ALGORYTMY

VOL. II • NO. 3 • 1964

INSTYTUT MASZYN MATEMATYCZNYCH PAN

A L G O R Y T M Y

Vol. II N°3 1964

P R A C E

Instytutu Maszyn Matematycznych

P o l s k i e j A k a d e m i i N a u k

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

T R E Ś Ć

CONTENTS

N U M E R I C A L   A N A L Y S I S

Instytut Maszyn Matematycznych PAN
Algorytmy N° 3
© 1964.03

518.12:512.31:518.6

OBLICZANIE WSPÓŁCZYNNIKÓW WIELOMIANU INTERPOLACYJNEGO

Klaudia BRATKOWSKA
Włodzimierz OSTALSKI

Pracę złożono 16 kwietnia 1963 r.

Podano opis programu dla maszyny ZAM-2 wyliczającego współczynniki wielomianu interpolacyjnego.

1. Wstęp

W często występującym w praktyce zagadnieniu interpolacji wielomianem wystarcza czasami otrzymywanie wartości tego wielomianu, jednakże niejednokrotnie konieczne są współczynniki wielomianu interpolacyjnego w postaci:

$$a_0 + a_1x + a_2x^2 + \dots + a_nx^n \qquad /1/$$

Do wyliczenia wartości wielomianu interpolującego tablicę podaną w nierównych odstępach wygodnie jest używać wzoru interpolacyjnego Newtona, którego stosowanie nie nastręcza żadnych trudności:

$$y = [x] + [x_0x_1](x-x_0) + [x_0x_1x_2](x-x_0)(x-x_1) + \dots + \\ + [x_0x_1 \dots x_n](x-x_0)(x-x_1)\dots(x-x_{n-1}) , \qquad /2/$$

gdzie: $\left[x_0x_1\right] = \frac{y_0-y_1}{x_0-x_1}$ $\left[x_1x_2\right] = \frac{y_1-y_2}{x_1-x_2}$

$$\left[x_0x_1\ldots x_n\right] = \frac{\left[x_0x_1\ldots x_{n-1}\right] - \left[x_1x_2\ldots x_n\right]}{x_0 - x_n}$$

Wzór /2/ jest jednak niewygodny do obliczania współczynników wielomianu interpolacyjnego w postaci /1/.

2. Opis algorytmu

Jeżeli wartości $y_0, y_1, \ldots, y_n$ w interpolowanej tablicy podane są w punktach x_0, x_0+H, x_0+2H, ..., x_0+nH, wzór interpolacyjny Newtona można zapisać w postaci:

$$y = y_0 + \left(\frac{\Delta y_0}{1!} S_0^1 + \frac{\Delta^2 y_0}{2!} S_1^2 + \ldots + \frac{\Delta^n y_0}{n!} S_{n-1}^n\right)\frac{(x-x_0)}{H} + \left(\frac{\Delta^2 y_0}{2!} S_0^2 + \ldots + \frac{\Delta^n y_0}{n!} S_{n-2}^n\right)\left(\frac{x-x_0}{H}\right)^2 + \ldots + \frac{\Delta^n y_0}{n!} S_0^n\left(\frac{x-x_0}{H}\right)^n, \qquad /3/$$

gdzie: $\Delta^i y_0$ - i-ta różnica w punkcie y_0,

$H = x_i - x_{i-1}$,

S_i^k - liczby Stirlinga pierwszego rodzaju, określone jak następuje:

$S_k^k = 0, \quad S_0^k = 1, \quad S_i^{k+1} = S_i^k - kS_{i-1}^k.$

Jeżeli we wzorze /3/ przyjmiemy $x_0 = 0$, możemy zapisać go w postaci:

$$y = y_0 + \left(\Delta y_0 D_0^1 + \Delta^2 y_0 D_1^2 + \ldots + \Delta^n y_0 D_{n-1}^n\right)x + \left(\Delta^2 y_0 D_0^2 + \Delta^3 y_0 D_1^3 + \ldots + \Delta^n y_0 D_{n-2}^n\right)x^2 + \ldots + \left(\Delta^n y_0 D_0^n\right)x^n, \qquad /4/$$

gdzie: $D_0^k = \frac{1}{k!H^k}$ $D_k^k = 0$

$$D_i^{k+1} = \frac{1}{k+1}\left(\frac{D_i^k}{H} - kD_{i-1}^k\right).$$

Porównując wzory /1/ i /4/, otrzymujemy proste wzory na współczynniki wielomianu interpolacyjnego:

$$\begin{aligned} a_0 &= y_0 \\ a_1 &= \Delta y_0 D_0^1 + \Delta^2 y_0 D_1^2 + \ldots + \Delta^n y_0 D_{n-1}^n \\ a_2 &= \Delta^2 y_0 D_0^2 + \Delta^3 y_0 D_1^3 + \ldots + \Delta^n y_0 D_{n-2}^n \\ &\cdot \\ &\cdot \\ &\cdot \\ a_n &= \Delta^n y_0 D_0^n . \end{aligned} \qquad /5/$$

Aby móc zastosować wzory /5/ /przy założeniu, że tablica zadana jest w punktach dowolnych/ należy najpierw obliczyć różnice dzielone występujące we wzorze /2/, a następnie ze wzoru /2/- wartości wielomianu interpolacyjnego w punktach

$$X = 0, \quad H, \quad 2H, \ldots, nH.$$

Dalej wylicza się różnice w punkcie 0 oraz liczby D_i^k .

3. Opis programu

Opisany powyżej algorytm został zaprogramowany dla maszyny ZAM-2 w arytmetyce stałoprzecinkowej, na liczbach podwójnie długich /tzn. 70 bitowych, co odpowiada ok. 21 cyfrom dziesiętnym/. Dla H przyjęto wartość początkową 1. Przy H = 1 występuje obliczanie wartości wielomianu interpolacyjnego dla x = n , co dla dużych $n\,(>10)$ powoduje powstawanie w trakcie liczenia bardzo dużych liczb, a więc zwiększenie skali i zmniejszenie dokładności. W związku z tym, wraz z tablicą danych wprowadza się liczbę całkowitą ograniczającą z góry skalę binarną. Znaczenie tej liczby jest następujące: niech liczba ograniczająca skalę = c; jeżeli w trakcie wyliczania wartości wielomianu w punkcie i·H i = 1, 2, ..., n wystąpi liczba większa /co do modułu/ niż 2^c, wówczas na miejsce H zostanie wstawiona wartość $\frac{H}{2}$ i powtarza się liczenie wartości wielomianu.

Program realizujący podany wyżej schemat został napisany przez K. Bratkowską dla wielomianów stopnia nie większego niż 30. Z przeprowadzonych prób wynika, że dla wielomianów stopnia niższego niż 12 otrzymuje się na ogół wyniki z zadowalającą dokładnością.

4. Opis wyników

Przy pomocy tego programu dokonano obliczeń dla kilkunastu tablic zawierających od 3 do 20 punktów. Otrzymane wielomiany sprawdzono przez wyliczenie ich wartości w węzłach.

Jak widać z przykładów, interpolowane tablice miały bardzo różny charakter.

Z przytoczonych przykładów można wyciągnąć następujące wnioski:

1. Mimo liczenia na liczbach 70-bitowych /co odpowiada około 21 cyfrom dziesiętnym/, błąd zaokrąglenia jest duży. Bardzo dokładne wyniki otrzymuje się jedynie dla wielomianów niskiego stopnia. /Przykłady oznaczone literami A, G, H, J, P/.
2. 'Regularność' tablicy ma duży wpływ na wyniki. Dla tablicy o dużych i gęstych wahaniach otrzymuje się wielomian dający bardzo małą zgodność /przykład L/. Jednakże dla tablic takich jak w przykładzie /K/, mimo ich regularności, otrzymujemy wielomian dobrze zgodny z tablicą tylko dla argumentów bliskich zera.

3. W jednym przypadku /przykład J/, po otrzymaniu wielomianu dobrze zgodnego z tablicą, wyliczono wartości tego wielomianu, tak że otrzymano 2 nowe tablice: jedna mieszcząca się w przedziale tablicy pierwotnej, druga - w przedziale szerszym. Dla pierwszej otrzymano współczynniki wielomianu identyczne do 13 cyfr dziesiętnych po kropce ze współczynnikami dla tablicy pierwotnej /przykład J'/, dla drugiej 11 cyfr dziesiętnych /przykład J''/.

