ASUPRA PROGRAMĂRII LA MAŞINILE ELECTRONICE DE CALCULE A PRODUSULUI A DOUĂ MATRICI

DE

TEODOR RUS

(Cluj)

Lucrare prezentată la sesiunea ştiinţifică din 7—9 decembrie 1962 *a Academiei R.P.R. — Filiala Cluj*

În multe din problemele economice şi mai ales în acele probleme care se rezolvă cu ajutorul metodei programării liniare, apar operaţii cu matrici de un caracter special. Acest caracter special al matricilor rezidă în faptul că coeficienţii multor necunoscute fiind nuli, distribuţia elementelor nenule în matricile cu care se operează este foarte rară. Pentru problemele cu un număr mare de necunoscute şi un număr mare de relaţii care leagă necunoscutele, nu se pot executa manual calculele, sau dacă se pot, ele necesită o muncă foarte grea şi de lungă durată. De aceea, rezolvarea acestor probleme este util să se programeze la o maşină electronică de calcul.

Programarea însă ne impune restricţia limitării la matrici de un anumit ordin, deoarece memoriile maşinilor electronice au capacităţi de înmagazinare a datelor limitate, şi mai mult, sînt mici în raport cu dimensiunile matricilor care intervin de obicei în problemele economice.

Ţinînd cont însă de faptul că numărul elementelor nenule ale matricilor sînt puţine în general, în raport cu numărul elementelor nule, se pot construi programe care să necesite păstrarea numai a elementelor nenule ale matricilor. Un astfel de program pentru produsul a două matrici a fost dat în lucrarea [1].

În această lucrare vom da o altă metodă de programare a produsului a două matrici, care în anumite ipoteze despre densitatea elementelor nenule ale matricilor şi despre dimensiunile acestora, îmbunătăţeşte metoda din lucrarea citată.

Să considerăm matricile

$$\mathbf{A} = \| a_{ji} \|, \quad \mathbf{B} = \| b_{ik} \| \tag{1}$$

$$j = 1, 2, \ldots, m; \; i = 1, 2, \ldots, n; \; k = 1, 2, \ldots, p.$$

şi produsul lor

$$\mathbf{C} = \mathbf{A} \cdot \mathbf{B} = \| c_{jk} \| \tag{2}$$

unde

$$\begin{aligned} c_{jk} &= \sum_{i=1}^{n} a_{ji}\, b_{ik} \\ j &= 1, 2, \ldots, m \\ k &= 1, 2, \ldots, p. \end{aligned} \tag{3}$$

Pentru realizarea produsului acestor două matrici cu ajutorul unei maşini electronice, pe baza metodei clasice, sînt necesare în afara celulelor ocupate de program, $m \times n + p \times n = n(m + p)$ celule, pentru păstrarea elementelor celor două matrici.

În cele ce urmează vom prezenta algoritmul precum şi schema logică operatorială a programului acesztui algoritm, prin care se realizează produsul matricilor **A** şi **B** după formula (3), înscriind în memoria maşinii electronice la care lucrăm exclusiv elementele diferite de zero.

Pentru aceasta, vom presupune că fiecare linie, respectiv coloană a matricilor **A** şi **B** posedă cel puţin un element diferit de zero. În caz contrar, matricile se reduc la matrici de dimensiuni mai mici, prin eliminarea liniilor, respectiv coloanelor care conţin numai elemente nule.

Să presupunem că matricea **A** are l elemente diferite de zero, iar matricea **B** are r elemente diferite de zero, şi fie

$$a_1, a_2, \ldots, a_l \tag{4}$$

elementele diferite de zero ale matricei **A**, numărate pe linii, în aşa fel ca dacă am epuizat elementele diferite de zero ale liniei j a matricei **A** pînă la elementul a_ν, din şirul (4), elementul $a_{\nu+1}$ din acelaş şir este primul element diferit de zero a liniei $j+1$ a matricei **A**, numărat în ordinea crescătoare a indicelui de coloană a acestei linii, iar

$$b_1, b_2, \ldots, b_r \tag{5}$$

elementele diferite de zero ale matricei **B**, numărate pe coloane în mod analog.

Elementele diferite de zero ale matricei **A** le vom înscrie în memoria maşinii în celulele succesive

$$a + 1, \quad a + 2, \ldots, \quad a + l \tag{6}$$

astfel ca elementul a_ν al şirului (4) să fie înscris în celula $a + \nu$, unde a este codul unei anumite celule a memoriei maşinii, iar elementele diferite de zero ale matricei **B** le vom înscrie în celulele succesive

$$b + 1, \quad b + 2, \ldots b + r \tag{7}$$

astfel ca elementul b_μ din şirul (5) să fie înscris în celula $b + \mu$, unde b este codul unei anumite celule a memoriei maşinii.

