macierzy.

Mark-2	Most-1
SETS NMXYKR	INTEGER NMXYKR
SETV A(25) B(25) CD	REAL A25B25CD
SETR 1	LABEL 1
1) TITLE	BEGIN
MNOŻENIE MACIERZY	1:PUNCH ₰
READ N	MNOŻENIE MACIERZY ₰
VARY M=0:1:N	READ N
X=M * N	FOR M=0 STEP 1 REPEAT N
LINES 2	X=M * N
VARY Y=1:1:N	PUNCHLINE 2
SPACES 2	FOR Y=1 STEP 1 REPEAT N
READ A(X+Y)	PUNCHSPACE 2
PRINT A(X+Y),4/	READ A(X+Y)
REPEAT Y	PUNCH A(X+Y),4'
REPEAT M	END Y
LINES 4	END M
VARY M=1:1:N	PUNCHLINE 4
LINES 2	FOR M=1 STEP 1 REPEAT N
VARY Y=0:1:N	PUNCHLINE 2
X=N * Y	FOR Y=0 STEP 1 REPEAT N
READ B(X+M)	X=N * Y
SPACES 2	READ B(X+M)
PRINT B(X+M)	PUNCHSPACE 2
REPEAT Y	PUNCH B(X+M)
REPEAT M	END Y
LINES 4	END M
VARY M=0:1:N	PUNCHLINE 4
X=M * N	FOR M=0 STEP 1 REPEAT N
VARY Y=0:1:N	X=M * N
K=Y * N	FOR Y=0 STEP 1 REPEAT N
C=0	K=Y * N
VARY R=1:1:N	C=0
D=A(X+R) * B(K+R)	FOR R=1 STEP 1 REPEAT N
C=C+D	D=A(X+R) * B(K+R)
REPEAT R	C=C+D
SPACES 2	END R
PRINT C	PUNCHSPACE 2
REPEAT Y	PUNCH C
LINES 2	END Y
REPEAT M	PUNCHLINE 2
STOP	END M
START 1	STOP START 1

Z przeprowadzonego w artykule porównania obu autokodów wynika, że w procesie tłumaczenia należy zwrócić szczególną uwagę na te właściwości autokodu Mark-2, które nie mają odpowiedników w Most-1. Można tutaj wymienić:

1. wskaźniki zmiennych o postaci:
   $(n\pm I)$, $(K\pm nI)$, $(m\pm nI)$
2. przygotowanie argumentu dla funkcji trygonometrycznych
3. funkcja INT i FRAC
4. druk liczby zmiennoprzecinkowej rozkazem PRINT A,n
5. druk znaku dalekopisowego rozkazem OUTPUT n lub OUTPUT I.

Mogą być automatycznie tłumaczone te programy opracowane dla maszyny E 803, które nie współpracują z pamięcią filmową /a więc pisane w autokodzie Mark-2/ i które nie posiadają bloków pisanych w kodzie maszyny. Sposób tłumaczenia większości rozkazów uwidaczniają podane w artykule tablice porównawcze.

Osobnego potraktowania wymagają wyjątki wymienione w punktach 1 do 5.

1. Wskaźniki postaci $(n\pm I)$, $(K\pm nI)$, $(m\pm nI)$ mogą występować w autokodzie Mark-2 w rozkazach arytmetycznych, funkcyjnych, w rozkazach READ, INPUT, PRINT, CHECK, w warunku rozkazu skokowego, w pętlach VARY I=J:K:L, VARY A=B:C:L, CYCLE I=J:K:L, CYCLE A=B:C:D, CYCLE I=m,n, ...; CYCLE A=x,y, ... przy zmiennych I, A; natomiast przy J,K,L,B,C,D, z wymienionych wskaźników dopuszczalny jest tylko $(n\pm I)$.

Tłumaczenie wskaźników $(n+I)$, $(K+nI)$, $(m+nI)$, używanych w autokodzie Mark-2, sprowadza się do przestawienia ich składników.

Przykład

Mark-2	Most-1
A=B(3+I)*X(K+2J)	A=B(I+3)*X(2J+K)

W przypadku wskaźników $(n-I)$, $(K-nI)$, $(m-nI)$ można zaproponować rozwiązanie, którego schemat ilustruje następujący przykład:

Mark-2	Most-1	Uwagi:
X=A(K-3I)-2.3	/zapamiętanie w dostępnym miejscu roboczym wartości zmiennej I/ I=-3*I X=A(K+I)-2.3 /Ustawienie zmiennej I na pierwotną wartość/	W nawiasach należy wpisać odpowiednie rozkazy w kodzie maszyny ODRA 1003

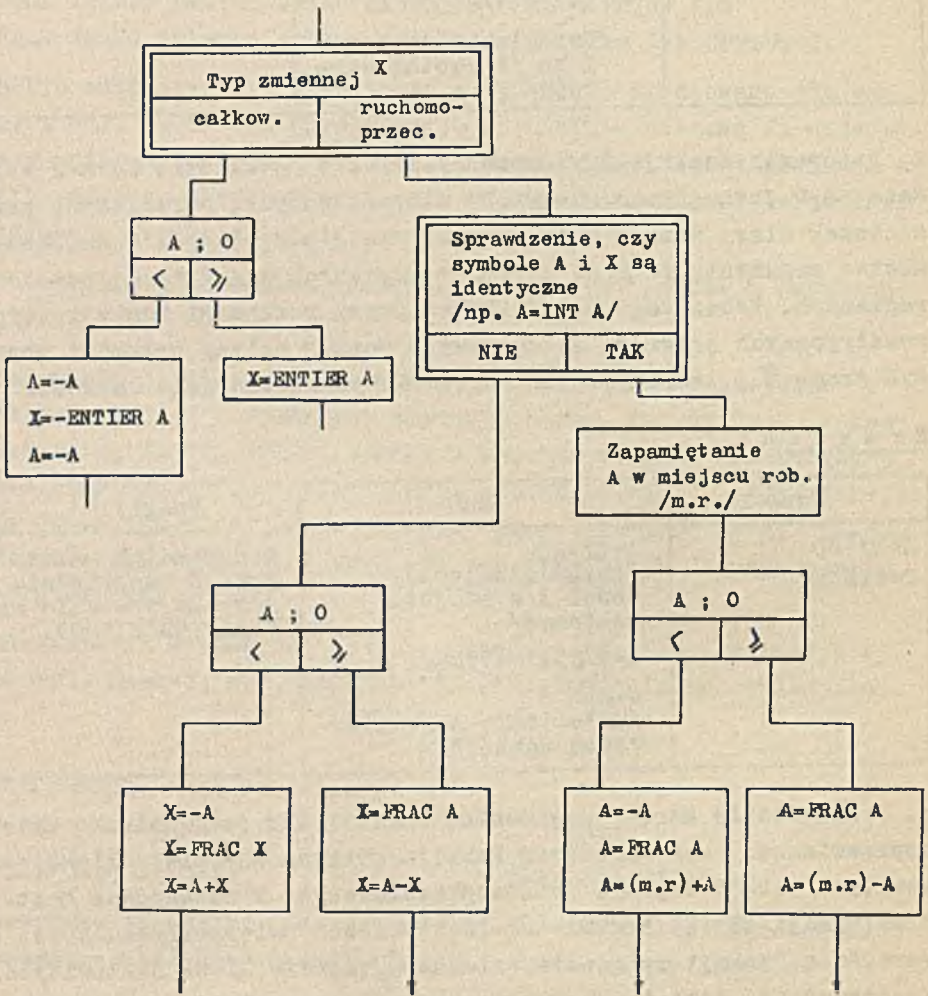
2. Argumenty funkcji trygonometrycznych w autokodzie Mark-2 wyrażane są w jednostkach umownych, niemianowanych, określonych przez stosunek miary danego kąta do miary kąta półpełnego. W autokodzie Most-1 argumenty funkcji trygonometrycznych wyrażone są zawsze w radianach. Wobec tego przed tłumaczonymi rozkazami funkcji trygonometrycznych programu autokodowego Mark-2 należy argument pomnożyć przez π, zapamiętując uprzednio pierwotną jego wartość.

