

ASUPRA PROGRAMĂRII LA MAȘINILE DE CALCUL A UNOR FORMULE DE INTERPOLARE

DE

B. JANKÓ și T. RUS

(Cluj)

1. În această lucrare ne vom ocupa de programarea la mașinile electronice de calcul a formulelor de interpolare ale lui Lagrange și Newton, precum și de studiul schemelor logice operatoriale ale algoritmilor care realizează calculul după formulele de interpolare ale lui Hermite și Bernstein.

2. Considerăm funcția $f(x)$ dată numai pe nodurile

$$x_1, x_2, \dots, x_{n+1} \quad (1)$$

pe segmentul $[a, b]$, unde

$$a \leq x_1 < x_2 < \dots < x_{n+1} \leq b.$$

Fie $f(x_1), f(x_2), \dots, f(x_{n+1})$ valorile funcției $f(x)$ pe nodurile (1). Se pune problema calculării valorilor aproximative ale funcției $f(x)$ în punctele

$$\xi_1, \xi_2, \dots, \xi_m \in [a, b] \quad (2)$$

cu ajutorul formulei de interpolare a lui Lagrange.

$$L(x_1, x_2, \dots, x_{n+1}; f(x)) = l(x) \sum_{i=1}^{n+1} \frac{f(x_i)}{(x - x_i) l'(x_i)}, \quad (3)$$

unde

$$l(x) = \prod_{k=1}^{n+1} (x - x_k),$$

iar

$$l'(x_i) = \prod_{k=1}^{n+1} (x_i - x_k), \quad k \neq i. \quad (5)$$

Schema logico-operatorială a algoritmului prin care se realizează calcularea valorilor $f(\xi_j)$, $j = 1, 2, \dots, m$, folosind formula (3), este de forma

$$\begin{aligned} & \downarrow \downarrow A_k^1 p(k < n + 1) \uparrow \uparrow F(k) O \uparrow \downarrow F^{-n}(k) \downarrow \downarrow A_2^5 \downarrow p(k \neq i) \uparrow \uparrow \times \\ & \times p(k < n + 1) \uparrow \downarrow F(k) O \uparrow \downarrow A_4 F^{-n}(k) p(i < n + 1) \uparrow \uparrow F(i) \times \\ & \times O \uparrow \downarrow A_5 T(<P>) F^{-n}(k) F^{-n}(i) p(j \geq m) \uparrow \uparrow \text{Stop}. \end{aligned} \quad (6)$$

Semnificația operatorilor și a condițiilor logice care intervin în schema (6) este următoarea:

1) A_1^k este un operator aritmetic, care calculează pentru un ξ_j dat valoarea funcției $l(\xi_j) = \prod_{k=1}^{n+1} (\xi_j - x_k)$ și înscrie rezultatul într-o celulă fixă α .

Condiția logică $p(k < n + 1)$ stabilește dacă produsul $\prod_{k=1}^{n+1} (\xi_j - x_k)$, pentru un j dat, este efectuat pînă la $k = n + 1$.

2) $F(k)$ este un operator de redresare care mărește parametrul k cu o unitate, k fiind inițial 1.

3) $F^{-n}(k)$ este un operator de restabilire care reface parametrul k la valoarea sa inițială.

4) A_2^i este un operator aritmetic care realizează diferența $\xi_j - x_i$, unde j este dat în decursul calculării valorii $f(\xi_j)$ iar i variază de la un termen la altul al sumei din formula (3).

5) Condiția logică $p(i \neq k)$ stabilește dacă i este diferit de k .

6) A_3^k este un operator aritmetic care realizează pentru $i \neq k$ produsul $\prod_{k=1}^{n+1} (x_i - x_k)$ și îl înscrie într-o celulă fixă β .

7) A_4 este un operator aritmetic care calculează termenul i al sumei din formula (3) adică termenul

$$\frac{f(x_i)}{(\xi_j - x_i) \prod_{k=1}^{n+1} (x_i - x_k)}, \quad k \neq i,$$

și-l adună la conținutul celei P care inițial este 0.

8) A_5 este un operator aritmetic care realizează produsul dintre $<\alpha>$ și $<P>$.

9) $T(<P>)$ este un operator de tipărire, care execută tipărirea conținutului celei P pentru un j dat. Tipărirea ar putea fi executată și la sfîrșitul procesului de calcul, ceea ce ar implica ocuparea de către program a încă $m - 1$ celule. În general m fiind destul de mare, aceasta ar duce la dezavantaje din punctul de vedere al folosirii economice a memoriei mașinii.

Formula (3) poate fi pusă și sub forma

$$L(x_1, x_2, \dots, x_{n+1}; f(x)) = \sum_{i=1}^n \frac{l(x) f(x_i)}{(x - x_i) l'(x_i)}. \quad (3')$$

Schemele logice operatoriale ale formulelor (3) și (3') sînt echivalente [6], însă programele corespunzătoare diferă prin faptul că la prima programare se urmărește un număr minim de operații iar la a doua o mai bună stabilitate a calculului referitoare la acumularea erorilor de rotunjire.