În şirul (4) un element oarecare $a_\nu = a_{ji}$, unde j şi i sînt indicii săi de linie, respectiv de coloană. Vom nota cu $i_\nu = i(a_\nu)$ indicele de coloană al unui element oarecare a_ν din şirul (4). În mod analog în şirul (5), dacă $b_\mu = b_{ik}$, atunci $i_\mu = i(b_\mu)$ este indicele de linie al elementului b_μ.

Deoarece

$$\begin{aligned} 1 \leqq i(a_\nu) \leqq n \\ 1 \leqq i(b_\mu) \leqq n, \end{aligned} \tag{8}$$

rezultă uşor că numărul maxim de cifre binare al numerelor $i(a_\nu)$ şi $i(b_\mu)$ scrise în sistem binar de numeraţii nu depăşeşte numărul $1 + [\log_2 n]$.

Fie acum şirul indicelor de coloană al elementelor diferite de zero ale matricei **A**

$$i(a_1), i(a_2), \ldots i(a_\nu), \ldots i(a_l) \tag{9}$$

şi şirul analog al indicilor de linie pentru matricea **B**

$$i(b_1, i(b_2), \ldots i(b_\mu), \ldots i(b_r). \tag{10}$$

Să considerăm şirurile de numere binare corespunzătoare şirurilor (9) şi (10), obţinute prin transformarea acestora din urmă în sistem binar de numeraţie:

$$\alpha\ , \alpha_2, \ldots, \alpha_\nu, \ldots, \alpha_l \tag{9'}$$

şi

$$\gamma_1, \gamma_2, \ldots, \gamma_\mu, \ldots, \gamma_r, \tag{10'}$$

Se ştie că şirul de numere (9) determină coloana în care se găseşte oricare element al şirului (4) în matricea **A**, iar şirul (10) determină linia în care se găseşte oricare element al şirului (5) în matricea **B**. Pentru a fi însă complet determinat locul unui element în matricea **A**, respectiv **B**, trebuie să mai ataşăm şirurilor (9) şi (10) încă cîte un şir de indici, care ne vor arăta linia, respectiv coloana diferitelor elemente ale şirurilor (4) şi (5) în matricea **A**, respectiv **B**. Pentru aceasta, vom ataşa şirurilor de numere binare (9') şi (10') şirurile

$$\alpha_1\beta_\alpha^1, \alpha_2\beta_\alpha^2, \ldots, \alpha_\nu\beta_\alpha^\nu, \ldots, \alpha_l\beta_\alpha^l \tag{9''}$$

şi respectiv

$$\gamma_1\beta_\gamma^1, \gamma_2\beta_\gamma^2, \ldots, \gamma_\mu\beta_\gamma^\mu, \ldots, \gamma_r\beta_\gamma^r \tag{10''}$$

unde

$$\beta_\alpha^\nu = \begin{cases} 1, & \text{dacă după } a_\nu \text{ mai urmează şi alte elemente diferite de zero în aceeaşi linie} \\ 0, & \text{în caz contrar} \end{cases}$$

iar

$$\beta_{\gamma}^{\mu} = \begin{cases} 1, & \text{dacă după } b_{\mu} \text{ mai urmează și alte elemente diferite de zero} \\ & \text{în aceeași coloană} \\ 0, & \text{în caz contrar.} \end{cases}$$

β_{α}^{ν} și β_{γ}^{μ} se numesc indicatoare.

Se poate verifica ușor în ipoteza noastră (nu avem linii sau coloane complet nule) că numărul indicatoarelor β_{α}^{ν} nule este egal cu numărul liniilor matricei **A**, primul indicator β_{α}^{ν} nul corespunzînd primei linii, al doilea la a doua linie ș.a.m.d., iar numărul indicatoarelor β_{γ}^{μ} este egal cu numărul coloanelor matricei **B**, primul indicator β_{γ}^{μ} nul corespunzînd primei coloane a matricei **B**, al doilea la a doua coloană ș.a.m.d. În acest fel șirurile (9'') și (10'') determină complet poziția elementelor șirurilor (4) și (5) în matricile **A**, respectiv **B**.