4. Zmiana ograniczenia skali /a w wyniku zmiana H/ powoduje zmiany we współczynnikach niewielkie, o ile nie przekracza ona pewnej granicy /przykłady C i C'/. Ograniczenie bliskie maksymalnej skali liczenia /np. większe niż 50/ może spowodować błędy w wynikach.

Literatura

1. ŁUKASZEWICZ J., WARMUS M.: Metody numeryczne i graficzne, PWN, Warszawa 1956.

COMPUTING OF COEFFICIENTS OF INTERPOLATING POLYNOMIAL

Summary

The paper presents an algorithm and the general description of a program by means of which coefficients of an interpolating polynomial are computed. The program is prepared for ZAM-2 digital computer and it operates on seventy-bit numbers. This permits to compute coefficients of an interpolating polynomial for a table containing up to 30 values. Examples prove the efficiency of the program containing up to 12 values. However, in case of a very irregular diagram of the interpolated function the result will be unsatisfactory, even if the number of values is small.

INTERPOLACJA WIELOMIANEM

Algorytm w ALGOL

```
begin integer  N, i, l, j, k, p, w, A;

read  N, A;
real array  X [0:30] , F [0:30] , R [0:30] , Y [0:30] , D [0:30] ;
   i :=N-1;

for  l:=0  step 1 until i do read X [l] ;
for  l:=0  step 1 until i do read F [l] ;
for  k:=1  step 1 until i do
begin for l:=i step -1 until k do
begin F [l] := (F [l]  - F [l-1]) / (X [l]  - X [l-k]) ;
      D [l] := F [l] ;
end l;
end k;
H := 1;

LL:  for l:=0 step 1 until i do
     begin  Y [l]  := - X [l] ;  F [l]  := D [l] ;  end;
     for j:=0 step 1 until i do
     begin  R [j]  := F [i] ;

     for l:=i step -1 until 1 do R [j] := R [j] × Y [l-1]  + F [l-1] ;
     for l:=0 step  1 until i do Y [l] := Y [l]  + H;
     end j;
     for j:=0 step  1 until i do F [j] := R [j] ;
     for k:=1 step  1 until i do
     begin for p:=i step 1 until k do
        F [p] := F [p]  - F [p-1] ;
     end k;
```

```
        Y[0] := 1/H;
        for l:=1 step 1 until i do Y[l]:=0;
        for l:=1 step 1 until i do
        begin w:=l+1; F[l]:=Y[0]× F[l];
        if l=i then go to MM else p:=1; R:=Y[0];
          for k:=w step 1 until i do
          begin Y[p+1] := (Y[p+1]/H -(k-1)×R)/k;
                F[l]:= F[l]/H + F[k] × Y[p+1] ;
                p:=p+1; R:=Y[p];
          end k;
        if abs(F[l]) ≥ 2↑A then go to KK
           Y[0]:= Y[0]/(l+1) ;
        end l;
MM:     for l:=0 step 1 until i do
        begin carriage return; punch F[l]; end;
KK:     H:=H/2;
        if H > 0 then go to LL else go to KK
        end
```

Uwaga:

Podany tu program w języku ALGOL dosyć znacznie różni się od programu rzeczywiście realizowanego. Wynika to z różnic języków, w których programy te zostały napisane.

P R Z Y K Ł A D Y

A. $n = 5$, $H = 1.0$

Tablica danych x	y	wartości wyliczone*)
5.0	30.0	30.00000000920000
10.0	105.0	
15.0	270.0	
20.0	570.0	570.00001062356662018
45.0	6480.0	6480.0002952706390715
60.0	16800.0	16800.0015922245347610

Współczynniki wielomianu interpolacyjnego.

$$a_0 = 13.8311688311669467$$
$$a_1 = -1.3132034632026392$$
$$a_2 = 0.8277417027415766$$
$$a_3 = 0.0109668109668195$$
$$a_4 = 0.0010894660894657$$
$$a_5 = -0.0000033477633477$$

B. $n = 10$, $H = 0.5$

Tablica danych x	y	wartości wyliczone
0.0	0.0	
0.1	0.5206578756	
0.2	0.6154579181	0.615457918099955
0.3	0.6742996467	
0.5	0.7416565702	0.741656570200137
1.0	0.7522313333	0.752231333309000
1.5	0.6192139088	0.619213909111801
2.2	0.2962272134	
2.5	0.1405701237	
3.0	-0.1006370643	-0.100636858916083
3.5	-0.281924138	-0.281923249674127

---

* Ilość cyfr po kropce odpowiednia do wartości skali.

Współczynniki wielomianu interpolacyjnego.

a_0 = 0.0
a_1 = 10.5296513950570 79
a_2 = -79.216074018031025
a_3 = 326.015490443919255
a_4 = -756.601199387760592
a_5 = 1030.482326394069385
a_6 = -847.934011840452446
a_7 = 424.570864214461692
a_8 = -126.079007913641411
a_9 = 20.358416956755956
a_{10} = 1.374224911074612

C. n = 11 H = 0.125

Tablica danych wartości wyliczone

x	y	wartości wyliczone
-0.5235987	-0.5	
-0.3490658	-0.342020143	
-0.1745329	-0.173648177	
0.01745329	0.017452406	
0.0319066	0.0340899496	
0.08726645	0.087155742	
0.1745329	0.173648177	0.213820839435395
0.19198619	0.190808995	
0.5235987	0.5	
0.78539805	0.707106781	
1.0471974	0.866025403	

Współczynniki wielomianu interpolacyjnego.

	C	C' (H = 0.25)
a_0 =	0.001886889976357	0.0018868899762
a_1 =	0.856234524933541	0.856232403781
a_2 =	2.143550217575408	2.143575306452
a_3 =	-4.631437223160027	-4.631559410572
a_4 =	-82.050034366918333	-82.049706301393
a_5 =	418.442164015096774	418.441619327971
a_6 =	185.373897699007850	185.374487673791
a_7 =	-4071.547312058572197	-4071.547737041081
a_8 =	4812.053260588207043	4812.053462583638
a_9 =	7412.402277699596420	7412.402216226898
a_{10} =	-16417.405637446650426	-16417.405627289590
a_{11} =	7710.231152463859587	7710.231151570360

D. n = 12 H = 1

Tablica danych

x	y	
-0.5235987	-0.5	
-0.3490658	-0.342020143	-0.344898815025
-0.1745329	-0.173648177	
0.01745329	0.017452406	
0.03490658	0.034899496	
0.08726645	0.087155742	
0.1745329	0.173648177	
0.19198619	0.190808995	
0.3490658	0.342020143	
0.5235987	0.5	
0.78539805	0.707106781	
1.0471974	0.866025403	
9.108655	-0.173648178	

Współczynniki wielomianu interpolacyjnego.

a_0 = 0.000000000110
a_1 = 1.003201143462
a_2 = -0.009417057452
a_3 = -0.155295204435
a_4 = -0.007586091205
a_5 = 0.011584225976
a_6 = -0.000857274709
a_7 = -0.001359785289
a_8 = 0.001963804314
a_9 = 0.002076068471
a_{10} = -0.005935319400
a_{11} = 0.003389427252
a_{12} = -0.000303339231

G. n = 3 H = 1

Tablica danych.

x	y
-2.0	99.0
1.0	192.0
2.0	175.0
4.0	105.0

Współczynniki wielomianu interpolacyjnego.*)

a_0 = 189.0
a_1 = 15.0
a_2 = -13.0
a_3 = 1.0

---

* Błąd obliczenia współczynników $< 10^{-18}$

H. $n = 4$ $H = 1$

Tablica danych

x	y
-2.0	8.0
-1.0	-2.0
0.0	1.0
1.0	-4.0
2.0	10.0

Współczynniki wielomianu interpolacyjnego /z błędem $< 3\cdot 10^{-17}$/

$a_0 = 1.0$
$-\varepsilon_1 = -1.5$
$a_2 = -6.0$
$a_3 = 0.5$
$a_4 = 2.0$

I. $n = 3$ $H = 1$

Tablica danych

x	y
1.1	0.769
1.2	0.472
1.4	-0.344
1.5	-0.875

Współczynniki wielomianu interpolacyjnego /z błędem $< 2\cdot 10^{-17}$/

$a_0 = 1.0$
$a_1 = 1.0$
$a_2 = 0.0$
$a_3 = -1.0$

J. n = 7 H = 1

Tablica danych

x	y
0.2	0.57926
0.7	0.75804
1.2	0.88493
1.7	0.95543
2.2	0.98610
2.7	0.99653
3.2	0.99931
3.7	0.99989