Pe baza schemei operatoriale date și descrise mai sus vom întocmi programul pentru calculatorul electronic CIFA-2.

001	01 - ξ_j	03 - x_k
002	04 - $<\alpha>$	11 - $<\alpha>$
003	01 - k	03 - $n + 1$
004	06 - 008	01 - k
005	02 - 1	11 - k
006	01 - 001	02 - N_1
007	11 - 001	10 - 001
008	01 - $<k>$	03 - $<n>$
009	11 - $<k>$	01 - $<\xi_j>$
010	03 - x_i	04 - $<\beta>$
011	11 - $<\beta>$	01 - $<k>$
012	03 - $<i>$	07 - 015
013	01 - $<i>$	03 - $<k>$
014	06 - 017	13 - 000
015	01 - $<x_i>$	03 - $<x_k>$
016	04 - $<\beta>$	11 - $<\beta>$
017	01 - $<k>$	03 - $<n + 1>$
018	06 - 021	01 - $<k>$
019	02 - $<1>$	11 - $<k>$
020	10 - [011] _{II}	13 - 000
021	01 - $<f(x_i)>$	05 - $<\beta>$
022	02 - $<P>$	11 - $<P>$
023	01 - $<k>$	03 - $<n>$
024	11 - $<k>$	01 - $<i>$
025	03 - $<n + 1>$	06 - 030
026	01 - $<i>$	02 - $<1>$
027	11 - $<i>$	01 - 010
028	02 - $<N_2>$	11 - 010
029	10 - [009] _{II}	13 - 000
030	01 - $<\alpha>$	04 - $<P>$
032	16 - 000	01 - $<i>$
032	03 - $<n>$	11 - $<i>$
033	01 - $<k>$	03 - $<n>$
034	11 - $<n>$	01 - $<j>$
035	03 - $<m>$	06 - 046
036	01 - $<j>$	02 - $<1>$
037	11 - $<j>$	01 - 001
038	02 - $<N_2>$	03 - $<N_3>$
039	11 - 001	01 - 009
040	02 - $<N_1>$	11 - 009
041	01 - 010	03 - $<N_4>$
042	11 - 010	01 - 015

043	03 - $\langle N_5 \rangle$	11 - 015
044	01 - 021	03 - $\langle N_4 \rangle$
045	11 - 021	10 - 001
046	00 - 000	

unde prin $[00 n]_{II}$ am notat a doua parte a comenzii $00n$.

Pe lângă cele 46 de celule ocupate de program în cazul mașinii CIFA-2 mai avem nevoie de încă 14 celule operative pentru păstrarea anumitor constante și a parametrilor, după cum urmează:

047	- $\langle k \rangle$	inițial	$k = 1$
048	- $\langle j \rangle$	inițial	$j = 1$
049	- $\langle i \rangle$	inițial	$i = 1$
050	- $\langle \alpha \rangle$	inițial	$\alpha = 1$
051	- $\langle \beta \rangle$	inițial	$\beta = 1$
052	- $\langle P \rangle$	inițial	$P = 0$
053	- $\langle n \rangle$		
054	- $\langle 1 \rangle$		
055	- $\langle m \rangle$		
056	$\langle N_1 \rangle$	= 00 - 000	00 - 001
057	$\langle N_2 \rangle$	= 00 - 001	00 - 000
058	$\langle N_3 \rangle$	= 00 - 000	00 - 00n
059	$\langle N_4 \rangle$	= 00 - 00n	00 - 000
060	$\langle N_5 \rangle$	= 00 - 00n	00 - 00n

Atât în programul formulei lui Lagrange (3) cât și în programele care vor urma, celulele în care se păstrează variabilele și constantele problemei sînt simbolic identificate cu variabilele și constantele problemei.

3. Vom pune acum aceeași problemă ca și în cazul punctului 2, numai că vom hotărî executarea calculelor cu ajutorul formulei de interpolare a lui Newton.

Pentru aceasta vom avea nevoie în prealabil să calculăm diferențele divizate pe nodurile (1) ale funcției $f(x)$. Dacă calculele se execută după formula de definiție a diferențelor divizate, avem nevoie de tabela

x	$f(x)$	Δ	Δ^2	Δ^n
x_1	$f(x_1)$			
x_2	$f(x_2)$	$[x_1, x_2; f]$		
x_3	$f(x_3)$	$[x_2, x_3; f]$	$[x_1, x_2, x_3; f]$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_n	$f(x_n)$			
x_{n+1}	$f(x_{n+1})$	$[x_n, x_{n+1}; f]$		

(7)

care se obține după formula de recurență

$$[x_1, x_2, \dots, x_i, x_{i+1}; f] = \frac{[x_2, x_3, \dots, x_{i+1}; f] - [x_1, \dots, x_i; f]}{x_{i+1} - x_1} \quad (8)$$