Din această cauză, șirurile de celule ale memoriei mașinii electronice (6) care conține elementele diferite de zero ale matricei **A** îi punem în corespondență un șir de celule ale memoriei mașinii, care conține pe ordine elementele șirului (9''), iar șirului de celule (7) care conține elementele diferite de zero ale matricei **B** îi punem în corespondență un șir de celule ale memoriei mașinii care conține pe ordine elementele șirului (10''). Se vede că dacă H este numărul ordinelor unei celule a memoriei mașinii, într-o celulă vom introduce $K = \left[\frac{H}{2 + [\log_2 n]}\right]$ numere binare din șirul (9''), respectiv (10'') și prin urmare șirul de celule corespunzător șirului (9'') va fi

$$d + 1, d + 2, \ldots, d + s, \tag{11}$$

iar șirul de celule corespunzător șirului (10'') va fi

$$h + 1, h + 2, \ldots, h + t, \tag{12}$$

unde

$$s = \left[\frac{l}{K}\right] + 1,$$

iar

$$t = \left[\frac{r}{K}\right] + 1.$$

Pentru a executa produsul matricei **A** și **B** după formula (3), vom determina cu ajutorul șirurilor (11) și (12) dacă elementului a_{ν} cu indicele de coloană $i(a_{\nu})$ și linia j (aceasta se stabilește cu ajutorul indicatoarelor β_{α}^{ν}) îi corespunde în matricea **B** un element b_{μ} cu indicele de linie $i(b_{\mu})$ și coloana k (aceasta se stabilește cu ajutorul indicatoarelor β_{γ}^{μ}) astfel ca $i(a_{\nu}) = i(b_{\mu})$.

Dacă asemenea perechi de elemente a_{γ} și b_{μ} există, se face produsul lor și se înscrie într-o celulă $c + p$. De asemenea, numerele j și k determinate pentru această pereche de elemente a_{ν} și b_{μ} se înscriu pe ordine într-o celulă $c' + p$, astfel ca primele $\frac{H}{2}$ ordine ale celulei $c' + p$ să fie ocupate de j iar următoarele de k.

Dacă asemenea pereche de elemente a_{ν} și b_{μ} nu există, se trece mai departe, adică se caută asemenea perechi de elemente pentru alți indici de linie, respectiv coloană.

Schema logică operatorială a programului care realizează algoritmul descris mai sus, este formată din trei blocuri, A, B și C.

Blocul A separă o porțiune dintr-o celulă a șirului (11), respectiv (12), care conțin termenii șirurilor (9''), respectiv (10''). pentru a putea determina dacă ei sînt egali sau nu, și la fel separă ordinele indicatoarelor β_{α}^{ν} și β_{γ}^{μ} corespunzătoare.

Dacă $\sigma_1\ \sigma_2$, σ_2', δ_1, δ_2, δ_2' sînt celule operative și dacă N_q și N_q' sînt două numere ajutătoare, astfel ca

$$N_q = \underbrace{00\ldots0}_{2+[\log_2 n]}\ \ldots\ \underbrace{11\ldots10}_{2+[\log_2 n]}\ \underbrace{00\ldots0}_{2+[\log_2 n]}\ \ldots\ \underbrace{00\ldots0}_{2+[\log_2 n]}\ \underbrace{0\ldots0}_{\chi}$$

$$\overset{1}{\qquad}\qquad\overset{q}{\qquad}\qquad\overset{q+1}{\qquad}\qquad\overset{K}{\qquad}$$

$$N_q^1 = \underbrace{00\ldots0}_{2+[\log_2 n]}\ \ldots\ \underbrace{00\ldots01}_{2+[\log_2 n]}\ \underbrace{00\ldots0}_{2+[\log_2 n]}\ \ldots\ \underbrace{00\ldots0}_{2+[\log_2 n]}\ \underbrace{0\ldots0}_{\chi}$$

unde

$$K = \left[\frac{H}{2 + [\log_2 n]}\right],\ \chi = H - K(2 + [\log_2 n])$$

schema logică operatorială a programului care realizează algoritmul descris de blocul A este de tipul:

$$\overset{a''}{\downarrow}[d + k_1 \to \sigma_1]\overset{a_2}{\downarrow}[\sigma_1 \to \sigma_2 \otimes N_q]\,D_{+}^{H-qk'+1+\chi}(\sigma_2)[\sigma \to \sigma_2' \otimes N_q'] \times$$

$$\times\ D_{+}^{H-qk'+\chi}(\sigma_2')A_1\overset{a_1'}{\downarrow}[h + k_2 \to \tau_1]\overset{a_1}{\downarrow}[\tau_1 \to \tau_2 \otimes N_q]\,D_{+}^{H-qk'+1+\chi}(\tau_2) \times$$

$$\times\ [\tau_1 \to \tau_2' \otimes N_q']\,D_{+}^{H-qk'+\chi}(\tau_2')A_2\overset{b}{\uparrow}$$

valoarea inițială a indicilor k_1 și k_2 fiind egală cu 1.