Współczynniki wielomianu interpolacyjnego.

a_0 = 0.502203384089600000
a_1 = 0.378952759850666642
a_2 = 0.058957999217777835
a_3 = -0.147589872000000053
a_4 = 0.056933271111111136
a_5 = -0.008249546666666672
a_6 = 0.000139911111111111
a_7 = 0.000048000000000000

J' n = 7 H = 1

Tablica danych.

x	y
0.3	0.617651731020799900
0.5	0.691273622041599900
1.0	0.841395906713599000
2.0	0.977267080217471977
2.1	0.982148142003019860
2.5	0.993772001048989250
3.0	0.998683432599405500
3.3	0.999465790024508770

Współczynniki wielomianu interpolacyjnego.

a_0 = 0.502033840896020084
a_1 = 0.378952759850649078
a_2 = 0.058957999217830526
a_3 = -0.147589872000077219
a_4 = 0.056933271111172744
a_5 = -0.008249546666694168
a_6 = 0.000139911111117581
a_7 = 0.000047999999998372

J'' n = 7 H = 1

Tablica danych.

x	y
0.5	0.691273622041599900
1.0	0.841395906713599000
2.0	0.977267080217471977
2.1	0.982148142003019860
3.0	0.998683432599405500
3.3	0.999465790024508770
4.0	1.002480131849084000
8.0	31.958810340766560000

Współczynniki wielomianu interpolacyjnego.

a_0 = 0.502203384192007966 3
a_1 = 0.378952759352715316 4
a_2 = 0.058958000134364287 1
a_3 = -0.147589872850415386 3
a_4 = 0.056933271549704492 8
a_5 = -0.008249546794225448 2
a_6 = 0.000139911130679265 3
a_7 = 0.000047999998768506 3

K. $n = 9$ $H = 1$

Tablica danych.

x	y
-5.0	-1.0
-4.0	-1.0
-3.0	-1.0
-2.0	-1.0
-1.0	-1.0
1.0	1.0
2.0	1.0
3.0	1.0
4.0	1.0
5.0	1.0

Współczynniki wielomianu interpolacyjnego.

a_0 = -0.0000000000000001
a_1 = 1.2912698412622723
a_2 = 0.000000000266800
a_3 = -0.3302469136027034
a_4 = 0.0000000000129141
a_5 = 0.0410879629586022
a_6 = 0.0000000000008936
a_7 = -0.0021494708995804
a_8 = 0.0000000000000073
a_9 = 0.0000385802469132

L. n = 14 H = 0.125

Tablica danych.

x	y	wartość wyliczona.
-2.5	-0.125	-0.13448320736700
-2.4	-0.5	
-2.3	-0.155	
-2.0	-0.3408	
0.0	-0.05	
0.3	0.524	
0.5	0.65	
0.7	1.25	
0.9	1.5	
1.2	2,2	
1.5	2.5	
1.7	2.95	
1.8	3.0	
2.0	3.5	

Współczynniki wielomianu interpolacyjnego.

a_0 = -0.05
a_1 = 2159.56281380303735
a_2 = -19560.93062937252900
a_3 = 66717.71545104550442
a_4 = -102812.69534887361531
a_5 = 51767.30203405793610
a_6 = 45252.35246783454346
a_7 = -62288.37404333366032
a_8 = 7457.68842499172391
a_9 = 19044.35739096899258
a_{10}= -6591.50111741536506
a_{11}= -2336.12259769828206
a_{12}= 1163.63194305535420
a_{13}= 99.11705078267792
a_{14}= -67.05383984631813

P. $n = 4$ $H = 1$

Tablica danych.

x	y
-2.0	8.0
-1.0	-2.0
0.0	1.0
1.0	-4.0
2.0	10.0

Współczynniki wielomianu interpolacyjnego.

/z błędem $< 3\cdot 10^{-17}$/

$a_0 = 1.0$
$a_1 = -1.5$
$a_2 = -6.0$
$a_3 = 0.5$
$a_4 = 2.0$

Instytut Maszyn Matematycznych PAN 517.91:518.61
Algorytmy N° 3
© 1964.03

A METHOD OF SOLVING THE BOUNDARY VALUE PROBLEM FOR A SYSTEM OF LINEAR ORDINARY DIFFERENTIAL EQUATIONS

by Krzysztof MOSZYŃSKI

Received April 16th 1963

The method presented replaces the boundary value problem by an initial value problem by means of a linear orthogonal transformation, assuring the 'numerical stability' of the process, as further given in the paper. Methods of a similar kind have been recently published by A.A. Abramov in [2],[3],[4].

1. Introduction

The method of solving boundary value problems for linear ordinary differential equations, based on computation of the so-called 'fundamental system of solutions', is very often 'instable' from the numerical point of view.

The 'fundamental system' can be chosen in various ways and when starting computations it is not certain whether the chosen one is suitable for the given case. The character of the functions forming the 'fundamental system' may be distinctly different from the solution of the boundary value problem, and the resulting linear algebraic system can be illconditioned /cf. [1]/.

Such a case may be illustrated by the following example:

Let's investigate the system of equations

$$\dot{y}_1 = y_2$$

$$\dot{y}_2 = 256\,(y_1 + 1)$$

with boundary conditions

$$y_1(o) + y_1(1) + 2 = 0$$

$$y_2(o) - y_2(1) = 0$$

The solution of the problem is of the form

$$y_1(x) = e^{-16x} - e^{16(x-1)} - 1$$

$$y_2(x) = -16\left(e^{-16x} + e^{16(x-1)}\right).$$

We have

$$e^{-16} - 2 \leqslant y_1(x) \leqslant -e^{-16}$$

$$-16\left(1 + e^{-16}\right) \leqslant y_2(x) \leqslant 0.$$

Choosing a 'fundamental system' for the homogeneous system of equations, so that

$$\bar{y}_1(0) = 1 \qquad \bar{\bar{y}}_1(0) = 0$$

and

$$\bar{y}_2(0) = 0 \qquad \bar{\bar{y}}_2(0) = 1$$

we obtain

$$\bar{y}_1(x) = \frac{1}{2}\left(e^{16x} + e^{-16x}\right)$$

$$\bar{y}_2(x) = 8\left(e^{16x} - e^{-16x}\right)$$

$$\bar{\bar{y}}_1(x) = \frac{1}{32}\left(e^{16x} - e^{-16x}\right)$$

$$\bar{\bar{y}}_2(x) = \frac{1}{2}\left(e^{16x} + e^{-16x}\right).$$

The above functions reach great values in point $x = 1$.

Many significant figures can be lost when using this system to determine the solution of the boundary value problem.

The method given below seems to omit the above-mentioned difficulties. The boundary value problem for functions $y_i(x)$ satisfying the system of equations

$$y_i(x) = \sum_{j=1}^{N} a_{ij}(x)\, y_i(x) + f_i(x) \qquad i = 1.\ 2,\ \ldots,\ N$$

is replaced by such initial value problem for functions $u_i(x)$ satisfying similar system of linear ordinary differential equations, that

$$\sum_{i=1}^{N} y_i^2(x) = \sum_{i=1}^{N} u_i^2(x) .$$

A new system of functions $u_i(x)$ can be obtained by linear transformation of $y_i(x)$. The matrix of this transformation is orthogonal for any x. This last property ensures the 'stability' of the numerical process evaluating the elements of transformation matrix.

2. Orthogonal transformations of the system of linear ordinary differential equations.

Let's consider the linear transformation of the form

$$y_i(x) = \sum_{j=1}^{N} \varphi_{ij}(x)\, u_j(x) \qquad i = 1,\ 2,\ \ldots,\ N \qquad /1/$$

transforming the set of functions $u_i(x)$ into another set of functions $y_i(x)$ for $x \in [a,\ b]$. The functions $y_i(x)$, $\varphi_{ij}(x)$, $u_i(x)$ for $i, j = 1, 2, \ldots, N$ are supposed to be differentiable in $[a,\ b]$.