Formula de interpolare a lui Newton pentru n noduri distincte oarecare este

$$f(x) = f(x_1) + (x - x_1)[x_1, x_2; f] + (x - x_1) \dots (x - x_n)[x_1, x_2, \dots, x_{n+1}; f] \quad (9)$$

Se observă însă că în formula (9) intervin numai diferențele divizate din tabela (7) care sînt la începutul fiecărei coloane, respectiv

$$f(x_1), [x_1, x_2, f], \dots, [x_1, x_2, \dots, x_{n+1}; f] \quad (10)$$

Diferențele divizate ale șirului (10) se pot obține însă și direct cu ajutorul formulei

$$[x_1, x_2, \dots, x_{n+1}; f] = \sum_{i=1}^{n+1} \frac{f(x_i)}{(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_{n+1})} \quad (11)$$

în care se vede că intră numai datele inițiale ale problemei $f(x_i)$ și x_i , $i = 1, 2, \dots, n + 1$.

Calculul după o asemenea formulă se poate programa destul de ușor ținînd cont de felul ei recursiv și de faptul că unii operatori au fost deja programați la formula lui Lagrange. Schema logică-operatorială a algoritmului prin care se realizează calculul diferențelor divizate (10) după formula (11) este de forma:

$$\begin{aligned} & \downarrow \downarrow \downarrow p(i \neq k) \uparrow A_1^k p(k < j + 1) \uparrow \downarrow F(k) O \uparrow \downarrow A_2^i \times \\ & \times p(i < j + 1) \uparrow F^{-(j+1)}(k) F(i) O \uparrow \downarrow p(j < n) \uparrow F^{-(j+1)}(k) \times \\ & \times F^{-(j+1)}(j) F(j) O \uparrow \downarrow B. \end{aligned} \quad (12)$$

Nu vom descrie amănunțit schema logică operațională (12) deoarece așa cum am mai menționat, operatorii ei au aceeași semnificație ca și în cazul schemei logice operatoriale (6). Calculul după schema (12) începe cu $i, j, k = 1$.

1) A_1^k este operator aritmetic după care se realizează produsul

$$\prod_{k=1}^j (x_i - x_k), \quad k \neq i.$$

2) A_2^i este de asemenea un operator aritmetic după care se execută cîtul

$$\frac{f(x_j)}{\prod_{k=1}^j (x_i - x_k)}, \quad (k = i)$$

și se adună la o celulă P_j .

În felul acesta, cînd j va lua valorile $1, 2, \dots, n+1$ în celulele $P_1, P_2, \dots, P_{n+1}$ vom avea înscrise diferențele divizate (10).

Se poate însă simplifica problema programării calculului elementelor șirului (10) în felul următor:

Vom aplica formula de recurență (8) pentru a obține elementele șirului (10). Programarea algoritmului care ne calculează șirul (10) se face în felul acesta mult mai simplu și fără a folosi celulele $P_1, P_2, \dots, P_{n+1}$ care în cazul numărului mare de noduri pot să ocupe o bună parte a memoriei mașinii. Se vede în formula (9) că dintre cele $n+1$ valori $f(x)$ ale funcției date, intră în formulă numai valoarea $f(x_1)$. Atunci ne vom rezerva de la început o celulă pentru $f(x_1)$ și în același timp $f(x_1), f(x_2), \dots, f(x_{n+1})$ se vor introduce în $n+1$ celule. Fie aceste celule $\alpha_1, \alpha_2, \dots, \alpha_n, \alpha_{n+1}$. În aceste celule vom face să apară diferențele divizate ale șirului (10).

Schema logică-operatorială a algoritmului care realizează acest proces este următoarea:

$$\downarrow \downarrow A_1^i p(i < n - k + 1) \uparrow F(i) O \uparrow \downarrow p(k < n) \times \\ F^{-(n-k)}(i) F(k) O \uparrow \downarrow F^{-(n+1)}(k) B. \quad (12')$$

Operatorul aritmetic A_1^i care intră în această schemă, realizează calculul după următoarea formulă

$$\frac{\alpha_{i+k} - \alpha_{i+k-1}}{x_{i+k} - x_i}$$

și înscrie rezultatul în celula α_{i+k-1} . Pe baza schemei logice-operatoriale (12') am întocmit programul pentru calculatorul electronic CIFA-2. El este următorul:

001	01 - $\langle x_{i+k} \rangle$	03 - $\langle x_i \rangle$
002	11 - $\langle P \rangle$	01 - $\langle \alpha_{i+k} \rangle$
003	03 - $\langle \alpha_{i+k-1} \rangle$	05 - $\langle P \rangle$
004	11 - $\langle \alpha_{i+k-1} \rangle$	01 - $\langle n \rangle$
005	03 - $\langle k \rangle$	02 - $\langle 1 \rangle$
006	11 - $\langle \alpha \rangle$	01 - $\langle i \rangle$
007	03 - $\langle \alpha \rangle$	06 - 016
008	01 - $\langle i \rangle$	02 - $\langle 1 \rangle$
009	11 - $\langle i \rangle$	01 - 001
010	02 - N_2	11 - 001
011	01 - 002	02 - N_2
012	11 - 002	01 - 003
013	02 - N_3	11 - 003
014	01 - 004	02 - N_3
015	11 - 004	10 - 001
016	01 - $\langle k \rangle$	03 - $\langle n \rangle$
017	06 - [023] _{II}	01 - $\langle n \rangle$
018	03 - $\langle k \rangle$	11 - $\langle \beta \rangle$
019	01 - $\langle i \rangle$	03 - β
020	11 - $\langle i \rangle$	01 - $\langle k \rangle$
021	02 - $\langle 1 \rangle$	11 - $\langle k \rangle$
022	02 - 001	02 - N_3
023	11 - 001	01 - 002

024	02 - N_2	11 - 002
025	01 - 003	02 - N_3
026	11 - 003	01 - 004
027	02 - N_3	11 - 004
028	10 - 001	00 - 000
029	$N_1 = 00 - 001$	00 - 001
030	$N_2 = 00 - 000$	00 - 001
031	$N_3 = 00 - 001$	00 - 000

Valorile i, k , inițial sînt egale cu unitatea.

Se vede că pentru un k dat, se obține o coloană a tabelului (7), care se păstrează în celulele $\alpha_1, \alpha_2, \dots, \alpha_{n+1}$.

Cînd k variază această coloană se schimbă în așa fel că la urmă pentru $k = n$ coloana noastră este chiar șirul (10), în care $f(x_1)$ lipsește, el fiind înscris într-o altă celulă stabilită dinainte. De asemenea celula α_{n+1} va putea fi eliberată deoarece totul se deplasează în sus cu un loc.

În schema logico-operatorială descrisă mai sus am notat prin B schema logică-operatorială a algoritmului prin care se realizează calculul valorilor aproximative $f(\xi_j)$ după formula lui Newton (9). Ea este de forma următoare

$$\downarrow \downarrow \downarrow A_1^i p(k < i) \uparrow F(k) O \uparrow \downarrow A_2^i A_3 p(i < n) \uparrow F^{-(i-1)}(k) F(i) \times \\ \times O \uparrow \downarrow T(\langle \alpha \rangle) p(j < m) \uparrow F^{-n}(k) F^{-n}(i) [f(x_1) \rightarrow \langle \alpha \rangle] F(j) O \uparrow \downarrow \text{Stop}. \quad (13)$$

1) Semnificația operatorului A_1^i este aceeași ca și în cazul schemelor descrise mai sus. După acest operator se execută produsul

$$\prod_{k=1}^i (\xi_j - x_k).$$

2) Operatorul A_2 este un operator aritmetic care execută produsul

$$\langle P_i \rangle > \prod_{k=1}^i (\xi_j - x_k),$$

unde

$$\langle P_i \rangle = [x_1, x_2, \dots, x_i, x_{i+1}; f].$$

3) Operatorul A_3 este de asemenea un operator aritmetic care adună rezultatul calculilor executate după operatorul de mai sus la o celulă fixă α .

4) Operatorul $[f(x_1) \rightarrow \langle \alpha \rangle]$ este un operator care transportă conținutul celulei $\langle P_1 \rangle = f(x_1)$ în celula α .

Subliniem aci că parametrii i, j, k care intervin în schema (13) nu sînt aceiași cu parametri j, i, k care intervin în schema (12).

Să observăm acum că formula (9) se poate pune și sub forma

$$f(x) = f(x_1) + u_1(\alpha_1 + u_2(\dots u_{n-3}(\alpha_{n-3} + \\ + u_{n-2}(\alpha_{n-2} + u_{n-1}(\alpha_{n-1} + u_n \alpha_n) \dots))), \quad (14)$$

unde am făcut următoarea notație

$$u_i = (x - x_i), \quad i = 1, 2, \dots, n,$$

$$\alpha_i = [x_1, x_2, \dots, x_i, x_{i+1}; f], \quad i = 1, 2, \dots, n$$

$$\alpha_0 = f(x_1),$$

iar indicii de deasupra parantezelor indică perechile de paranteze corespunzătoare.

Schema logico-operatorială a algoritmului prin care se realizează calculul valorilor aproximative $f(\xi_j)$ după formula (14) va fi de forma următoare

$$\begin{aligned} & \downarrow \downarrow A_i^n p(n > 1) \uparrow F(i) F^{-i}(n) O \uparrow \downarrow T(<\alpha>) \times \\ & \times p(j > m) \uparrow F^i(n) F^{-i}(j) F(j) O \uparrow \downarrow \text{Stop}. \end{aligned} \quad (15)$$

Semnificația singurului operator aritmetic al acestei scheme este că după el se execută calculul $u_n \alpha_n + \alpha_{n-1}$ și se trimite rezultatul în celula α .