P r z y k ł a d

Mark-2	Most-1	Uwagi:
A=B/180 C=SIN A	A=B/180 /zapamiętanie wartości A w miejscu roboczym/ A=A*3.1415926 C=SIN A /ustawienie A na pierwotną wartość/	W nawiasach należy wpisać odpowiednie rozkazy w kodzie maszyny ODRA 1003.

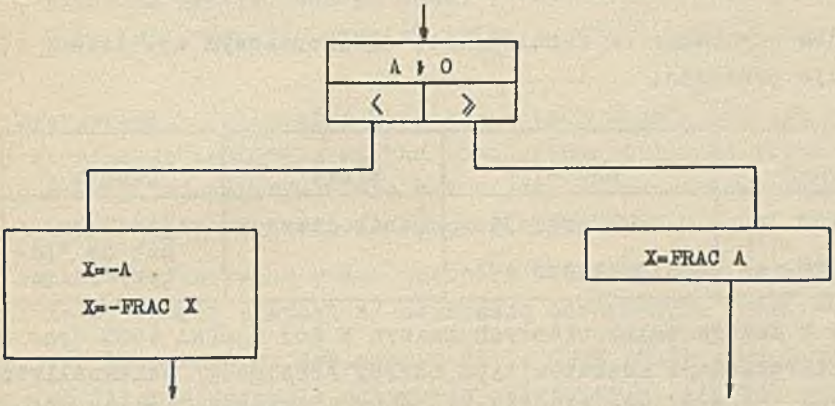
3. W autokodzie Mark-2 argumentem funkcji INT jest zmienna zmiennoprzecinkowa, zaś wartością funkcji zmienna w zapisie liczb całkowitych albo w zapisie zmiennoprzecinkowym. W autokodzie Most-1 dla funkcji ENTIER argumentem jest zmienna zmiennoprzecinkowa, a wartością funkcji wyłącznie zmienna w zapisie liczb całkowitych. W poprzedniej części artykułu wskazano dodatkowo na numeryczne różnice między tymi rozkazami. Tłumaczenie rozkazu X=INT A na X=ENTIER A można uzyskać korzystając z następującego prostego

schematu, w którym bloki narysowane podwójnie określają operacje wykonywane przez translator, a bloki narysowane linią pojedynczą określają rozkazy w autokodzie Most-1 wyprodukowane przez translator.



Rys. 1.

W produkowanych przez translator sekwencjach rozkazów, wystąpią w tym przypadku bloki w kodzie maszyny. Tłumaczenie rozkazu Mark-2 X=FRAC A można otrzymać przez wyprodukowanie za pośrednictwem translatora sekwencji rozkazów odpowiadających następującemu schematowi:



Rys. 2.

4. W autokodzie Most-1 brak rozkazu odpowiadającego rozkazowi Mark-2 postaci PRINT A,n. Dość proste rozwiązanie, uzupełniające ten brak, wymaga zastąpienia rozkazu PRINT A,n rozkazem PUNCH A, k.m spełniającym warunek n=k+m. Przy tym sposobie tłumaczenia należy rozkaz PUNCH A,k.m poprzedzić sekwencją rozkazów kodu maszynowego ODRA 1003, ustalającą liczbę cyfr k części całkowitej zmiennej A, oraz wartość numeryczną m=n-k. Szczupłe ramy artykułu nie pozwalają na bardziej szczegółowe rozwinięcie tej koncepcji.

5. Rozkazy OUTPUT n lub OUTPUT I/PUNCHOUT n lub PUNCHOUT I/ używane są w programach w dwojakim znaczeniu:

I. uzyskanie określonej perforacji na 5-cio ścieżkowej taśmie /np. znaki sterujące/

II. uzyskanie na tabulogramie dalekopisowym określonego znaku pisarskiego /po odczytaniu taśmy/.

W przypadku pierwszym argument pozostaje bez zmiany.

P r z y k ł a d

Mark-2	Most-1	Efekt
OUTPUT 21	PUNCHOUT 21	● ○ • ● ○ ●

Przypadek uzyskania na tabulogramie dalekopisowym np. litery Z ilustruje przykład:

Mark-2	Most-1	Efekt	Uwagi:
OUTPUT 31	PUNCHOUT 31	znak liter	przełączenie na "Litery"
OUTPUT 26	PUNCHOUT 17	z	

Różnice w kodach dalekopisowych maszyn E 803 i ODRA 1003 oraz dwojaka interpretacja rozkazów typu OUTPUT /PUNCHOUT/ uniemożliwiają bezbłędne automatyczne tłumaczenie takich rozkazów. W tej sytuacji można przyjąć, że translator będzie tłumaczył OUTPUT n /OUTPUT I/ na PUNCHOUT n /PUNCHOUT I/ z zastrzeżeniem dokonania ewentualnej ręcznej korekty w przetłumaczonym programie.

4. OGÓLNY SCHEMAT BLOKOWY TRANSLATORA

Translator działający zgodnie z zasadami podanymi wyżej zapewnić może automatyczne tłumaczenie programów napisanych w autokodzie Mark-2 na autokodową wersję tych programów w Most-1.