Am notat prin $\sigma_1 \to \sigma_2 \otimes N_q$ operatorul prin care $<\sigma_1>$ trece a în (σ_2) înmulțit logic cifră cu cifră, cu numărul N_q. Prin $D_{+}^{H-qk'+1+\chi}(\sigma_2)$ am no-

tat operatorul care face ca conținutul celulei (σ_2) să se deplaseze la dreapta cu un număr de $H - qk' + 1 + \chi$ ordine, unde $k' = 2 + [\log_2 n]$.

Numărul q este dat de porțiunea de celulă pe care o separăm. Pentru $q = 1$, avem de-a face cu prima parte a celulei $d + k_1$, respectiv a celulei $h + k_2$ ș.a.m.d. Valoarea maximă a lui q poate fi K.

Operatorul A_1 adună o unitate la o celulă fixă P_1 pentru a determina valoarea lui q pentru șirul de celule (11). În mod analog, operatorul A_2 adună o unitate la o celulă fixă P_2 pentru a determina valoarea lui q pentru șirul de celule (12).

Blocul A predă conducerea calculului operatorului care are la stînga săgeata $\overset{b}{\downarrow}$ din blocul B. Aceasta realizează compararea elementelor corespunzătoare separate, a șirurilor (9'') și (10'') și execută produsul elementelor a_ν și b_μ corespunzătoare, dacă este cazul. Schema logică operatorială a programului algoritmului descris de blocul B este

$$\overset{b}{\downarrow} p(<\sigma_2> \neq <\tau_2>) \overset{10}{\uparrow} \overset{11}{\downarrow} p(<\sigma_2'> \neq 0) \overset{8}{\uparrow} \overset{9}{\downarrow} p(<\tau_2'> \neq 0) \overset{6}{\uparrow} \overset{7}{\downarrow} p(j<l) \overset{4}{\uparrow} \times$$
$$\times p(<P_1 < K) \overset{2}{\uparrow} p(<P_2> < K) \overset{1}{\uparrow} F(j) F_2(q) O \overset{a_1}{\uparrow} \overset{1}{\downarrow} F(k_1) F_2^{-K}(q) \times$$
$$\times F^{-K}(<P_2>) F(j) O \overset{a'}{\uparrow} \overset{2}{\downarrow} F(k_1) F_1^{-K}(q) F^{-K}(<P_1>) p(<P_2> < K) \overset{3}{\uparrow} \times$$
$$\times F(j) F_2(q) O \overset{a''}{\uparrow} \overset{3}{\downarrow} F^{-K}(<P_2>) F(k_2) F_2^{-K}(q) F(j) O \overset{a''}{\uparrow} \overset{4}{\downarrow} \times$$
$$\times p(i < r) \overset{c}{\uparrow} F^{-(l-1)}(j) F_1(q) F_2^{-K}(q) F^{-K}(q) F^{-K}(<P_2>) p(<P_1> < K) \overset{5}{\uparrow} \times$$
$$\times O \overset{a_2}{\uparrow} \overset{5}{\downarrow} F(k_1) O \overset{a''}{\uparrow} \overset{6}{\downarrow} A_1 O \overset{7}{\uparrow} \overset{8}{\downarrow} A_2 O \overset{9}{\uparrow} \overset{10}{\downarrow} A_3 \left[<r_2> \rightarrow (c^1 + p)_0^{\frac{H}{2}} \right] \times$$
$$\times \left[<r_1> \rightarrow (c' + p)_{\frac{H}{2}}^{H} \right] F(p) O \overset{11}{\uparrow},$$

parametrii j, i, p, q, k_1, k_2 inițial fiind egali cu unitatea.

Condițiile logice care intervin în această schemă considerăm că nu necesită o explicație aparte. Operatorul $F_2(q)$ este un operator de readresare, care face ca numerele N_q și N_q' corespunzătoare modificării survenite din cauza trecerii la determinarea termenului următor, al șirului (11''), să devină N_{q+1}, respectiv N_{q+1}'. El revine la a executa o deplasare la dreapta cu un număr de k' cifre a conținutului celulelor care conțin pe $N_q'^d$ și respectiv N_q'. Operatorul $F_2^{-K}(q)$ este de asemenea un operator de readresare, care face ca conținutul celulelor care conțin numerele N_q și N_q' în cazul epuizării unei celule din șirul (12), sau în cazul cînd se termină de parcurs șirul (10'') pentru un termen al șirului (9''), să se deplaseze la stînga cu un număr de Kk' ordine, adică să primim N_1 și N_1' în celulele respective. Operatorii $F_2(q)$ și $F_2^{-K}(q)$ îndeplinesc același rol, referindu-se la șirul de celule (11).