Schemele (15) și (13) sînt echivalente din punctul de vedere al rezultatului realizării lor. Dar se poate observa simplu că schema (15) este mult mai avantajoasă decît schema (13) atît din punctul de vedere al simplității ei cît și din punctul de vedere al folosirii economice și cît mai eficace a memoriei mașinii pe care se realizează. Pentru justificarea acestei afirmații, vom da mai jos programul realizării acestor scheme pentru calculatorul electronic CIFA-2. Programul realizării schemei logico-operatoriale (13) este următorul:

001	01 - $\langle \xi_j \rangle$	03 - $\langle x_k \rangle$
002	04 - $\langle \beta \rangle$	11 - $\langle \beta \rangle$
003	01 - $\langle k \rangle$	03 - $\langle i \rangle$
004	06 - 008	01 - k
005	02 - $\langle 1 \rangle$	11 - $\langle k \rangle$
006	01 - 001	02 - N_1
007	11 - 001	01 - N_3
008	02 - N_1	11 - N_3
009	10 - 001	13 - 000
010	01 - $\langle \alpha_i \rangle$	04 - $\langle \beta \rangle$
011	02 - $\langle \alpha \rangle$	11 - $\langle \alpha \rangle$
012	01 - $\langle i \rangle$	03 - $\langle n \rangle$
013	06 - 022	01 - $\langle k \rangle$
014	03 - $\langle i - 1 \rangle$	11 - $\langle k \rangle$
015	01 - $\langle i \rangle$	02 - $\langle 1 \rangle$
016	11 - $\langle i \rangle$	01 - 001
017	03 - N_2	11 - 001
018	01 - 010	02 - N_3
019	11 - 010	01 - N_4
020	02 - N_3	11 - N_4
021	10 - 002	13 - 000

022	16 - $\langle \alpha \rangle$	01 - $\langle j \rangle$
023	03 - $\langle m \rangle$	06 - $[034]_{II}$
024	01 - $\langle k \rangle$	03 - $\langle n \rangle$
025	11 - $\langle k \rangle$	01 - $\langle i \rangle$
026	03 - $\langle n \rangle$	11 - $\langle i \rangle$
027	01 - $\langle f x_1 \rangle$	11 - $\langle \alpha \rangle$
028	01 - 001	03 - N_2
029	11 - 001	01 - 010
030	03 - N_4	11 - 010
031	01 - $\langle i \rangle$	02 - $\langle 1 \rangle$
032	11 - $\langle j \rangle$	01 - 001
033	02 - N_5	11 - 001
034	10 - 001	00 - 000

unde pe lângă cele 34 de celule folosite pentru program mai avem nevoie de încă 14 celule operative după cum urmează:

β al cărei conținut inițial = 1
 α al cărei conținut inițial = $f(x_1)$
 n inițial = 0
 m inițial = 0
 i inițial = 1
 j inițial = 1
 k inițial = 1
 l inițial = 1

$N_1 = 00 - 000 - 00 - 001$
 $N_2 = \text{inițial} = 00 - 000 \quad 00 - 001$
 $N_3 = 00 - 001 \quad 00 - 000$
 $N_4 = \text{inițial} = 00 - 001 \quad 00 - 000$
 $N_5 = 00 - 001 \quad 00 - 000$

În total avem nevoie de 48 de celule pentru program și pentru constantele operative în afara datelor inițiale.

Programul realizării schemei operatoriale (15) este următorul:

001	01 - $\langle \xi_j \rangle$	03 - $\langle x_n \rangle$
002	04 - $\langle \alpha \rangle$	02 - $\langle \alpha_{n-1} \rangle$
003	11 - $\langle \alpha \rangle$	02 - $\langle 1 \rangle$
004	03 - $\langle n \rangle$	06 - 013
005	01 - $\langle i \rangle$	02 - $\langle 1 \rangle$
006	11 - $\langle i \rangle$	01 - $\langle n \rangle$
007	03 - $\langle 1 \rangle$	11 - $\langle n \rangle$
008	01 - N_1	02 - N_2
009	11 - N_1	01 - 001
010	03 - N_1	11 - 001
011	01 - 002	03 - N_1
012	11 - 002	10 - 001
013	16 - $\langle \alpha \rangle$	01 - $\langle j \rangle$
014	03 - $\langle m \rangle$	06 - $[024]_{II}$
015	01 - $\langle n \rangle$	02 - $\langle i \rangle$
016	11 - $\langle n \rangle$	01 - $\langle i \rangle$
017	03 - $\langle i \rangle$	11 - $\langle i \rangle$
018	01 - 001	02 - N_2

019	11 - 001	01 - 002
020	02 - N_2	11 - 002
021	01 - $\langle j \rangle$	02 - $\langle 1 \rangle$
022	11 - $\langle j \rangle$	01 - 001
023	02 - N_3	11 - 001
024	10 - 001	00 - 000

unde pe lângă cele 24 de celule ocupate de program mai avem nevoie de încă 9 celule operative după cum urmează :

α al cărei conținut inițial este $= \alpha_n$
 i inițial = 0
 j inițial = 1
 n inițial = n
 m inițial = m
 1 inițial = 1

$N_1 = 00 - 000 \quad 00 - 000$
 $N_2 = 00 - 000 \quad 00 - 001$
 $N_3 = 00 - 001 \quad 00 - 000$

Confruntând cele două programe, rezultă ușor că afirmația noastră este adevărată.