Załączony schemat blokowy translatora uwidacznia w najogólniejszym zarysie proponowany proces tłumaczenia programów z autokodu Mark-2 na autokod Most-1.

Tłumaczenie sprowadza się do odczytania rozkazu napisanego w autokodzie Mark-2, jego identyfikacji, a następnie wyprodukowania odpowiednika tego rozkazu w dopuszczalnych zapisach autokodu Most-1 /rozkazy mnemotechniczne, bloki w kodzie maszyny ODRA 1003/. Odczytywanie odbywa się znak po znaku z jednoczesnym zapamiętywaniem występujących w danym rozkazie znaków identyfikujących. W schemacie figurują następujące znaki identyfikujące /każdy wymie-

niony w cudzysłowie/: " & ", " = ", " % ", " : ", ",", " * ", "/", "sp", pierwszy w nowym wierszu znak ") ", pierwsza litera w nowym wierszu "C", "E", "I", "J", "L", "O", "P", "R", "S", "T", "W", druga litera w nowym wierszu "P", "T", "V", trzecia litera "A", "P", pierwsza występująca po znaku "=" litera "A", "C", "E", "F", "I", "L", "M", "S", "T" oraz pierwsza po znaku "sp" litera "I", "V".

Występowanie /albo nie/ któregoś z tych znaków lub ich określonej kombinacji umożliwia szybką identyfikację danego rozkazu autokodu Mark-2. Np. występowanie "=" z jednoczesnym niewystępowaniem w rozkazie znaku "sp" określa rozkaz arytmetyczny.

Funkcje, wykonywane przez niektóre bloki opisane są niżej. Używane tutaj numery postaci n/ oznaczają odpowiednie bloki schematu.

Blok:1/ Tłumaczone są deklaracje SETS, SETV, SETR oraz zapamiętywane są literowe symbole zmiennych całkowitych, których identyfikowanie następuje w bloku 20/. Deklaracja SETF jest pomijana. Po przetłumaczeniu SETR n translator produkuje BEGIN.

2/ Odczytywanie i zapamiętywanie rozkazu znak po znaku oraz zapamiętywanie wskaźników znaków identyfikujących.

3/ Badanie czy w rozkazie występuje numer referencyjny /etykieta/, który jest rozpoznawany przez występowanie ") ". Jeżeli bieżący rozkaz poprzedzony jest numerem referencyjnym, to następny blok

4/ powoduje zastąpienie ") " przez ":", ponieważ etykieta w autokodzie Most-1 jest postaci "n:".

5/ Badanie ilości spacji w bieżącym rozkazie. Jeżeli nie występuje "sp" oraz występuje albo nie "=", to mamy do czynienia z jednym z rozkazów: rozkaz arytmetyczny, EXIT, LINE, STOP, WAIT. Bloki:

6/,7/ identyfikują wymienione rozkazy. Jeżeli w rozkazie występuje /występują/ "sp", to w bloku 8/ bada się czy mamy do czynienia z TITLE. Napotkany TITLE zgodnie z 9/ jest kopiowany.

10/ Jeżeli bieżącym rozkazem nie jest TITLE /dalsza część opisu schematu blokowego dotyczy tego przypadku/ a występuje w nim dwu-

krotnie "sp", to mamy do czynienia z warunkowym rozkazem skoku.

11/,12/ i 13/ Bloki te rozróżniają rozkazy typu "JUMP IF"; "JUMP UNLESS" oraz w zależności od występowania znaku " & ", " = ", " $ ", /relacje < = > / umożliwiają, jak to pokazano dalej, wyprodukowanie odpowiednich rozkazów skoku warunkowego w autokodzie Most-1.

Bloki:

14/,15/,16/ i 17/ umożliwiają identyfikację rozkazów organizujących pętle.

Blok 18/ identyfikuje rozkazy funkcyjne, które za wyjątkiem rozkazów MOD /"M"/, INT/"I"/ są przekopiowywane /po doprowadzeniu, o ile to jest konieczne, wskaźników zmiennych do właściwej postaci/. Teksty MOD, LOG zastępowane są przez ABS, LN. Produkowanie odpowiednika funkcji INT było opisane w poprzedniej części artykułu. Proces tłumaczenia pozostałych rozkazów wejścia, wyjścia, skoku bezwarunkowego, REPEAT, SUBR oraz START odbywa się po zidentyfikowaniu rozkazu przez odpowiednie badanie pierwszej, drugiej albo trzeciej litery, w sposób uwidoczniony w blokach 19/, 20/ i 21/.

Blok 24/ powoduje zatrzymanie maszyny po zakończeniu procesu tłumaczenia, to znaczy po napotkaniu i przekopiowaniu rozkazu START n. W schemacie oznaczono przez [W] podprogram, który bada postać wskaźników zmiennej i o ile zachodzi potrzeba produkuje odpowiednią sekwencję zmieniającą wskaźniki oraz przekształca postać rozkazu. Podprogram ten sprzężony jest zawsze z blokiem 22/ i ewentualnie 23/, które umożliwiają, jeśli to konieczne, powrót do pierwotnych wartości całkowitych zmienianych w podprogramie.

Uwaga:

W schemacie pominięto blok badający występowanie znaku "?". Znak jest używany dla zaznaczenia, że dany wiersz nie ma być tłumaczony translatorem autokodu Mark-2 /np. w przypadku błędnej perforacji/ znak ten jest umieszczany zawsze na końcu ciągu znaków danego wiersza.

5. UWAGI KOŃCOWE

W artykule wykazano możliwość opracowania translatora tłumaczącego programy z autokodu Mark-2 na Most-1, a pominięto zupełnie zagadnienie tłumaczenia programu napisanego w Mark-2 na binarną postać wprowadzaną bezpośrednio do maszyny ODRA 1003. Stanowisko takie wydaje się uzasadnione, ponieważ przy opisanym działaniu translatora można, po dokonaniu niewielkich zmian w opisach programów dla maszyny Elliott, włączyć je do biblioteki maszyny cyfrowej ODRA 1003. Ponadto w przypadkach modyfikacji programu, użytkownik maszyny ODRA 1003 wyposażony w automatycznie przetłumaczony tekst nie musi się w tym celu zapoznawać z autokodem Mark-2. Autorzy wychodząc z porównania parametrów maszyn E 803 i ODRA 1003 zakładają, że translator winien być opracowany dla maszyny E 803. Przyjęcie takiego założenia pociąga za sobą konieczność produkowania tłumaczonego programu w kodzie dalekopisowym ODRA 1003, co nie przedstawia żadnych trudności.