Prin A_1 și A_2 am notat operatorii aritmetici care adună cîte o unitate la celulele (r_1), respectiv (r_2) atunci cînd β_α^ν respectiv β_γ^μ sînt nule, pentru a forma perechea de indici care se atașează elementului produs. Pentru aceasta se presupune însă că, inițial, conținutul celulelor (r_1) și (r_2) este 1.

Operatorul A_3 este aritmetic și execută produsul a două elemente din șirurile (6) și (7) dacă este cazul, înscriind rezultatul într-un șir de celule succesive $c + 1, c + 2, \ldots, c + p$, unde p este numărul elementelor diferite de zero ale produsului executat de operatorul A_3.

Prin $\left[<r_1> \rightarrow (c' + p)_0^{\frac{H}{2}} \right]$ și $\left[<r_2> \rightarrow (c' + p)_{\frac{H}{2}}^{H} \right]$ am notat operatorii care fac ca $<r_1>$ și $<r_2>$ să treacă respectiv primul în primele $\frac{H}{2}$ ordine ale celulei $c' + p$ și al doilea în următoarele $\frac{H}{2}$ ordine ale celulei $c' + p$. În felul acesta indicii de linie, respectiv coloană apar în celulele succesive

$$c' + 1, c' + 2 \ldots, c' + p.$$

Blocul B predă conducerea calculului fie operatorilor din blocul A care au în stînga lor săgețile $\overset{a_1}{\downarrow}, \overset{a'}{\downarrow}, \overset{a_2}{\downarrow}, \overset{a''}{\downarrow}$, fie operatorului care are în stînga sa săgeata $\overset{c}{\downarrow}$ din blocul C.

Deoarece s-ar putea ca în șirul de celule $c' + 1, c' + 2, \ldots, c' + p$ să avem elemente egale, ceea ce corespunde faptului că elementele corespunzătoare din șirul de celule $c + 1, c + 2, \ldots, c + p$ trebuie adunate algebric, și deoarece blocul B nu aranjează într-o ordine determinată elementele matricei produs, blocul C are menirea de a face sumele respective și de a aranja pentru tipărire (sau pentru alte scopuri) elementele matricei produs.

Pentru aceasta, el execută ordonarea șirului de numere

$$<c' + 1>, <c' + 2>, \ldots, <c' + p>$$

astfel ca

$$<c' + \nu_1> < <c' + \nu_2> < \ldots < <c' + \nu>$$

iar

$$<c' + \nu_1> \rightarrow (c' + 1)$$
$$\ldots\ldots\ldots$$
$$<c' + \nu> \rightarrow (c' + p)$$

și în același timp face schimbările și sumele (dacă este cazul) corespunzătoare în șirul

$$<c + 1>, <c + 2>, \ldots, <c + p>$$

astfel ca el să devină succesiv

$$<c+\nu_1>\rightarrow(c+1)$$
$$<c+\nu_2>\rightarrow(c+2)$$
$$\cdots\cdots\cdots$$
$$<c+\nu>\rightarrow(c+p).$$

Schema logică operatorială a programului care realizează acest algoritm* este de forma următoare