Menționăm că în lucrările [4] și [5] s-au realizat programe pentru cazurile particulare a 3 respectiv 4 noduri.

Pentru a găsi însă o formulă de interpolare admisă de tabloul (7), care să aibă cei mai mici coeficienți în valoare absolută [1], [2], [3], va trebui să găsim o permutare a indicilor $v_1, v_2, \dots, v_n$ astfel ca $|\xi_j - v_1|, |\xi_j - v_2|, \dots, |\xi_j - v_n|$ să formeze un șir nedescrescător. Această problemă se poate rezolva atașând algoritmului care rezolvă calcularea valorilor aproximative $f(\xi_i)$ un algoritm care să execute înaintea celui amintit, ordonarea unui șir de forma

$$a_1, a_2, \dots, a_n,$$

în așa fel ca

$$a_{v_1} \leq a_{v_2} \leq \dots \leq a_{v_n}.$$

Un astfel de algoritm există, și asupra lui vom reveni într-o lucrare ulterioară.

Formula lui Gauss

$$\begin{aligned} f(a + hx) = & f(a) + x\Delta f(a) + \frac{x(x-1)}{2!} \Delta^2 f(a-h) + \\ & + \frac{(x+1)x(x-1)}{3!} \Delta^3 f(a-h) + \frac{(x+1)x(x-1)(x-2)}{4!} \Delta^4 f(a-2h) + \\ & + \frac{(x+2)(x+1)x(x-1)(x-2)}{5!} \Delta^5 f(a-2h) + \dots \end{aligned} \quad (16)$$

este un caz particular al formulei lui Newton, unde nodurile sînt considerate echidistante și sînt așezate simetric față de nodul a , astfel

$$a, a + h, a - h, a + 2h, a - 2h, \dots,$$

sau

$$a, a - h, a + h, a - 2h, a + 2h, \dots$$

Din acest motiv considerăm că nu este necesar să studiem separat din punct de vedere al programării formula lui Gauss, ea fiind doar un caz particular al formulei generale a lui Newton studiată mai sus.

Formulele lui Stirling

$$\begin{aligned} f(a + hx) = & f(a) + x \frac{\Delta f(a) + \Delta f(a-h)}{2} + \frac{x}{2!} \Delta^2 f(a-h) + \\ & + \frac{x(x^2-1)}{3!} \frac{\Delta^3 f(a-h) + \Delta f(a-2h)}{2} + \frac{x^2(x^2-1)}{4!} \Delta^4 f(a-2h) + \\ & + \frac{x(x^2-1)(x^2-2)}{5!} \frac{\Delta^5 f(a-2h) + \Delta^3 f(a-3h)}{2} + \frac{x^2(x^2-1)(x^2-2)}{6} x \Delta^6 f(a-3h) + \dots \end{aligned} \quad (17)$$

și a lui Bessel

$$\begin{aligned} f(a + hx) = & \frac{f(a) + f(a+h)}{2} + \left(x - \frac{1}{2}\right) \Delta f(a) + \\ & + \frac{x(x-1)}{2!} \cdot \frac{\Delta^2 f(a-h) + \Delta^2 f(a)}{2} + \frac{x(x-1)\left(x - \frac{1}{2}\right)}{3!} \Delta^3 f(a-h) + \\ & + \frac{(x+1)x(x-1)(x-2)}{4!} \cdot \frac{\Delta f(a-2h) + \Delta^4 f(a-h)}{2} + \dots \end{aligned} \quad (18)$$

sînt consecințe ale formulei lui Gauss obținute prin regruparea termenilor și scrierea sub o altă formă a diferențelor de diferite ordine. Din punctul de vedere al programării formulele (17) și (18) nu prezintă avantaje față de formula generală (9) respectiv (14) tratată de noi, ci dimpotrivă realizarea programelor pentru aceste formule duce la anumite dificultăți din punctul de vedere al ciclizării calculelor.