W artykule sprecyzowano wymagania jakim winien odpowiadać tłumaczony program. Mianowicie: musi być napisany w autokodzie Mark-2 i nie może zawierać bloków w kodzie maszyny. Ostatnie ograniczenie można złagodzić, żądając od translatora wykonywania tłumaczenia z ignorowaniem bloków napisanych w kodzie maszyny, natomiast ignorowane fragmenty przeprogramować ręcznie i włączyć do uzyskanego maszynowo tłumaczenia. Opisaną dla tego przypadku procedurę można zautomatyzować przez opracowanie bibliotecznego programu, spełniającego następujące korektorskie funkcje:

1. wymazywanie rozkazów
2. zastępowanie rozkazów
3. dopisywanie rozkazów.

Warto dodać, że istnienie takiego programu ułatwi także wprowadzenie poprawek, które mogą być konieczne w przypadku występowania w programie Mark-2 rozkazów typu OUTPUT.

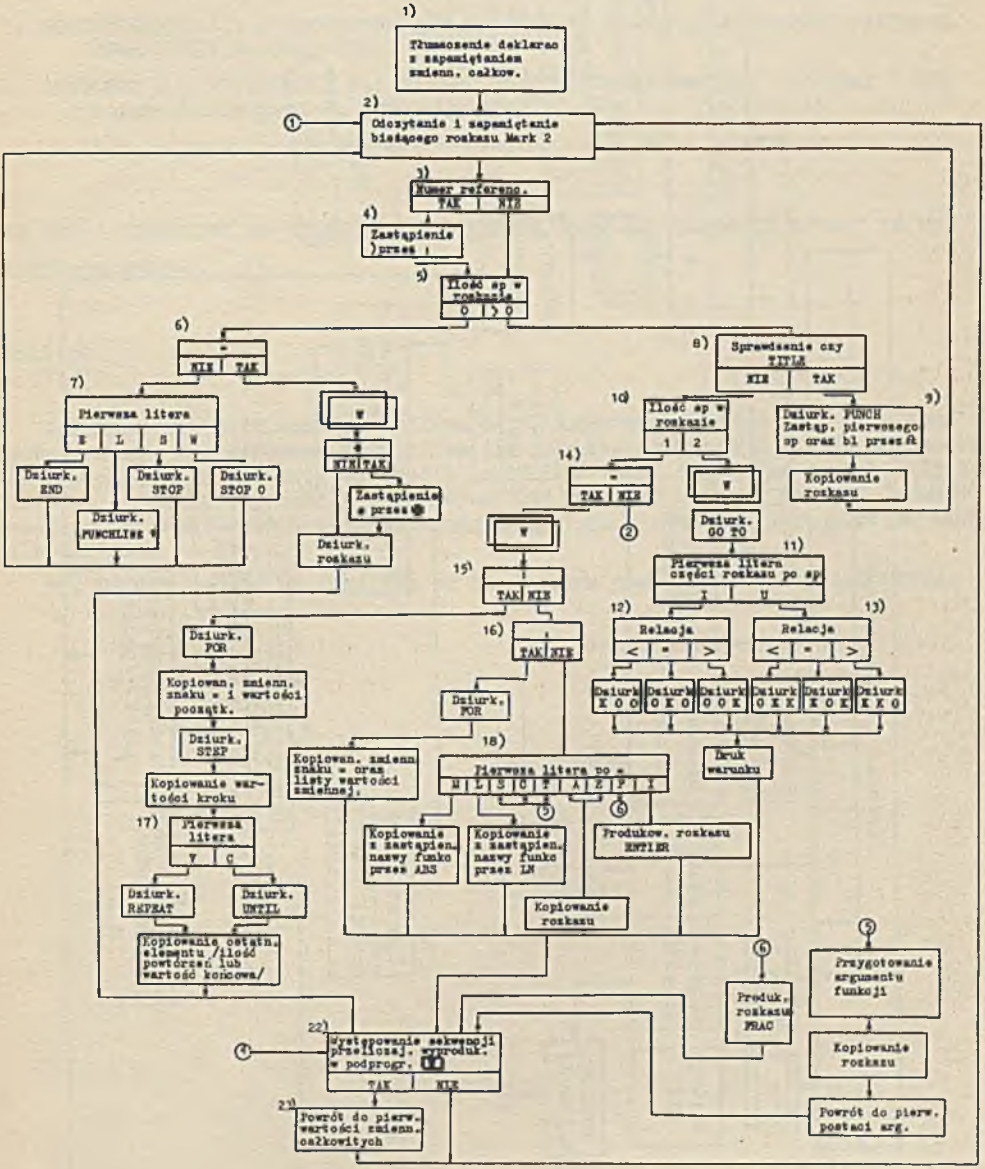
Autorzy zwracają uwagę, że umieszczony w artykule schemat blokowy translatora nie jest szczegółowym schematem dla programisty

opracowującego translator. Celem autorów było jedynie pokazać graficznie główne problemy związane z automatyzacją procesu tłumaczenia.