$$\overset{c}{\downarrow}\overset{3}{\downarrow}\overset{1}{\downarrow}p(<c'+k>\,<\,<c'+i+1>)\overset{4}{\uparrow}\overset{7}{\downarrow}\overset{5}{\downarrow}p(i+1<p)\overset{2}{\uparrow}\mathrm{F}(i)\,\mathrm{O}\overset{1}{\uparrow}\overset{2}{\downarrow}\times$$
$$\times\,p(k<p-1)\overset{8}{\uparrow}\mathrm{F}(k)\mathrm{F}^{i-k}\quad(i)\,\mathrm{O}\overset{3}{\uparrow}\overset{4}{\downarrow}p(<c'+k>\neq c'+i+1>)\overset{6}{\uparrow}\times$$
$$\times\,[<c'+i+1>\rightarrow(\delta_1)]\times$$
$$\begin{bmatrix}<c'-\nu>\rightarrow(c'+\nu+1)\\ \nu=i,\,i-1,\ldots,i-k+1\end{bmatrix}[<\delta_1>\rightarrow(c'+k)][<c+i+1>\rightarrow(\delta_2)]\times$$
$$\times\begin{bmatrix}<c+\nu>\rightarrow(c+\nu+1)\\ \nu=i,\,i-1,\ldots,i-k+1\end{bmatrix}[<\delta_2>\rightarrow(c+k)]\mathrm{O}\overset{5}{\uparrow}\overset{6}{\downarrow}\Phi(\mathrm{A}_1)\mathrm{A}_1\times$$
$$\times\begin{bmatrix}<c+\mu+1>\rightarrow(c+\mu)\\ \mu=i+1,\,i+2,\ldots,p-1\end{bmatrix}\begin{bmatrix}<c'+\mu+1>\rightarrow(c'+\mu)\\ \mu=i+1,\,i+2,\ldots\,p-1\end{bmatrix}\mathrm{O}\overset{7}{\uparrow}\overset{8}{\downarrow}\mathrm{R}.$$

parametrii i şi k iniţial fiind egali cu unitatea.

Am notat prin $\begin{bmatrix}<c'+\nu>\rightarrow(c'+\nu+1)\\ \nu=i,\,i-1,\ldots,i-k+1\end{bmatrix}$ operatorul care face ca conţinutul celulelor $(c'+i)$, $(c'+i-1)$, $\ldots$ $(c'+i-k+1)$să se deplaseze toate cu un loc la dreapta, adică

$$<c'+i>\rightarrow(c'+i+1)$$
$$<c'+i-1>\rightarrow(c'+i)$$
$$\cdots\cdots\cdots\cdots$$
$$<c'+i-k+1>\rightarrow(c'+i-k)$$

iar prin

$$[<c+\mu+1>\rightarrow(c+\mu)]$$
$$\mu=i+1,\,i+2,\,\ldots,\,p-1$$

* Acest algoritm intervine şi în alte probleme, cum ar fi spre exemplu aranjarea nodurilor unei formule de interpolare a lui Newton pentru ca restul să fie minim, sau pentru aranjarea cuvintelor într-un dicţionar, în lingvistica matematică.

am notat operatorul care face ca conţinutul celulelor $(c+i+2)$, $(c+i+3)$, $\ldots$, $(c+p)$ să se deplaseze toate cu un loc la stînga, adică

$$<c+i+2>\rightarrow(c+i+1)$$
$$<c+i+3>\rightarrow(c+i+2)$$
$$\cdots\cdots\cdots\cdots$$
$$<c+p>\rightarrow(c+p-1).$$

Operatorul $\Phi\,(\mathrm{A}_1)$ este un operator de formare a operatorului aritmetic A_1 care atunci cînd indicii a două elemente sînt egali, adică $<c'+k>=<c'+i+1>$, face suma algebrică a acestor două elemente $<c+k>+<c+i+1>$ şi o înscrie în celula $c+k$. Dacă rezultatul produsului matricilor **A** şi **B** trebuie tipărit, atunci R este operatorul de tipărire a şirului de celule succesive

$$c+1,\,c+2,\,\ldots,\,c+p_1,$$

împreună cu indicii corespunzători care se găsesc în celulele

$$c'+1,\,c'+2,\,\ldots,\,c'+p_1,$$

unde p_1 este numărul elementelor diferite de zero ale matricei produs. În general avem $p_1\leqq p$. Dacă rezultatul produsului este necesar să rămînă în continuare în memoria maşinii, R este un subbloc al blocului C, care face ca şirul indicilor corespunzători elementelor produs, să se transforme într-un şir de forma şirurilor (9''), respectiv (10''), care să fie înscrisîntr-un anumit şir de celule (dintre cele p folosite) astfel ca matricea produs **C** să fie complet determinată şi pasibilă la operaţii ulterioare.