4. Considerăm acum formula de interpolare a lui Hermite

$$H(x_1, x_2, \dots, x_{n+1}; f(x)) = \sum_{i=1}^{n+1} h_i(x) f(x_i) + \sum_{i=1}^{n+1} k_i(x) f'(x_i), \quad (19)$$

unde pe lângă valorile funcției $f(x)$ pe nodurile $x_1, x_2, \dots, x_{n+1}$ se dau și valorile derivatei $f'(x)$, pe aceleași noduri. În formula (19) funcțiile $h_i(x)$ și $k_i(x)$ sînt date de relațiile următoare.

$$h_i(x) = l_i^2(x) V_i(x),$$

unde

$$V_i(x) = 1 - 2l_i'(x_i)(x - x_i),$$

iar

$$k_i(x) = l_i^2(x)(x - x_i).$$

Din punct de vedere al folosirii economice a memoriei mașinii și al ciclizării calculelor este avantajos ca formula (19) să se pună sub forma

$$H(x_1, x_2, \dots, x_{n+1}; f(x)) = \sum_{i=1}^{n+1} (k_i(x)f'(x_i) + h_i(x)f(x_i)). \quad (19')$$

Schema logică operatorială a algoritmului prin care se realizează calcularea valorilor aproximative $f(\xi_j)$, ($j = 1, 2, \dots, m$), după formula (19') este de forma următoare:

$$\begin{aligned} & \downarrow \downarrow \downarrow A_1^k p(k < n+1) \uparrow F(k) O \uparrow \downarrow F^{-n}(k) \downarrow p(i \neq k) \uparrow \times \\ & \times A_2^k p(k < n+1) \uparrow \downarrow F(k) O \uparrow \downarrow A_3^i A_4 A_5 A_6 A_7 A_8 A_9^j \times \\ & \times F^{-n}(k) p(i < n+1) \uparrow F(i) O \uparrow \downarrow F^{-n}(k) F^{-n}(i) p(j < m) \uparrow \times \\ & \times F(j) O \uparrow \downarrow T\{<P_j>\} \text{Stop}. \end{aligned} \quad (20)$$

1) A_1^k este un operator aritmetic care calculează produsul $\prod_{k=1}^{n+1} (\xi_j - x_k)$ și îl înscrîm în celula r_1 .

2) A_2^k este de asemenea un operator aritmetic care realizează produsul

$$\prod_{k=1}^{n+1} (x_i - x_k), \quad i \neq k,$$

și îl înscrîm în celula α . În acest fel avem $<\alpha> = l_i(x_i)$.

3) Operatorul A_3^i este aritmetic și realizează produsul

$$<\alpha> \cdot (\xi_j - x_i).$$

4) Operatorul A_4 este aritmetic, realizează cîmul

$$<r_1> : <\alpha> \cdot (\xi_j - x_i) = l_i(\xi_j)$$

și îl înscrîm în celula β .

5) Operatorul A_5 este aritmetic și realizează pătratul lui $l_i(\xi_j)$.

6) Operatorul A_5 este aritmetic și realizează valoarea funcției $k_i(x)$ în punctul ξ_j , conform formulei

$$k_i(\xi_j) = <\beta>^2 \cdot (\xi_i - x_i).$$

7) Operatorul A_7 este aritmetic și realizează produsul $k_i(\xi_j)f'(x_i)$, apoi adună acest produs la o celulă P_j .

8) Operatorul A_8 este aritmetic și realizează valoarea funcției $h_i(\xi_j)$ după formula $1 - 2<\alpha>(\xi_j - x_i) <\beta>^2$, face produsul acestei valori cu $f(x_i)$ și adună la celula P_j .

9) Operatorul $T\{<P_j>\}$ este operator de tipărire și realizează tipărirea șirului de celule $P_1, P_2, \dots, P_m$. Dacă memoria mașinii nu permite păstrarea valorilor $f(\xi_j)$ în celulele $P_1, P_2, \dots, P_m$, atunci se poate realiza tipărirea imediat după ce s-a obținut o valoare $f(\xi_j)$.

Restul operatorilor și condițiilor logice considerăm că nu merită o explicație specială. Realizarea schemei începe cu $i, j, k = 1$.

Deoarece aproape toți operatorii schemei (20) au fost programați în punctele de mai sus, precum și pentru a nu lungi prea mult expunerea, considerăm că nu este necesar să întocmim programul de realizare a schemei logice-operatoriale (20).

5. În încheierea acestei lucrări vom da schema logică operatorială a algoritmului prin care se realizează calcularea valorilor aproximative $f(\xi_j)$ după formula lui Bernstein.

$$B_n(x; f) = \sum_{i=0}^n \binom{n}{i} f\left(\frac{i}{n}\right) x^i (1-x)^{n-i}. \quad (21)$$

Această schemă se prezintă astfel:

$$\begin{aligned} & A_0 \downarrow \downarrow \downarrow A_1^k p(k < i-1) \uparrow A_2^i A_3^i A_4^i A_5 \times \\ & \times p(i < n) \uparrow F^{-(i-1)}(k) F(i) O \uparrow \downarrow p(j < m) \uparrow F^{-n}(k) \times \\ & \times F^{-n}(i) F(j) O \uparrow \downarrow T\{<P_j>\} \text{Stop}. \end{aligned}$$

Semnificația operatorilor aritmetici ai acestei scheme este următoarea:

1) A_1 execută calculul $f(0)(1 - \xi_j)^n$ iar rezultatul îl înscrîm în celula β .