Assuming the matrix $\Phi(x) = \left(\varphi_{ij}(x)\right)$ for $i, j = 1, 2, \ldots, N$ orthogonal for every $x \in [a,\ b]$, i.e.:

$$\sum_{i=1}^{N} \varphi_{ij}(x)\, \varphi_{ik}(x) = \delta_{jk} \qquad j,k = 1,\ 2,\ \ldots,\ N$$

we obtain

$$\sum_{i=1}^{N}\left[\dot{\varphi}_{ij}\ \varphi_{ik} + \varphi_{ij}\ \dot{\varphi}_{ik}\right] = 0 \qquad j,k = 1, 2, \ldots, N.$$

Denoting $H_{kj}(x) = \sum_{i=1}^{N} \dot{\varphi}_{ij}\ \varphi_{ik}$, we can present the latter system of equations in the form

$$H_{kj}(x) + H_{jk}(x) = 0 \qquad j, k = 1, 2, \ldots, N \qquad /2/$$

The system /2/ contains $\frac{N\ (N-1)}{2} + N$ differential equations for the functions $\varphi_{ij}(x)$. Let's note the following simple fact.

If the matrix $\Phi(x)$, satisfying the system /2/ is orthogonal in a certain fixed point $x \in [a, b]$, it is orthogonal in every $x \in [a, b]$.

We shall apply the transformation /1/ with the orthogonal matrix $\Phi(x)$ to the system of equations

$$\dot{y}_i(x) = \sum_{j=1}^{N} a_{ij}(x)\ y_j(x) + f_i(x) \qquad i = 1, 2, \ldots, N \qquad /3/$$

This gives the new system of linear equations

$$\dot{u}_k(x) = \sum_{j=1}^{N} A_{kj}(x)\ u_j(x) + F_k(x) \qquad k = 1, 2, \ldots, N \qquad /4/$$

where

$$A_{kj}(x) = \Psi_{kj}(x) - H_{kj}(x)$$

$$\Psi_{kj}(x) = \sum_{s=1}^{N} \varphi_{sj}(x) \sum_{i=1}^{N} \varphi_{ik}(x)\ a_{is}(x) \qquad /5/$$

$$F_k(x) = \sum_{i=1}^{N} f_i(x) \quad \varphi_{ik}(x) \qquad /5/$$

Now, it is possible to pose the following problem, basic for further investigations:

For the given system /3/ it is necessary to find such an orthogonal transformation of the form /1/, that

$$A_{kj}(x) = 0 \qquad \text{for } j > k. \qquad /6/$$

Let's solve this problem. First of all /6/ gives

$$H_{kj}(x) = \Psi_{kj}(x) \qquad \text{for } j > k.$$

The orthogonality of the matrix $\phi(x)$ and /2/ give

$$H_{kj}(x) = -\Psi_{jk}(x) \qquad \text{for } j < k$$

$$\Xi_{kk}(x) = 0.$$

Hence, we obtained the following system of N^2 differential equations for the matrix $\phi(x)$

$$\begin{aligned} H_{kj}(x) &= \Psi_{kj}(x) && \text{for } j > k \\ H_{kk}(x) &= 0 && \qquad /7/ \\ H_{kj}(x) &= -\Psi_{jk}(x) && \text{for } j < k \end{aligned}$$

Any solution of /7/ satisfying orthogonal initial conditions is orthogonal in the whole interval. Hence, any solution of the system /7/ with 'orthogonal initial conditions' solves the problem.

Using /7/ we deduce the following formulae for /4/

$$A_{kj}(x) = \begin{cases} 0 & \text{for } j > k \\ \Psi_{kk}(x) & \text{for } j = k \\ \Psi_{kj}(x) + \Psi_{jk}(x) & \text{for } j < k \end{cases} \qquad /8/$$

From /5/ and /7/ we get another system of N^2 equations for $\phi(x)$

$$\dot{\varphi}_{ij} = \sum_{s=1}^{N} \varphi_{sj} \sum_{l=1}^{N} \left\{ \left[\sum_{\alpha=1}^{j-1} \varphi_{i\alpha} \varphi_{l\alpha} \right] a_{ls} - \left[\sum_{\alpha=j+1}^{N} \varphi_{i\alpha} \varphi_{l\alpha} \right] a_{sl} \right\} \qquad /9/^{*)}$$

$i,j = 1, 2, \ldots, N$

Any orthogonal matrix satisfying /7/ satisfies /9/.

Hence, in this case, any solution of /9/ satisfying orthogonal initial conditions solves our problem /if the solution of equations /7/ exists/.

Equations /9/ can be expressed in two equivalent forms convenient for further applications. Using the relation

$$\sum_{\alpha=j+1}^{N} \varphi_{i\alpha} \varphi_{l\alpha} = \delta_{il} - \sum_{\alpha=1}^{j} \varphi_{i\alpha} \varphi_{l\alpha}$$

we obtain

$$\dot{\varphi}_{ij} = \sum_{s=1}^{N} \varphi_{sj} \sum_{l=1}^{N} \left\{ \left[\sum_{\alpha=1}^{j-1} \varphi_{i\alpha} \varphi_{l\alpha} \right] a_{ls} - \left[\delta_{il} - \sum_{\alpha=1}^{j} \varphi_{i\alpha} \varphi_{l\alpha} \right] a_{sl} \right\}. \qquad /10/$$

In a similar way we obtain

$$\dot{\varphi}_{ij} = \sum_{s=1}^{N} \varphi_{sj} \sum_{l=1}^{N} \left\{ \left[\delta_{il} - \sum_{\alpha=j}^{N} \varphi_{i\alpha} \varphi_{l\alpha} \right] a_{ls} - \left[\sum_{\alpha=j+1}^{N} \varphi_{i\alpha} \varphi_{l\alpha} \right] a_{sl} \right\}. \qquad /11/$$

---

* If the upper bound for subscripts is less than the lower one, the sum should be replaced by zero.

Let's observe that in /10/ only columns 1, 2, ..., j^{th} appear in the equations for the j^{th} column of the matrix $\Phi(x)$.

Similarly in /11/ only columns $j, j+1, \ldots, N^{th}$ appear in the equations for the j^{th} column of matrix $\Phi(x)$.

If the condition /6/ is satisfied, the right hand side of the j^{th} equation of /4/ depends only on $u_1, u_2, \ldots, u_j$.

From the orthogonality of the matrix $\Phi(x)$ there follows

$$\sum_{i=1}^{N} u_i^2(x) = \sum_{i=1}^{N} y_i^2(x) \qquad \text{for every} \quad x\,[a, b].$$

This shows 'the numerical character' of the functions $u_i(x)$ and of the solution $y_i(x)$ to be similar. This condition ensures the 'stability' of the computing process if such a stability is possible in the considered case.

For matrices $\left(a_{ij}(x)\right)$ satisfying the condition

$$a_{ij}(x) = -a_{ji}(x)$$

equations /10/ are much simpler

$$\dot{\varphi}_{ij} = \sum_{s=1}^{N} \varphi_{sj} \sum_{l=1}^{N} \left[\delta_{il} - \varphi_{ij}\varphi_{lj}\right] a_{ls}\,.$$

In this case, the right hand sides of equations defining j^{th} column of $\Phi(x)$ depend only on this column.

3. The separated boundary value problem

Let's investigate the problem

$$\dot{y}_i(x) = \sum_{j=1}^{n} a_{ij}(x)\, y_j(x) + f_i(x) \qquad i = 1, 2, \ldots, n \qquad /12/$$

with boundary value conditions

$$\sum_{j=1}^{n} b_{ij}\, y_j(p_i) = c_i \qquad i = 1, 2, \ldots, n \qquad /13/$$

where $p_1 \leq p_2 \leq p_3 \leq \ldots \leq p_n$, b_{ij} and c_i are given numbers.

Using results of preceeding section, we shall define the linear orthogonal transformation of the form /1/ for $N = n$, so as to replace the boundary value problem /12/, /13/ by some initial value problem for the system /4/. It is then necessary to determine suitable initial conditions for /4/ and /10/.