Schema logică operatorială a programului care realizează această transformare este de forma

$$\overset{1}{\downarrow}[c'+k\rightarrow\xi_1][\xi_1\rightarrow\xi_1'\otimes N_1][\xi_1\rightarrow\xi_2'\otimes N_2][c'+k+1\rightarrow\zeta_1]\times$$
$$\times\,[\zeta_1\rightarrow\zeta_1'\otimes N_1][\zeta_1\rightarrow\zeta_2'\otimes N_2]\,\Phi\left(\mathrm{D}_-^{\mu}(\xi_2')\right)[\mathrm{D}_-^{\mu}(\xi_2')\rightarrow f+i]\times$$
$$\times\,p(<\xi_1'>\neq<\zeta_1'>)\overset{4}{\uparrow}[f+i\oplus 0]\overset{5}{\downarrow}p(k<p-1)\overset{6}{\uparrow}p(\lambda<K)\overset{3}{\uparrow}\times$$
$$\times\,\mathrm{F}(\lambda)\mathrm{F}(q)\overset{2}{\downarrow}\mathrm{F}(k)\mathrm{O}\overset{1}{\uparrow}\overset{3}{\downarrow}\mathrm{F}(i)\,\mathrm{F}^{-K+1}(\lambda)\mathrm{F}^{-K}(q)\mathrm{O}\overset{2}{\uparrow}\overset{4}{\downarrow}[f+i\oplus N_q']\times\mathrm{O}\overset{5}{\uparrow}\overset{6}{\downarrow}\mathrm{Op},$$

parametrii i, k, q, λ iniţial fiind egali cu unitatea.
Prin N_1 şi N_2 am notat numerele ajutătoare

$$N_1=\underbrace{11\ldots1}_{\frac{\mathrm{H}}{2}}\ \underbrace{00\ldots0}_{\frac{\mathrm{H}}{2}}$$
$$N_2=\underbrace{00\ldots0}_{\frac{\mathrm{H}}{2}}\ \underbrace{11\ldots1}_{\frac{\mathrm{H}}{2}},$$

prin $\otimes$ operaţia de înmulţire logică, cifră, cu cifră, iar prin $\oplus$ operaţia de adunare logică cifră cu cifră.

Operatorul $\Phi\left(D_{-}^{\mu}(\xi_2')\right)$ este un operator de formare a numărului din celula ξ_2' deplasat la stînga cu un număr de ordine $\mu = H - (2 + [\log_1 n])\lambda$, unde λ inițial este 1, iar maxim este K. Operatorul $[D_{-}^{\mu}(\xi_2') \to f + i]$ transportă conținutul celulei ξ_2' deplasat în ordinele spuse de operatorul D_{-}^{μ} în aceeași ordine ale celulei $f + i$. Se vede simplu că atunci cînd $\lambda = K$, am epuizat o celulă $f + i$ și trebuie să trecem la o celulă următoare. Operatorul $[f + i \oplus N_q']$ adună logic cifră cu cifră numărul N_q' la celula $f + i$, dacă $<\xi_2'> = <\zeta_1'>$, adică dacă într-o linie a matricei produs mai sînt și alte elemente diferite de zero în afară de unul inițial pus în evidență. $F(q)$ și $F^{-K}(q)$ au aceeași semnificație ca și în cazul schemei logico-operatoriale a programului algoritmului descris de blocul B.

În urma executării programului algoritmului descris de blocul R în celulele succesive $f + 1$, $f + 2$, ..., $f + q$, avem înscris pe ordine termenii unui șir de forma

$$\delta_1\beta_\delta^1,\ \delta_2\beta_\delta^2,\ \ldots,\ \delta_{p_1}\beta_\delta^{p_1} \tag{13}$$

unde p_1 este numărul elementelor diferite de zero ale matricei produs **C**, δ_i sînt indicii de coloană a elementelor matricei produs, scriși în sistem binar de numerație, iar

$$\beta_\delta^\rho = \begin{cases} 1, & \text{dacă după } c_\rho \text{ mai urmează și alte elemente diferite} \\ & \text{de zero în aceeași coloană} \\ 0, & \text{în caz contrar.} \end{cases}$$

$$\rho = 1, 2, \ldots, p_1.$$

β_δ^ρ sînt indicatoarele corespunzătoare, c_ρ fiind un element oarecare diferit de zero al matricei produs.

Vom arăta acum printr-un exemplu că această metodă este într-adevăr mai bună decît metoda citată. Să presupunem că dimensiunile matricei **A** sînt 100×60, dimensiunile matricei **B** sînt de asemenea 60×100 și fie $l = r = 100$. Atunci, pentru a determina poziția diferitelor elemente ale șirurilor (4) și (5) în matricile **A** și respectiv **B**, în metoda citată se foloseau două șiruri de celule, referitoare la șirurile (4) și respectiv (5), al căror număr era egal cu $\frac{100 \times 60}{H}$ pentru fiecare matrice. Dacă $H = 30$, atunci în afara celulelor ocupate de program și de datele inițiale se mai foloseau încă 400 celule, pentru cele două șiruri amintite. Cu ajutorul metodei descrisă mai sus, ținînd cont că $[\log_2 100] = 6$, numărul celulelor folosite pentru cele două șiruri (9'') și (10'') de care avem nevoie va fi $2\left(\left[\frac{100}{\frac{30}{8}}\right] + 1\right) = 68$, ceea ce că reprezintă o economie destul de serioasă.