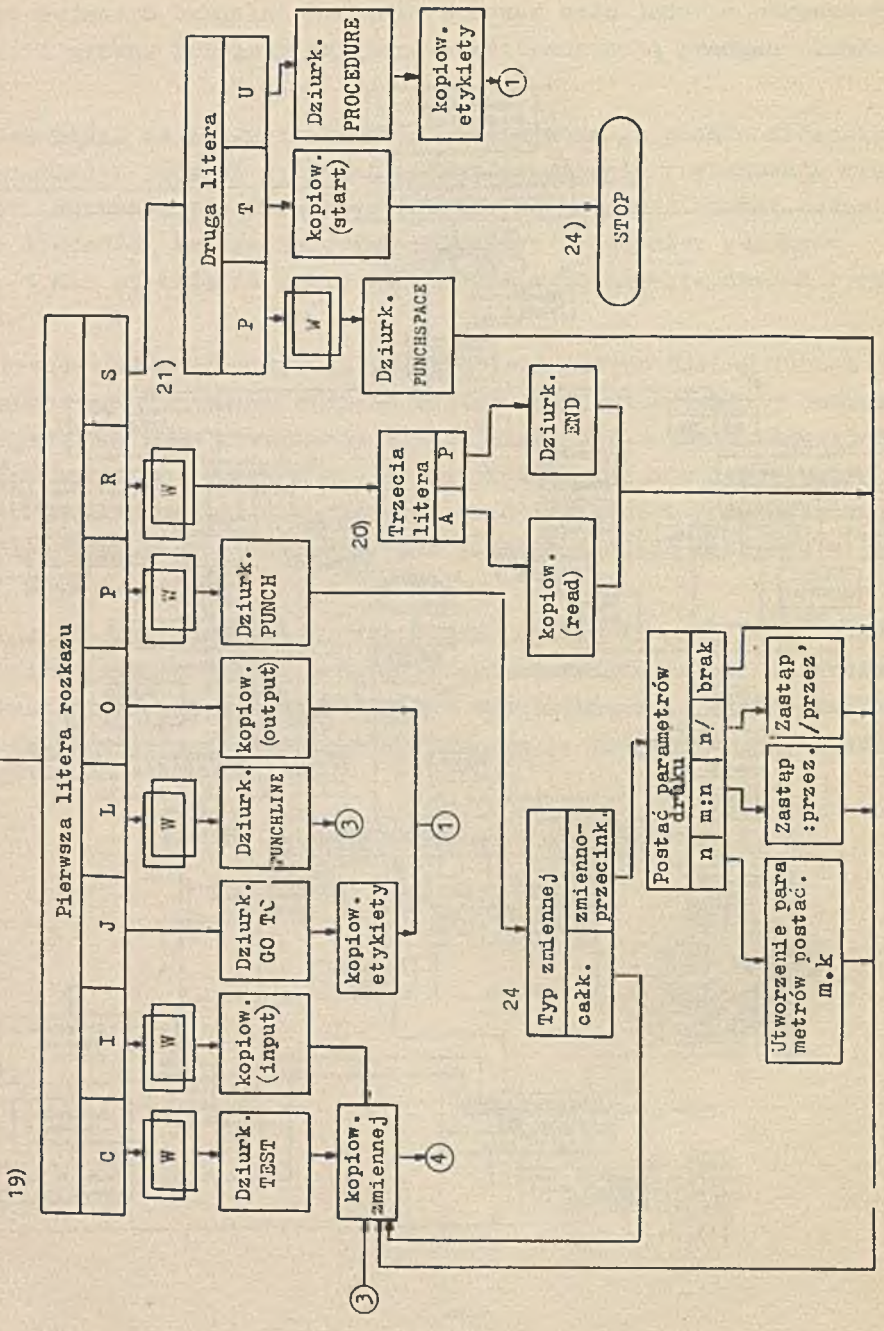
Jak widać ze schematu translatora, drobne /z punktu widzenia programisty/ różnice w treści rozkazów znacznie rozbudowują translator. Wprowadzając dodatkowe rozkazy do autokodu Most-1 można by było trudności te wyeliminować. Niestety translator autokodu Most-1 nie posiada na razie możliwości wprowadzenia nowych rozkazów.

Przedstawione w artykule porównanie autokodu Mark-2 i Most-1 wskazuje na nieznaczne różnice między tymi autokodami. W zasadzie polegają one na wprowadzeniu nowych nazw /też w języku angielskim/. Biorąc m.in. pod uwagę fakt, że autokod Mark-2 nie jest opatentowany, autorzy wyrażają wątpliwość co do celowości wprowadzenia w eksploatowanych w kraju maszynach jeszcze jednego autokodu w postaci Most-1.

Pamiętając o konsekwencjach, jakie niesie ze sobą każdy nowy autokod pożądane jest aby decyzje opracowywania nowych autokodów i translatorów były podejmowane po przeprowadzeniu bardzo wnikliwej analizy i po przedyskutowaniu w gronie przyszłych użytkowników.


1)
Tłumaczenie deklarac z zapamiętaniem zmienn. całkow.
2)
Odczytanie i zapamiętanie bieżącego rozkazu Mark 2
3)
Numer referenc.
TAK
NIE
4)
Zastąpienie przez
5)
Ilość sp w rozkazie
0
> 0
6)
=
NIE
TAK
7)
Pierwsza litera
E
L
S
W
Dziurk. END
Dziurk. STOP
Dziurk. STOP 0
Dziurk. PUNCHLIST 9
W
NIE
TAK
Zastąpienie przez
Dziurk. rozkazu
8)
Sprawdzenie czy TITLE
NIE
TAK
9)
Dziurk. PUNCH Zastąp. pierwszego sp oraz bl przez ft
Kopiowanie rozkazu
10)
Ilość sp w rozkazie
1
2
14)
=
TAK
NIE
W
Dziurk. GO TO
11)
Pierwsza litera części rozkazu po sp
I
U
12)
Relacja
<
=
>
13)
Relacja
<
=
>
Dziurk K O O
Dziurk O K O
Dziurk O O K
Dziurk O K K
Dziurk K O K
Dziurk K K O
15)
W
TAK
NIE
16)
,
TAK
NIE
Dziurk. FOR
Kopiowan. zmienn. znaku = oraz wartości początk.
Dziurk. STEP
Kopiowanie wartości kroku
17)
Pierwsza litera
W
C
Dziurk. REPEAT
Dziurk. UNTIL
Kopiowanie ostatn. elementu /ilość powtórzeń lub wartość końcowa/
Dziurk. FOR
18)
Pierwsza litera po =
N
L
S
C
T
A
Z
F
I
Kopiowan. zmienn. znaku = oraz listy wartości zmiennej.
Kopiowanie z zastąpien. nazwy funkc przez ABS
Kopiowanie z zastąpien. nazwy funkc przez LN
Druk warunku
Produkow. rozkazu ENTIER
Kopiowanie rozkazu
Produk. rozkazu FRAC
Przygotowanie argumentu funkcji
Kopiowanie rozkazu
Powrót do pierw. postaci arg.
22)
Występowanie sekwencji przełącz. wyprodukuj. w podprogr.
TAK
NIE
23)
Powrót do pierw. wartości zmienn. całkowitych

Rys. 3a


19)
Pierwsza litera rozkazu
C
I
J
L
O
P
R
S
W
Dziurk. TEST
kopiow. (input)
Dziurk. GO TO
Dziurk. PUNCHLINE
kopiow. (output)
Dziurk. PUNCH
kopiow. (read)
kopiow. etykiety
kopiow. zmiennej
21)
Druga litera
P
T
U
Dziurk. PROCEDURE
kopiow. (start)
kopiow. etykiety
Dziurk. PUNCHSPACE
24)
STOP
20)
Trzecia litera
A
P
Dziurk. END
24
Typ zmiennej
całk.
zmienno-przecink.
Postać parametrów druku
n
m:n
n/
brak
Zastąp :przez.
Zastąp /przez/
Utworzenie para metrów postać. m.k

Ryc. 3b

Literatura

1. SZCZEPKOWICZ J.: Programowanie w autokodzie Most-1 dla maszyny cyfrowej ODRA 1003, Wrocław 1964.
2. JAWORSKI W., PASIKOWSKI A.: Zasady programowania maszyny cyfrowej E 803 w autokodzie Mark-2, Warszawa 1962.