Let's put

$$\varphi_{j1}(p_1) = \frac{b_{1j}}{\sqrt{\sum_{j=1}^{n} (b_{1j})^2}} \qquad j = 1, 2, \ldots, n$$

$$u_1(p_1) = \frac{c_1}{\sqrt{\sum_{j=1}^{n} (b_{1j})^2}} \qquad /14/$$

Using these initial conditions we can determine

- functions $\varphi_{j1}(x)$, $j = 1, 2, \ldots, n$ from /10/
- function $u_1(x)$ from the first equation of /4/.

Now let's assume the functions $\varphi_{\alpha j}(x)$ and $u_j(x)$ for $1 \leqslant j \leqslant i$, $\alpha = 1, 2, \dots, n$ to be already determined from /10/ and /4/.

We shall state the following initial conditions for $\varphi_{\alpha i}(x)$ and $u_i(x)$

$$\varphi_{\alpha i}(p_i) = \frac{z_i}{\sqrt{\sum_{s=1}^{n} z_{is}^2}},$$

$$u_i(p_i) = \frac{v_i}{\sqrt{\sum_{s=1}^{n} z_{is}^2}}, \qquad /15/$$

where

$$z_{is} = b_{is} - \sum_{\alpha=1}^{i-1} \left[\sum_{\beta=1}^{n} b_{i\beta} \varphi_{\beta\alpha}(p_i) \right] \varphi_{s\alpha}(p_i), \qquad s = 1, 2, \dots, n$$

$$v_i = c_i - \sum_{\alpha=1}^{i-1} \left[\sum_{\beta=1}^{n} b_{i\beta} \varphi_{\beta\alpha}(p_i) \right] u_\alpha(p_i).$$

The above formulae define a modified 'Schmidt's Process' of orthogonalization. It is determined if the matrix

$$\begin{bmatrix} \varphi_{1,1}(p_i), & \dots\dots\dots, & \varphi_{n,1}(p_i) \\ \varphi_{1,2}(p_i), & \dots\dots\dots, & \varphi_{n,2}(p_i) \\ \dots & \dots & \dots \\ \varphi_{1,i-1}(p_i), & \dots\dots\dots, & \varphi_{n,i-1}(p_i) \\ b_{i,1}, & \dots\dots\dots, & b_{i,n} \end{bmatrix}$$

is of the rank i for $i = 1, 2, \ldots, n$.

Using initial conditions /15/ we can determine

- functions $\varphi_{ji}(x)$ from equations /10/
- functions $u_p(x)$, $p = 1, 2, \ldots, i$ from the first i equations of /4/.

Following this way, we can determine all functions $\varphi_{ij}(x)$, $i,j = 1, 2, \ldots, n$ and all functions $u_i(x)$, $i = 1, 2, \ldots, n$. Using /1/ we can express functions $y_i(x)$, $i = 1, 2, \ldots, n$. The so obtaind functions satisfy

- equations /12/
- boundary conditions /13/.

In fact, equations /12/ are satisfied. This follows immediately from the definition of matrix $\Phi(x)$ and system /4/.

From /15/ follows

$$\sqrt{\sum_{s=1}^{n} (z_{is})^2} = \sum_{\alpha=1}^{n} b_{i\alpha} \varphi_{\alpha i}(p_i)$$

$$\sum_{\alpha=1}^{n} b_{i\alpha} \varphi_{\alpha\beta}(p_i) = 0 \quad \text{for} \quad \beta > i$$

and

$$c_i = \sum_{\alpha=1}^{n} \left[\sum_{s=1}^{n} b_{is} \varphi_{s\alpha}(p_i) \right] u_\alpha(p_i) .$$

If we replace c_i in /13/ by the above formula, then using /1/, we get

$$\sum_{j=1}^{n} b_{ij} y_j(p_i) = \sum_{j=1}^{n} b_{ij} \sum_{\alpha=1}^{n} \varphi_{j\alpha}(p_i) u_\alpha(p_i) = \sum_{\alpha=1}^{n} u_\alpha(p_i) \sum_{j=1}^{n} b_{ij} \varphi_{j\alpha}(p_i) = c_i$$

for $i = 1, 2, \ldots, n$.

Proposed algorithm

The following algorithm seems to be convenient in practice.

(1) 'Normalize' the first boundary condition from /13/ as in formulae /14/. This gives initial conditions in the point p_1 for $\varphi_{\alpha 1}(x)$, $\alpha = 1, 2, \ldots, n$ and $u_1(x)$.

(2) Integrate from p_1 to p_2 the system of equations formed by
- equations /10/ for $j = 1$
- first equation of /4/.

This gives $\varphi_{\alpha 1}(p_2)$, $\alpha = 1, 2, \ldots, n$ and $u_1(p_2)$.

(3) Using formulae /15/, determine $\varphi_{\alpha 2}(p_2)$, $\alpha = 1, 2, \ldots, n$ and $u_2(p_2)$.

(4) Integrate from p_2 to p_3 the system of differential equations formed by
- equations /10/ for $j = 1$ and $j = 2$
- first and second equation of /4/.

- -

- -

(2n - 2) Using /15/, determine $\varphi_{\alpha,n-1}(p_{n-1})$, $\alpha = 1, 2, \ldots, n$ and $u_{n-1}(p_{n-1})$.

(2n - 1) Integrate from p_{n-1} to p_n the system of equations formed by
- equations /10/ for $j = 1, 2, \ldots, n-1$
- all but the last equations of /4/.

(2n) Using /15/, determine $\varphi_{\alpha,n}(p_n)$, $\alpha = 1, 2, \ldots, n$ and $u_n(p_n)$.

(2n + 1) Using formulae /1/, express $y_i(p_n)$, $i = 1, 2, \ldots, n$. This gives the initial conditions for the system /12/.

The solution of equations /12/ with initial conditions $y_i(p_n)$, determined as in (2n + 1), satisfies boundary conditions /13/.

The case $n = 2$ /see also [2]/

This case is of special interest because of the very simple form of the matrix $\phi(x)$.

Every orthogonal matrix of dimension two is of the form

$$\begin{bmatrix} \varphi, & \psi \\ \psi, & -\varphi \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \varphi, & \psi \\ -\psi, & \varphi \end{bmatrix} .$$

Hence, allways in this case

$$H_{12}(x) + H_{21}(x) = 0.$$

It is then sufficient to determine only two funotions $\varphi(x)$ and $\psi(x)$. Problems /12/, /13/ now take the form

$$\begin{aligned} \dot{y}_1(x) &= a_{11}(x)\, y_1(x) + a_{12}(x)\, y_2(x) + f_1(x) \\ \dot{y}_2(x) &= a_{21}(x)\, y_1(x) + a_{22}(x)\, y_2(x) + f_2(x) \end{aligned} \qquad /16/$$

$$\begin{aligned} b_{11}\, y_1(p_1) + b_{12}\, y_2(p_1) &= c_1 \\ b_{21}\, y_1(p_2) + b_{22}\, y_2(p_2) &= c_2 \end{aligned} \qquad /17/$$

We can assume the boundary conditions to be normalized, i.e., $b_{11}^2 + b_{12}^2 = 1$. If we choose $\phi(x) = \begin{bmatrix} \varphi, & \psi \\ \psi, & -\varphi \end{bmatrix}$, equations /4/ and /10/ will take the following form

$$\begin{aligned} \dot{u}_1(x) &= A_{11}(x)\, u_1(x) + F_1(x) \\ \dot{u}_2(x) &= A_{21}(x)\, u_1(x) + A_{22}(x)\, u_2(x) + F_2(x) \end{aligned} \qquad /18/$$

where

$$A_{11}(x) = \varphi\left[\varphi a_{11} + \psi a_{21}\right] + \psi\left[\varphi a_{12} + \psi a_{22}\right]$$

$$A_{21}(x) = \varphi\left[2a_{11}\psi - \varphi(a_{12} + a_{22})\right] + \psi\left[(a_{12} + a_{21})\psi - 2\varphi a_{22}\right]$$

$$A_{22}(x) = \psi\left[\psi a_{11} - \varphi a_{21}\right] - \varphi\left[\psi a_{12} - \varphi a_{22}\right]$$

$$F_1(x) = f_1(x)\, \varphi(x) + f_2(x)\, \psi(x)$$

$$F_2(x) = f_2(x)\, \varphi(x) - f_1(x)\, \psi(x)$$

and
$$\dot{\varphi} = (\varphi^2 - 1)\left[a_{11}\varphi + a_{21}\psi\right] + \varphi\psi\left[a_{12}\varphi + a_{22}\psi\right]$$
$$\dot{\psi} = \varphi\psi\left[a_{11}\varphi + a_{21}\psi\right] + (\psi^2 - 1)\left[a_{12}\varphi + a_{22}\psi\right]. \qquad /19/$$

We shall further use equations /19/ and only the first equation of /18/.