Se poate însă găsi simplu o condiție asupra densității elementelor diferite de zero ale matricilor cu care operăm, pentru ca această metodă să fie mai eficace.

Dacă N este numărul elementelor diferite de zero ale matricei **A**, iar N_1 este numărul elementelor diferite de zero ale matricei **B**, pentru ca metoda descrisă mai sus să fie mai eficace este necesar să fie îndeplinite condițiile :

$$N \leqq \frac{n \cdot m}{2 + [\log_2 n]} \tag{14}$$

$$N_1 \leqq \frac{n \cdot p}{2 + [\log_2 n]}. \tag{14'}$$

Aceste condiții pot fi înglobate într-una singură, introducînd noțiunea de densitate a elementelor diferite de zero ale unei matrici.

Dacă densitatea elementelor diferite de zero ale matricei **A** este numărul subunitar $D_A = \frac{N}{n \cdot m}$, iar densitatea elementelor diferite de zero ale matricei **B** este numărul subunitar $D_B = \frac{N_1}{n \cdot p}$, se poate ușor arăta că condițiile (14) și (14') revin astfel ca numărul $D = \max\{D_A, D_B\}$ să satisfacă condiția

$$D \leqslant \frac{1}{2 + [\log_2 n]}. \tag{15}$$

О ПРОГРАММИРОВАНИИ НА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ ПРОИЗВЕДЕНИЯ ДВУХ МАТРИЦ

КРАТКОЕ СОДЕРЖАНИЕ

В этом труде автор исходит из той предпосылки, что во многих экономических проблемах, применяемые матрицы имеют специальную структуру, состоящую в большом количестве нулей, содержанных.

Поэтому полезно работать только с элементами матриц отличными от нуля, так как объёмы памятей электронных машин малы по сравнению с размерами применяемых матриц. Описывается алгоритм, который осуществляющий произведение двух матриц записывая в памяти машины только отличные от нуля элементы матриц, и даётся операторная логическая схема произведения, осуществляющего этот алгоритм. Эта схема состоит из четырёх блоков A, B, C и R, которые имеют назначения выявлять все пары элементов, произведение которых должно производиться необходимым образом, так как они отличные от нуля и в строках или столбцах, где они находятся, занимают то же место указанное индикаторами β_α^ν и β_γ^μ.

Подчёркиваем, что блок C имеет роль упорядочить по следовательность индексов элементов-произведений, а также и элементы-произведения вместе с их соответствующими индексами так, чтобы были возможны или печатаные в строго определённом порядке, или же переход к последующим действиям, которые выполняются тем же образом.

SUR LA PROGRAMMATION POUR LES MACHINES ÉLECTRONIQUES DE CALCUL DU PRODUIT DE DEUX MATRICES

RÉSUMÉ

Dans ce travail on part de la prémisse que dans nombre de problèmes économiques, les matrices qui interviennent possèdent une structure spéciale, qui relève du grand nombre des zéros qu'elles contiennent.

C'est pourquoi il est utile de travailler uniquement avec les éléments différents de zéro des matrices, parce que les capacités de mémoires de machines électroniques sont petites par rapport aux dimensions des matrices qui interviennent. Ainsi on décrit l'algorithme qui réalise le produit de deux matrices, en inscrivant dans la mémoire de la machine seulement les éléments différents de zéro des matrices; on donne aussi le schéma logique opératoire du produit que réalise cet algorithme. Ce schéma est formé de quatre blocs A, B, C et R qui sont destinés à mettre en évidence toutes les paires d'éléments dont le produit doit être nécessairement exécuté, parce qu'ils sont différents de zéro; dans les lignes ou les colonnes où ils se trouvent ils occupent la même place marquée par les indicateurs β_α^ν et β_γ^ν.

Nous soulignons que le bloc C est destiné à ordonner la suite d'indices des éléments produits, ainsi que les éléments produits et leurs indices correspondants, de telle sorte qu'il soit possible soit de les imprimer dans un ordre nettement précisé, soit de passer à des opérations ultérieures qui s'exécutent de la même façon.

BIBLIOGRAFIE

1. Munteanu E., Rus T. *O metodă de programare a produsului a două matrici.* Studii și cercet. de matem (Cluj), XII, Anexă, 1961–1962 (sub tipar).