ON THE POSSIBILITY OF TRANSLATING PROGRAMS FROM THE AUTOCODE MARK-2 TO THE AUTOCODE MOST-1

Summary

The paper is destined for employers of computing centres and their computer users. The autocode Mark-2 for the digital computer E 803 is presented as compared with the autocode Most-1 used in ODRA 1003. The possibility to develop a relatively simple translator for programs written in Mark-2 autocode to be translated to the autocode Most-1 is proved. A flow-chart of such a translator is given.

The reader should be familiar at least with one of the described codes.

Instytut Maszyn Matematycznych
Algorytmy N° 5
© 1965.12

OPIS JĘZYKA I TRANSLATORA ALGUM *) DLA MASZYN UMC

Jerzy LESZCZYŃSKI

Pracę złożono 10.01.1965 r.

Praca zawiera opis języka ALGUM specjalnie opracowanego dla bardzo małych maszyn matematycznych, krótki opis translatora tego języka wykonanego dla maszyn UMC oraz podsumowanie efektów uzyskanych w ponad rocznej jego intensywnej eksploatacji w Ośrodku Obliczeniowym.

1. WSTĘP

Praca niniejsza podaje opis języka ALGUM opracowanego dla maszyn UMC w Pracowni Maszyn Matematycznych Centralnego Ośrodka Badań i Rozwoju Techniki Kolejnictwa w Warszawie, oraz ogólny opis translatora tego języka na kod 20 W, używany powszechnie w maszynach UMC.

Praca opiera się na wynikach uzyskanych w czasie trwającej około jednego roku eksploatacji maszyny UMC korzystającej z translatora ALGUM. Podana wersja jest ostatnią wersją języka ALGUM.

2. OGÓLNE CECHY JĘZYKA

Mała szybkość maszyny UMC i ograniczona pojemność jej pamięci narzuciły warunek minimalizacji ilości rodzajów wyrażeń języka. Z

*) Nazwa ALGUM powstała ze skrótu wyrażenia "język algorytmiczny dla maszyn UMC".

tego powodu przyjęta została zasada pisania formuł w postaci:

Lewa strona formuły	:	prawa strona formuły

Lewa strona formuły może być tylko wyrażeniem arytmetycznym, zawierającym /rozumiane konwencjonalnie/ operacje dodawania, odejmowania, mnożenia i dzielenia w dowolnym układzie, wiążące symbole zmiennych prostych, zmiennych indeksowanych i podprogramów.

Prawa strona formuły może być wskazaniem miejsca odesłania wyniku, badaniem warunku, nazwą podprogramu lub jednocześnie szeregiem ww wyrażeń.

Przy powyższym ujęciu nie jest konieczne wprowadzanie takich wyrażeń jak: GDY, JESLI, TO, PRZECIWNIE itp. niezbędnych w zapisie badania warunkowego przy konwencjonalnej postaci formuły. Jednoczesne przyjęcie zasady identyfikacji formuły przez pojawienie się symbolu ":" pozwoliło na przyśpieszenie procesu translacji oraz na likwidację wszelkich ograniczeń w zapisie nazw zmiennych i podprogramów.

Deklaracje przyporządkowują określone miejsca w pamięci maszyny /adresy/ określonym symbolom, przy czym jeżeli deklarowana jest nazwa bloku /zmienna indeksowana/ lub nazwa podprogramu, adres określa miejsce pierwszej pozycji bloku lub pierwszego rozkazu podprogramu.

Symbole raz zadeklarowane zachowują swoje znaczenie do końca programu niezależnie od miejsca, w którym występują /w programie czy podprogramach/. W szczególnych przypadkach jedna wartość może być oznaczona dwoma lub więcej symbolami.

Podprogramy pisane w języku ALGUM rozpoczynają się od zdania DALEJ) < nazwa podprogramu > . i kończą się zdaniem WROC . Podprogramy te muszą być umieszczane zawsze przed programem /przed zdaniem DALEJ) PROGRAM./, przy czym podprogram wewnętrzny /wyższego rzędu/ jest zawsze pisany przed podprogramem zewnętrznym /niższego rzędu/. Komunikacja z podprogramami pisanymi w języku

ALGUM odbywa się przy wykorzystaniu symboli zmiennych deklarowanych w podprogramach, a wykorzystywanych w podprogramach i programie.

W formuły mogą być wpisywane wyłącznie symbole podprogramów jednoargumentowych zapisanych w kodzie 20 W. Argument podprogramu pisany w nawiasach może być prostym wyrażeniem arytmetycznym.

Komunikacja z wieloargumentowymi podprogramami pisanymi w kodzie 20 W realizowana jest w ten sam sposób jak z podprogramami pisanymi w języku ALGUM z tym, że symbole zmiennych deklarowane są w programie.

Zmienne indeksowane uzależnione są tylko od jednego indeksu,który może być prostym wyrażeniem arytmetycznym, tj. wyrażeniem zawierającym wyłącznie operacje dodawania i odejmowania.

Wszystkie badania przeprowadza się względem "zera", przy czym mogą być kolejno wykonane wszystkie cztery badania, tj. > 0, < 0, $= 0$, $\neq 0$, rozpatrywanego argumentu.

Nie przewiduje się automatycznej organizacji cyklu.

Poniżej podano listę wyrażeń języka:

POCZATEK. - początek tekstu programu

KONIEC. - koniec tekstu programu

DALEJ) alfa. - początek procedury "alfa"

WROC. - koniec procedury

DALEJ) PROGRAM. - początek procedury "program" /procedury "głównej"/

ZMIENNA) a 100. - przyporządkowanie symbolowi "a" znaczenia "zmienna" oraz miejsca "100" w pamięci maszyny

PODPROGRAM) beta 1000. - przyporządkowanie symbolowi "beta" znaczenia "procedura" oraz określenie miejsca w pamięci.

\+ - dodawanie arytmetyczne

\- - odejmowanie arytmetyczne

. – mnożenie w skali

' – mnożenie liczb całkowitych

/ – dzielenie w skali z zaokrągleniem

: – rozdzielenie "lewej" i "prawej" strony formuły

= – odesłanie wyniku wg "lewej" strony formuły na miejsce wg argumentu umieszczonego za znakiem "="

WZERA 11. – przejście wg etykiety 11 dla wyniku wg "lewej" strony formuły większego od zera

MZERA 11. – jak wyżej ale dla przypadku "mniejszy od zera"

NZERO 11. – jak wyżej dla przypadku "niezero"

ZERO 11. – jak wyżej dla przypadku "zero".