Let's put
$$\varphi(p_1) = b_{11}$$
$$\psi(p_1) = b_{12} \qquad /20/$$
$$u_1(p_1) = c_1$$

as initial conditions for $\varphi(x)$, $\psi(x)$ and $u_1(x)$.

Integrating /19/ and the first equation of /18/ from p_1 to p_2 using /20/, we can express $y_1(p_2)$ and $y_2(p_2)$ as

$$y_1(p_2) = \frac{\begin{vmatrix} u_1(p_2), & \psi(p_2) \\ c_2, & b_{22} \end{vmatrix}}{\begin{vmatrix} \varphi(p_2), & \psi(p_2) \\ b_{21}, & b_{22} \end{vmatrix}}, \qquad y_2(p_2) = \frac{\begin{vmatrix} \varphi(p_2), & u_1(p_2) \\ b_{21}, & c_2 \end{vmatrix}}{\begin{vmatrix} \varphi(p_2), & \psi(p_2) \\ b_{21}, & b_{22} \end{vmatrix}} \qquad /21/$$

The solution of /16/ with initial conditions /21/ satisfies the boundary problems /16/, /17/.

4. Boundary conditions on both ends of interval. /Non separated boundary conditions/

We shall now investigate the system of linear differential equations in $[a, b]$:

$$\dot{y}_i(x) = \sum_{j=1}^{n} m_{ij}(x)\, y_j(x) + g_i(x) \qquad i = 1, 2, \ldots, n \qquad /22/$$

with the boundary conditions

$$\sum_{j=1}^{n} b_{ij}^{(1)} y_j(a) + \sum_{j=1}^{n} b_{ij}^{(2)} y_j(b) = c_i \qquad i = 1, 2, \ldots, n \qquad /23/$$

Assume the coefficients $b_{ij}^{(1)}$ and $b_{ij}^{(2)}$ to form the $2n \times n$ matrix of the rank n.

Let's denote

$$N = 2n$$

$$b_{ij} = \begin{cases} b_{ij}^{(1)} & \text{for } i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, n \\ b_{ij-n}^{(2)} & \text{for } i = 1, 2, \ldots, n \quad j = n+1, \ldots, N \end{cases}$$

$$y_{i+n}(x) = y_i(a + b - x) \qquad i = 1, 2, \ldots, n \qquad /24/$$

$$a_{ij} = \begin{cases} m_{ij}(x) & \text{for } 1 \leq i,j \leq n \\ -m_{i-n,j-n}(a + b - x) & \text{for } n+1 \leq i,j \leq N \\ 0 & \text{for other values of } i, j \end{cases}$$

$$f_i(x) = \begin{cases} g_i(x) & \text{for } 1 \leq i \leq n \\ -g_{i-n}(a + b - x) & \text{for } n+1 \leq i \leq N \end{cases}$$

We now have the following boundary value problem

$$\dot{y}_i(x) = \sum_{j=1}^{N} a_{ij}(x)\, y_j(x) + f_i(x) \qquad i = 1, 2, \ldots, n \qquad /25/$$

$$\sum_{j=1}^{N} b_{ij}\, y_j(a) = c_i \qquad i = 1, 2, \ldots, n \qquad /26/$$

$$y_i(a) = y_{i-n}(b) \qquad i = n+1, \ldots, N \qquad /27/$$

rom definition /24/ and boundary conditions /27/ we get

$$y_i(b) = y_{i-n}(a) \qquad i = n+1, \ldots, N \qquad /27a/$$

Similarly as in the preceding section, we shall define such orthogonal transformation of /25/ that for the resulting system of equations of the form /4/ conditions /6/ are satisfied. It now remains to put suitable initial conditions for functions $\varphi_{ij}(x)$ and $u_i(x)$.

First of all, let's transform boundary conditions /23/ using the formulae similar to /15/.

$$z_{ij} = b_{ij} - \sum_{s=1}^{i-1}\left[\sum_{l=1}^{N} b_{il}\,\bar{b}_{sl}\right]\bar{b}_{sj}, \qquad z_{1j} = b_{1j}$$

$$i = 1, 2, \ldots, n$$

$$\bar{b}_{ij} = \frac{z_{ij}}{\sqrt{\sum_{s=1}^{N} z_{is}^2}}$$

$$v_i = c_i - \sum_{s=1}^{i-1}\left[\sum_{l=1}^{N} b_{il}\,\bar{b}_{sl}\right]\bar{c}_s \qquad /28/$$

$$\bar{c}_i = \frac{v_i}{\sqrt{\sum_{s=1}^{N} z_{is}^2}}$$

New boundary conditions

$$\sum_{j=1}^{N} \bar{b}_{ij}\, y_j(a) = \bar{c}_i, \qquad i = 1, 2, 3, \ldots, n$$

are equivalent to /26/ and 'orthogonal'. Let's put

$$\varphi_{ij}(a) = \bar{b}_{ji}$$
$$u_j(a) = \bar{c}_j \qquad /29/$$

for $j = 1, 2, \ldots, n$ $i = 1, 2, \ldots, N$.

Using the above initial conditions we can compute

- functions $\varphi_{ij}(x)$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, n$ from the system /10/
- functions $u_j(x)$, $j = 1, 2, \ldots, n$ from the first n equations of the system /4/.

Suppose the values of $\varphi_{ij}(b)$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, n$ and $u_j(b)$, $j = 1, 2, \ldots, n$ to be already computed. We can choose the n remaining columns of the $N \times N$ matrix $\Phi(b)$, so as to get the orthogonal $N \times N$ matrix, and then to integrate the system /11/ from b to a for $j = n+1, \ldots, N$, $i = 1, 2, \ldots, N$ using the defined columns as initial conditions. Hence, we have the matrices $\Phi(a)$ and $\Phi(b)$.

Now we have to determine initial values for $u_{n+1}(x), \ldots, u_N(x)$.

Using /1/ we can rewrite the condition /27/ in the form

$$\sum_{j=n+1}^{N} \varphi_{i,j}(a)\, u_j(a) - \sum_{j=n+1}^{N} \varphi_{i-n,j}(b)\, u_j(b) + r_i = 0 \qquad /30/$$

where

$$r_i = \sum_{j=1}^{n} \left[\varphi_{ij}(a)\, u_j(a) - \varphi_{i-n,j}(b)\, u_j(b)\right] \qquad /30a/$$

$i = n+1, \ldots, N$,

can already be computed.

In a similar way we get from /27a/

$$\sum_{j=n+1}^{N} \varphi_{i-n,j}(a)\; u_j(a) - \sum_{j=n+1}^{N} \varphi_{ij}(b)\; u_j(b) + \rho_i = 0 \qquad /31/$$

where

$$\rho_i = \sum_{j=1}^{n} \left[\varphi_{i-n,j}(a)\; u_j(a) - \varphi_{i,j}(b)\; u_j(b) \right] \qquad /31a/$$

$i = n+1, \ldots, N$

is already known.

Let's divide the matrices $\Phi(a)$ and $\Phi(b)$ into $n \times n$ blocks

$$\Phi(a) = \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array} \right] \qquad \Phi(b) = \left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array} \right]$$

and denote

$$\vec{u}(x) = \begin{bmatrix} u_{n+1}(x) \\ \cdots \\ u_N(x) \end{bmatrix}, \qquad \vec{r} = \begin{bmatrix} r_{n+1} \\ \vdots \\ r_N \end{bmatrix}, \qquad \vec{\rho} = \begin{bmatrix} \rho_{n+1} \\ \vdots \\ \rho_N \end{bmatrix}$$

The equations /30/ and /31/ can now be expressed in the form

$$\begin{aligned} A_4\, \vec{u}(a) &- B_2\, \vec{u}(b) + \vec{r} = 0 \\ A_2\, \vec{u}(a) &- B_4\, \vec{u}(b) + \vec{\rho} = 0 \end{aligned} \qquad /32/$$

It is a system of N linear algebraic equations. The solution is $\vec{u}(a)$ and $\vec{u}(b)$ – the initial values we were looking for.