2) A_1^k execută produsul $<\alpha> \cdot (n - k)$ și îl împarte la $k + 1$. Inițial $<\alpha> = 1$. Rezultatul se înscrîm în celula α .

3) A_2^i efectuează produsul $<\alpha> f\left(\frac{i}{n}\right)$ iar rezultatul îl înscrîm în celula α .

4) A_3^i execută ξ_j^i îl înmulțește cu $<\alpha>$ iar rezultatul îl înscrîm în α .

5) A_4^i execută $(1 - \xi_j)^{n-1}$, și îl înmulțește cu $<\alpha>$. Rezultatul se înscrîm în α .

6) A_5 efectuează produsul $<\alpha> \cdot <\beta>$ și îl adună la o celulă P_j .

7) $T\{<P_j>\}$ este operatorul care execută tipărirea șirului de celule P_j în care avem păstrate valorile $f(\xi_j)$.

О ПРОГРАММИРОВАНИИ ДЛЯ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН ИНТЕРПОЛЯЦИОННЫХ ФОРМУЛ

РЕЗЮМЕ

В работе проводится изучение некоторых интерполяционных формул с точки зрения их программирования для электронной вычислительной машины.

В случае формул Лагранжа и Ньютона, кроме изучения их логико-операторных схем, дается и соответствующая программа; при этом

Интерполяционные формулы Эрмита и Бернштейна изучаются только с точки зрения соответствующих им логико-операторных схем.

RÉSUMÉ

Pour les formules de Lagrange et de Newton, outre l'étude de leurs schémas logiques-opérationnels, les auteurs en indiquent aussi les programmes correspondants, en utilisant le code de la machine électronique CIFA-2. Ils étudient également deux variantes du schéma logique-opérationnel de l'algorithme de réalisation d'un tableau de différences divisées.

Les formules d'interpolation d'Hermite et de Bernstein ne sont étudiées qu'au point de vue des schémas logiques-opérationnels qui leur correspondent.

1. Поповициу Т., *Considerații asupra utilizării practice a unor formule de interpolare*. Bul. științ. Acad. R.P.R., Secț. de științe matematice și fizice, III, 441—449 (1951).
2. — *Despre precizia calcului numeric în interpolarea prin polinoame*. Bul. Științ. Acad. R.P.R., Secț. de științe matematice și fizice, VIII, 954—961 (1955)
3. — *Despre precizia calculului numeric în interpolarea prin polinomul lui Newton cu noduri echidistante*. Studii și cercet. științ. (Cluj), Ser. I, VI, 3—4, 27—35 (1955).
4. Э. Н. Чайковская, *Квадратичная интерполяция по формуле Ньютона с разностями*. Сборник стандартных и типовых программ для БЭСМ. Изд. Акад. Наук СССР, Москва, 1960.
5. Э. Н. Чайковская, *Кубическая интерполяция по формуле Ньютона с разделенными разностями*. Сборник стандартных и типовых программ для БЭСМ. Изд. Акад. Наук СССР, Москва, 1960.
6. И. Янов, *О логические схемах алгоритмов*. Проблемы кибернетики, 1, 75—127 (1958).

Primit la 12. I. 1962.

RESEARCH, PRODUCTION AND MARKETING OF
A NEW RANGE OF FINEST QUALITY CELESTIAL
AND MARINE ELECTRONICS IN CANADA.

4. If $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n$ are distinct in $\mathcal{G}$ precisely the often, $\mathcal{G}$ is never strictly stable. If $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n$ are not distinct in $\mathcal{G}$ precisely, then $\mathcal{G}$ is $\mathcal{G}$ is consistently generated. $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n$ are generated in $\mathcal{G}$ if $\mathcal{G} = 1$, the $\mathcal{G}$ is consistent in $\mathcal{G}$ precisely if $\mathcal{G}$ is not strictly stable. In the case of $\mathcal{G}$ is not strictly stable, $\mathcal{G}$ is not strictly stable, which is not strictly stable and not strictly stable. In the case of $\mathcal{G}$ is not strictly stable, $\mathcal{G}$ is not strictly stable.

Pravna stranica: Posrednik odgovoran za pravnu ispravnost i sadržaj informacija koje pruža, a ne garantira ispravnost ili sadržaj informacija koje pružaju drugi posrednici.

$\mathcal{W}_1, \mathcal{W}_2, \dots, \mathcal{W}_k$ gli insiemi ottenuti da ogni $\mathcal{W}_i$ in $\mathcal{W}$ sostituendo ad $\mathcal{W}_i$ l'insieme $\mathcal{W}_i'$.

Students will benefit as well as others by giving feedback to others about the material presented (Lippitt, 1969, p. 14) even as they learn to participate in it.

qualche $\mu(F_1, F_2)$ sotto struttura che ha una griglia F_1 ed è distributivo ha una griglia F_2 che è abeliana.

Received: accepted: published: 2014

⁷⁴ *Journal of American Studies*, 1990, 24, 1, 107–122. *Journal of American Studies*, 1990, 24, 1, 107–122.