Pozostałe wyrażenia, takie jak KOMENTARZ), TEKST), STOP, SKOCZ 11, LINIA, SPACJA, DRUKUJ nie wymagają wyjaśnienia ich znaczenia, a sposób ich używania określa podana w p-kcie 3 składania języka.

3. SKŁADNIA JĘZYKA

W opisie składni używane są symbole metajęzyka: < > | :: = Znaki dalekopisowe pisane są dużymi literami.

Formalny opis składni języka ilustrowany jest przykładami.

< cyfra > :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

< litera > :: = A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | X | Z

< separator > :: = + | - | . | / |) | (| ' | : | =

< znak dalekopisowy > :: = < cyfra > | < litera > | < separator > | , | SP | CR | LF | LI | CY

< liczba > :: = < cyfra > | < liczba > < cyfra >

< liczba w skali > :: = < liczba > , < liczba > | , < liczba > | < liczba > ,

Przykład: 12 154 9998

,12 154, 99,98 00,12400

< operator prosty > :: = + | -

< operator arytmetyczny > :: = < operator prosty > | . | /

< operator odniesienia > :: = WZERA < liczba > | MZERA < liczba > | NZERO < liczba > | ZERO < liczba >

< operator > :: = < operator arytmetyczny > | < operator odniesienia >

< napis > :: = < litera > | < napis > < litera >

< nazwa > :: = < napis > | < napis > < liczba > | < liczba > | < liczba w skali >

Przykład: A A1 A22 134

ALFA ALFA 44 PRZYKŁAD 1,120

< formuła prosta > ::= < nazwa > | < nazwa > < operator prosty > < nazwa > | < formuła prosta > < operator prosty > < nazwa >

Przykład: A + A12 - 1,120

ALFA - 2 + DELTA 3

DELTA 2

1 - A + BETA 12 - K2

< nazwa złożona > ::= < napis > (< formuła prosta >)

Przykład: PWK (A + A12 - 1,120)

X (DELTA 2)

BETA (ALFA - 2 + DELTA 3)

< operand > ::= < nazwa > | < nazwa złożona >

< formuła > ::= < lewa strona form > : < prawa strona form >

< lewa strona form > ::= < operand > | < operand > < operator

arytmet > < operand > |
< lewa strona form > < operator arytmet > < operand >

Przykład: B + A12.PWK (A + 12,1) /5,01. K2
H2 + 3,. K
,33 . H2 - K

< prawa strona form > ::= < operand > .| < operator odniesienia >.|
< prawa strona form > = < operand > .|
< prawa strona form > < operator odniesienia >.|
< prawa strona form > DRUKUJ < liczba > .|DRUKUJ.

Przykład formuły: B + A12 := K2.WZERA 33.
B + ALFA (K + 1) := K(H + 1) = P.ZERO 23.
B/ALFA - K2:WZERA 2.ZERO 4.=K.MZERA 23.
B + 1 := B.

< etykieta > ::= < liczba >)

Przykład: 1) 22) 99) 04)

< deklaracja > ::= ZMIENNA) < nazwa > .| PODPROGRAM) < nazwa >.|
< deklaracja > < nazwa > .

Przykład: ZMIENNA) A 100. BETA 1200. 300.
PODPROGRAM) KAPPA 1100. PWK 54.

< zdanie > ::= < formuła > CR |< deklaracja > CR |
POCZATEK. CR | KONIEC. CR | DALEJ) < napis > . CR|
WROC. CR | TEKST) CR < ciąg znaków niezawierają-
cy CR > CR|
KOMENTARZ) < ciąg znaków nie zawierający CR > CR|
STOP.CR | SKOCZ < liczba > . CR | DALEJ) PROGRAM. CR|
LINIA. CR | SPACJA < liczba > . CR|
< etykieta > < zdanie >

< program > ::= < zdanie > | < program > < zdanie >

Przykład budowy programu:

```
                POCZATEK.
                DALEJ) ALFA.
```

deklaracja i zdania procedury alfa

```
                WROC.
                DALEJ) PROGRAM.
```

deklaracja i zdanie programu

```
                KONIEC.
```

4. OPIS TRANSLATORA

Translator współpracuje z programem 20 W.

Translator zajmuje 768 miejsc /od poz. - 1365 do poz. - 598/, tj. całą wolną ujemną część pamięci.

Do wprowadzania liczb, przeliczania skal, wprowadzania podprogramów bibliotecznych, podprogramów funkcji elementarnych i drukowania wyników wykorzystuje się odpowiednie podprogramy programu 20 W.

Translacja programu napisanego w języku ALGUM odbywa się zdaniami i stąd konieczne jest korzystanie z czytnika START - STOP przy wprowadzaniu programu.

W związku z ograniczoną pojemnością pamięci maszyny, na zapis programu w języku ALGUM nakłada się następujące ograniczenia wynikające z techniki translacji:

a. maksymalna długość programu - 2 000 rozkazów w kodzie 20 W,
b. maksymalna ilość symboli zmiennych deklarowanych w programie - 99 symboli,
c. maksymalna ilość symboli podprogramów deklarowanych w programie - 27 symboli, z tym że trzy symbole nie wymagają deklarowania, mianowicie ABS - wartość absolutna, PWK - pierwiastek kwadratowy, D KROTKIE - drukowanie krótkie /ww podprogramy umieszczone są w translatorze/,

d. identyfikowane są tylko oztery pierwsze litery napisu,
e. w formuły mogą być wpisywane wyłącznie liczby dodatnie, przy czym ilość miejsc po przecinku jest stała i związana ze skalą przyjętą w liczeniu,
f. symbolami etykiet mogą być wyłącznie liczby jedno lub dwucyfrowe,
g. wykluoza się możność użycia symbolu literowego "0000" jako deklarowanego symbolu zmiennej lub podprogramu. Symbolu tego można używać jako formalnego wypełnienia lewej strony formuły /np. przy wywoływaniu podprogramów/,
h. poza programem, który w pamięci maszyny rozpoczyna się zawsze od pozycji 400 sięgając odpowiednio daleko w górę, od pozycji 171 umieszczane są kolejno parametry liczbowe występujące w programie, oraz około 10-ciu pozycji poczynając od 150 wykorzystanych jako komórki robocze programu. Wszystkie pozostałe miejsca dodatniej części pamięci maszyny /poza zajętymi przez program 20 W/ mogą być wykorzystywane na umieszczanie danych, podprogramów i wyników obliczeń. Również ujemna część pamięci maszyny może być wykorzystana do umieszczenia danych, podprogramów i wyników obliczeń przez wykorzystanie po przeprowadzonej translacji obszaru zajmowanego przez translator, przy czym należy pozostawić niewielką część translatora, w której mieszczą się stałe podprogramy wykorzystywane przez program /miejsca od -650 do -598/,
i. w czasie translacji poza miejscami wymienionymi w p-kcie h niszczona jest zawartość pamięci w obszarze pomiędzy 2400 - 2730,
j. w czasie translacji badana jest poprawność formalna tekstu programu; w przypadku znalezienia błędu drukowany jest napis "bd" i translacja zostaje zatrzymana. Identyfikację sygnalizowanego miejsca ułatwia lista adresów etykiet drukowana w czasie translacji,
k. w czasie wykonywania programu nadmiar badany jest tylko przy wykonywaniu operacji mnożenia i dzielenia. W przypadku stwierdzenia nadmiaru zostaje wydrukowany adres miejsca w programie, gdzie wystąpił nadmiar oraz napis "bd", po czym wykonanie programu zostaje zatrzymane.