Using $u(a)$ and /1/ we can express $y_i(a)$ from the formula

$$y_i(a) = \sum_{j=1}^{n} \varphi_{ij}(a)\, c_j + \sum_{j=n+1}^{N} \varphi_{ij}(a)\, u_j(a) \qquad /33/$$

$i = 1, 2, \ldots, N$

These are the initial conditions for functions $y_i(x)$, $i = 1, 2, \ldots, n$ solving the problem /25/, /26/, /27/. Similarly, $\bar{u}(b)$ can be used to express initial values of $y_i(x)$ in point b.

The first n functions $y_1(x)$, $y_2(x)$, ..., $y_n(x)$ solve the original problem /22/, /23/.

Because of relations

$$y_{i+n}(x) = y_i(a + b - x) \qquad i = 1, 2, \ldots, n$$

the remaining n functions $y_{n+1}(x)$, ..., $y_N(x)$ do not spoil the stability of the procedure.

Proposed algorithm

(1) 'Orthogonalize' boundary conditions /23/ or /26/ using formulae /28/. Store results as matrices A_1 and A_3.

(2) Integrate the system /10/ from a to b for $j = 1, 2, \ldots,$ using matrices A_1 and A_3 as the initial conditions in a. Solve together with /10/ the first n equations of /4/ using initial conditions /29/.

(3) Choose the matrices B_2 and B_4 so as to get the matrix $\left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array}\right]$ orthogonal.

(4) Using columns $\left[\begin{array}{c} B_2 \\ \hline B_4 \end{array}\right]$ as initial values for φ_{in+1}, ..., φ_{iN} in b, integrate the system /11/ from b to a. Store the results as A_2 and A_4.

(5) Compute vectors $\bar{r}$ and $\bar{\beta}$. /Formulae /30a/ and /31a/ /.

(6) Solve the system /32/ of linear algebraic equations for the values $u_{n+1}(a)$, ..., $u_N(a)$.

(7) Use formulae /33/ to express initial values $y_1(a)$, ..., $y_n(a)$ for the system /22/.

(8) Integrate the system /22/ using initial values obtained in point /7/ of this algorithm. This is the solution of the problem /22/, /23/.

References

1. BEREZIN I.S., ŽIDKOV N.P.: Metody Vyčislenij, 1960:2.
2. ABRAMOV A.A.: Variant metoda progonki, Žurnal Vyčislitelnoj Matematiki i Matematičeskoj Fiziki, 1961:1, 2, 349.
3. ABRAMOV A.A.: O perenose graničnych uslovij dla sistem liniejnych obyknovennych differencjalnych uravnenij, Žurnal Vyčislitelnoj Matematiki i Matematičeskoj Fiziki, 1961:1, 3, 542.
4. ABRAMOV A.A.: O primenenii metoda progonki k nachoždeniju periodičeskich rešenij differencjalnych i raznostnych uravnenij, Žurnal Vyčislitelnoj Matematiki i Matematičeskoj Fiziki, 1963:3, 2, 377.

STATISTICAL AND OTHER APPLICATIONS

Instytut Maszyn Matematycznych PAN
Algorytmy N° 3

ON A CERTAIN GENERALIZATION OF THE DIVISION WITH REMAINDER AND ITS APPLICATION

by Roman RĘDZIEJOWSKI

Received September 1962

The concepts of the integral quotient and remainder are often used when considering the number expansion with a positive base $b>1$. An attempt is made to generalize these concepts in order to use them in the case of an arbitrary integral base $|b|>1$. This permits to extend some relations and procedures known for $b>1$ to this general case and to solve certain practical problems concerning computers with a negative base of expansion.

1. Introduction

Some important relations exist between the division with remainder and the number expansion in the case of a positive base. The problem considered in this paper in to generalize the concepts of the integral quotient and remainder in order to extend these relations to the case of an arbitrary integral base.

2. A generalized division with remainder

The integral quotient and remainder are defined as follows [1]: For any natural number y and an integral number $x \geqslant 0$ there exists exactly one pair of integers $q(x, y)$, $r(x, y)$ satisfying conditions

$$x = y \cdot q(x, y) + r(x, y), \qquad /1/$$

$$0 \leqslant r(x, y) < y. \qquad /2/$$

This implies the following more general theorem.

Theorem 1

For any three integers x, y, t, such that $y \neq 0$, there exists exactly one pair of integers $Q(x, y, t)$, $R(x, y, t)$ satisfying conditions

$$x = y \cdot Q(x, y, t) + R(x, y, t), \qquad /3/$$

$$0 \leqslant R(x, y, t) - t < |y|. \qquad /4/$$

Proof

Let x, y, t be integers and $y \neq 0$. Assume integer n to be chosen so that $x + ny - t \geqslant 0$. Let A, B denote

$A = q(x + ny - t, |y|)$, $B = r(x + ny - t, |y|)$.

According to /1/, /2/ there is

$$x + ny - t = |y| \cdot A + B,$$

$$0 \leqslant B < |y|;$$

from this we obtain

$$x = y \cdot (\operatorname{sgn} y \cdot A - n) + (B + t),$$

$$0 \leqslant (B + t) - t < |y|.$$

Thus, the numbers

$$Q(x, y, t) = \operatorname{sgn} y \cdot A - n = \operatorname{sgn} y \cdot q(x + ny - t, |y|) - n, \qquad /5/$$

$$R(x, y, t) = B + t = r(x + ny - t, |y|) + t \qquad /6/$$

satisfy conditions /3/, /4/.

Suppose there exists another pair $Q'(x,y,t)$, $R'(x,y,t)$ satisfying these conditions. Let $k = Q'(x,y,t) - Q(x,y,t)$, (k be an integer). Applying /3/ we obtain $R'(x, y, t) = R(x, y, t) + ky$; according to /4/ there is

$$0 \leqslant R(x, y, t) - t < |y|$$

and

$$0 \leqslant R(x, y, t) + ky - t < |y|;$$

this implies

$$-|y| < -(R(x,y,t) - t) \leqslant ky < |y| - (R(x,y,t) - t) \leqslant |y|;$$

thus $-|y| < ky < |y|$.
Hence, $-1 < k < 1$ and $k = 0$. This proves the existence and the uniqueness of $Q(x, y, t)$, $R(x, y, t)$.

Let's note that for $x \geqslant 0$, $y > 0$ and $t = 0$ there is

$$Q(x, y, t) = q(x, y),$$
$$R(x, y, t) = r(x, y),$$

$Q(x, y, t)$ and $R(x, y, t)$ being a generalization of the integral quotient and remainder. Some applications of the above functions are given below.

3. Proof of the existence of number expansion

The existence of number expansion was proved separately and in a different way for two cases: the case of a positive base $b > 1$ [1], [2] and the case $b < -1$ [3], [4]. The same can be proved for both above-mentioned cases together as follows:

Theorem 2

For any pair of integers a, b such that

$b < -1$ and arbitrary a, or
$b > 1$ and $a \geqslant 0$

there exists exactly one infinite sequence of integers c_i $(i = 0, 1, 2, \dots)$ such that

$$a = c_0 + c_1 b + c_2 b^2 + \dots \qquad /7/$$

and

$$0 \leqslant c_i < |b| \qquad \text{for } i = 0, 1, 2, \dots \qquad /8/$$

/this sequence is called e x p a n s i o n o f n u m b e r a w i t h b a s e b/.

P r o o f

Let a, b – be numbers as assumed in Theorem 2. Suppose there exists the expansion of number a with base b; then, for any $m \geqslant 0$ /7/ may be written as follows

$$a = b^m(c_m + c_{m+1}b + c_{m+2}b^2 + \dots) + \\ + (c_0 + c_1 b + c_2 b^2 + \dots + c_{m-1}b^{m-1}). \qquad /9/$$

The inequality /8/ gives

$$0 \leqslant (c_0 + c_1 b + c_2 b^2 + \dots + c_{m-1}b^{m-1}) - s_m < |b^m|, \qquad /10/$$

where s_m is the sum of all negative terms of the sequence $\{b^j(|b| - 1)\}$ for $0 \leqslant j < m$ if such terms exist; otherwise $s_m = 0$.