Czas translacji programu o długości 1000 rozkazów w kodzie 20 W wynosi około 20 min.

Sposób wykorzystania pamięci przy korzystaniu z translatora umożliwia blokowanie po translacji grup ścieżek /po 8 ścieżek/ pamięci, na których zapisany jest program i parametry.

Program dostosowany jest do pracy na ścieżkach zablokowanych.

Powyższe odnosi się również do programu translatora, który w zasadzie pracuje na zablokowanej części pamięci.

5. UWAGI KOŃCOWE

Przeszło roczny okres eksploatacji opisanego wyżej translatora dla języka ALGUM wykazał, że efektywność pracy przeciętnego programisty wykonującego małopowtarzalne zadania obliczeniowe wzrosła 6 - 10 razy /zadania o problematyce inżynierskiej wyrażające się programami 800 - 1600 rozkazów w kodzie 20 W/.

Czas poświęcony na uruchomienie programu zmalał około 10 razy, a o 30% wzrósł czas poświęcony na właściwe obliczenia.

Program ułożony przez translator jest o 10 - 30% dłuższy od takiego samego programu napisanego przez średnio wykwalifikowanego programistę /translator wykorzystuje w niewielkim stopniu możliwości wynikające z mikroprogramowania, ograniczając się do listy 24 operacji/.

Poza opisanym wyżej translatorem istnieje jego wersja zmiennoprzecinkowa oraz wersja dla podwójnej precyzji z tym, że składnia językowa ALGUM dla tych przypadków uzupełniona jest dwoma dodatkowymi symbolami.

PRZYKŁADY

```
komentarz) tablicowanie wartości wielomianu wg schematu hornera

początek.
dalej)program.
zmienna) x 100. y 101. i 102.    n 110. a 111.

   0:=x.
1) n:=i.
   0:=y.
11)y.x+a(i):=y.
   i-1:=i.wzera 11.zero 11.
linia.
x:drukuj 4.
y:drukuj.
   x+0,05000:=x.
   x-1,00000:mzera 1.zero 1.
stop.

koniec.


komentarz) obliczanie pierwiastkow rzeczywistych wiel st drugiego

poczatek.
dalej)program.
zmienna) a 1000. b 1100. c 1200. delta 70. i 77. n 999.

linia.
tekst)
        pierwiastki rzeczywiste  wielomianow st drugiego
linia.
tekst)
      a         b         c        x1         x2

0:=i.

22)linia.
   a(i):drukuj 2.
   b(i):drukuj 2.
   c(i):drukuj 2.
b(i).b(i)-4,00.a(i).c(i):mzera 11.=pwk.=delta.
0-b(i)/2,00/a(i)-delta/2,00/a(i):drukuj 2.
0-b(i)/2,00/a(i)+delta/2,00/a(i):drukuj.
33)i+1:=i.
i-n:mzera 22.
stop.

11)tekst)
  nie ma pierw rzeczy
skocz 33.

koniec.
```

THE DESCRIPTION OF THE LANGUAGE AND TRANSLATOR ALGUM FOR UMC COMPUTERS

Summary

The paper describes the ALGUM language, developed for very small universal computers. Their storage capacity and operation speed make impossible an effective use of a translator being any reasonable subset of ALGOL.

The main feature of the described language is the form of the formula,the left hand side of which is always an arithmetic expression, and the right hand side - a sequence of organization expressions /such as transfering of results, testing characters or zeros with a conditional jump, printing/.

Due to this feature, the number of basic expressions has been reduced to minimum and full possibility of formulating the program - was kept.

The ALGUM translator, being used for more than one year of UMC computer intensive exploitation, confirmed its purposefulness.

A short description of the translator is given as also the results from its exploitation at the Computing Centre.

R E C E N Z J E

Począwszy od niniejszego numeru ALGORYTMÓW, redakcja będzie starała się zamieszczać recenzje z ciekawszych publikacji zarówno polskich jak i zagranicznych.

Starting with the present No of ALGORITHM the editor's office will endeavor to insert reviews of more interesting Polish and foreign publications.

HUSAIN Tagdir: The open mapping and closed graph theorems in topological vector spaces. Braunschweig, 1965. Friedr. Vieweg et Sohn, ss. 108, poz. bibliogr. 38.

This is a selfcontained booklet devoted to various generalizations of "three of the deepest results of Functional Analysis, namely, the open-mapping and closed-graph theorems, and the so-called Krein-Šmulian theorem". The first two chapters may be recommended as a short summary of basic facts on linear topological spaces /with Bourbaki-style exposition/. The remainder of the book is devoted to the study of these classes of locally convex spaces in which the mentioned above theorems hold. The author studies various notions of completeness suitable for validity of the open-mapping and closed-graph theorems, and so called S-spaces, introduced by the author, suitable for validity of a version of the Krein-Šmulian theorem. The theorem 6 of the chapter 6 "is the most general form, so far, of the Krein-Šmulian theorem". Only the last chapter is devoted to "locally convex spaces with the B/C/ property". It also contains historical notes.

The book is highly specialized, and may be considered as a sort of up-to-date encyclopedy on the problems connected with the mentioned above three theorems, as well as on various types of the concept of completeness of locally convex linear topological spaces, and may be recommended to specialists working on the above topics.

W. ŻELAZKO

W.D.N. Zam.1861/66 Nakład 630 egz.

POLITECHNIKA ŚLĄSKA
BIBLIOTEKA GŁÓWNA

BIBLIOTEKA GŁÓWNA
Politechniki Śląskiej
P 2223/65/66

Druk. „Znak" Zl. 1039/1111 Nakl. 650