According to Theorem 1 the above sums are equal to

$$(c_m + c_{m+1}b + c_{m+2}b^2 + \dots) = Q(a, b^m, s_m), \qquad /11/$$

$$(c_0 + c_1 b + c_2 b^2 + \dots + c_{m-1}\, b^{m-1}) = R(a, b^m, s_m); \qquad /12/$$

the element c_m of the expansion can be expressed as

$$c_m = (c_m + c_{m+1}b + c_{m+2}b^2 + \dots) - b(c_{m+1} + c_{m+2}b + \dots) = \\ = Q(a, b^m, s_m) - b \cdot Q(a, b^{m+1}, s_{m+1}). \quad /13/$$

Thus, if the expansion exists, its element c_m is unique for any $m \geqslant 0$.

The following lemma is needed to prove the existence of the expansion.

Lemma 1

For any pair a, b satisfying the conditions of Theorem 2 there exists such $m \geqslant 0$ that $Q(a, b^m, s_m) = 0$, s_m being defined as in /10/.

Proof

Let a, b - be numbers as assumed in Theorem 2; it will be shown that the pair $Q(a, b^m, s_m) = 0$, $R(a, b^m, s_m) = a$ satisfy conditions /3/, /4/ for certain $m \geqslant 0$, i.e.

$$a = b^m \cdot 0 + a$$

$$0 \leqslant a - s_m < |b^m|.$$

The first condition is satisfied for any m; it can be easily seen that the second condition is satisfied for m being great enough. Let us consider the two following cases:

1. $a \geqslant 0$, $b > 1$. Let m be an integer such that $b^m > a$. In this case $s_m = 0$. Thus, $0 \leqslant a - s_m < |b^m|$;

2. $b < -1$. Let m be an even integer such that $b^{m-2} > |a|$; this implies $b^{m-1} < a$. In this case

$$s_m = (b^{m-1} + b^{m-3} + \dots + b)(-b - 1) =$$

$$= -(b^m + b^{m-1} + \dots + b + 1) < b^{m-1} < a;$$

$$s_m + |b^m| = -(b^{m-1} + b^{m-2} + \dots + b + 1) > b^{m-2} > a.$$

Thus, $0 < a - s_m < |b^m|$.

This proves Lemma 1.

Let a, b - be numbers as assumed in Theorem 2; let $m \geq 0$ be an integer such that $Q(a, b^m, s_m) = 0$.

Let c_i' $(i = 0, 1, 2 \dots)$ be the following infinite sequence of integers:

$$c_i' = \begin{cases} Q(a, b^i, s_i) - b\,Q(a, b^{i+1}, s_{i+1}) & \text{for } 0 \leq i < m, \\ 0 & \text{for } i \geq m \end{cases} \qquad /14/$$

Let's note that

1. For any $i \geq 0$ the element c_i' has the property /8/. For $i \geq m$ it is obvious; for $0 \leq i < m$ using /3/ we have

$$c_i' b^i = (Q(a, b^i, s_i) - b\,Q(a, b^{i+1}, s_{i+1}))\, b^i =$$

$$= R(a, b^{i+1}, s_{i+1}) - R(a, b^i, s_i).$$

When applying /4/, we obtain the inequality /8/.

2. The sequence /14/ has the property /7/ as

$$c_0 + c_1 b + c_2 b^2 + \ldots = Q(a, b^0, s_0) - b\,Q(a, b^1, s_1) +$$
$$+ b\,Q(a, b^1, s_1) - b^2 Q(a, b^2, s_2) + \ldots +$$
$$+ b^{m-1} Q(a, b^{m-1}, s_{m-1}) - b^m Q(a, b^m, s_m) + 0 =$$
$$= Q(a, b^0, s_0)$$

and

$$Q(a, b^0, s_0) = a. \qquad /15/$$

/15/ results from $s_0 = 0$ and from the pair $Q(a, b^0, s_0) = a$, $R(a, b^0, s_0) = 0$ satisfying conditions /3/, /4/.

Hence, sequence /14/ is the expansion of number a with base b. As already shown, this expansion is unique. Thus, Theorem 2 is proved.

Let's note that for $b>1$, $a\geq 0$ expressions /11/, /12/, /13/ take the well known form [1]

$$(c_m + c_{m+1} b + c_{m+2} b^2 + \ldots) = q(a, b^m),$$
$$(c_0 + c_1 b + c_2 b^2 + \ldots + c_{m-1} b^{m-1}) = r(a, b^m),$$

and [2] $c_m = q(a, b^m) - bq(a, b^{m+1})$.

4. The method of successive divisions in the case $b < -1$

The evaluation of expansion of a given number with a given base is often needed in practice. Expression /13/ may be used to this purpose; another method is given below, which may be used as well.

Let a_i denote $Q(a, b^i, s_i)$. The following properties of a_i

can be found from /11/, /12/, /13/

$$c_i = R(a_i, b, s_i), \qquad /16/$$

$$a_{i+1} = Q(a_i, b, s_i). \qquad /17/$$

These expressions are the i-th step of the procedure to be applied: c_i and a_{i+1} are evaluated for the already known a_i. For step 0 we have $a_o = a$ /see /15//. According to Lemma 1, for a certain step m there is $a_m = 0$; the procedure is then finished, as $c_i = 0$ for all $i \geqslant m$ /see /14//.

There is $s_1 = 0$; let's notice that for $a \geqslant 0$, $b > 1$ expressions /16/, /17/ take the form

$$c_i = r(a_i, b),$$

$$a_{i+1} = q(a_i, b),$$

and the procedure becomes the well known method of successive divisions by the value of base [2]. For $b < -1$ the described procedure is identic with that given in [4].

For the practical use in the case $b < -1$ a modification is to be made.

Let n be an integer such that $nb \geqslant |a|$. It can be easily seen that there is $a_i + nb \geqslant 0$ for any $i \geqslant 0$. Applying /5/, /6/ we have

$$c_i = r(a_i + nb, |b|),$$

$$a_{i+1} = -q(a_i + nb, |b|) - n;$$

after substituting $a_i' = a_i + nb$ the above procedure becomes the following:

1. starting value $a_o' = a + nb$,

2. i-th step $c_i = r(a'_i, |b|)$, $a'_{i+1} = -q(a'_i, |b|) + n(b - 1)$,

3. procedure is finished if $a'_i = nb$.

5. Evaluation of the round-off error

The expansion of a number, being the infinite sequence, cannot be fully represented in an actual computer. Its certain elements may be represented only when remainings are conventionally assumed to be 0. Usually the number a' is generated if a certain number a is to be represented in such a computer; expansion c'_i of a' is such that $c'_i \neq 0$ only for elements represented in this computer /the difference a' - a is called round - off error/.

Number a' is usually so defined that its expansion c'_i is obtained from the expansion c_i or a as follows

$$c'_i = \begin{cases} c_i & \text{for } u \leqslant i < v, \\ 0 & \text{for } i < u \text{ or } i \geqslant v, \end{cases} \qquad /18/$$

where u, v are typical numbers of the given computer, and $0 \leqslant u < v$. For given a, b, u, v, one may evaluate a' without evaluating the expansion.

From /18/

$$a' = c_u b^u + c_{u+1} b^{u+1} + \dots + c_{v-1} b^{v-1} =$$

$$= (c_0 + c_1 b + \dots + c_u b^u + \dots + c_{v-1} b^{v-1}) +$$

$$- (c_0 + c_1 b + \dots + c_{u-1} b^{u-1})$$

thus

$$a' = R(a, b^v, s_v) - R(a, b^u, s_u). \qquad /19/$$

Let's notice that for $a \geqslant 0$, $b > 1$ the above expression takes the well known form

$$a' = r(a, b^v) - r(a, b^u).$$

In the case $b < -1$, /6/ is to be substituted.

6. Conclusion

Introducing the generalized concepts of the integral quotient and remainder, some relations and procedures being already known for $b > 1$ were extended to the general case of the number expansion with an integral base $|b| > 1$. Also, some features common for both cases $b > 1$ and $b < -1$ were found.

Acknowledgement

The author would like to thank Mr J. Swianiewicz and Mr T. Kulikowski for valuable remarks.

References

1. SIERPIŃSKI W.: Arytmetyka teoretyczna, PWN, Warszawa 1955:30-33.
2. SIERPIŃSKI W.: Teoria liczb, Monografie Matematyczne, Warszawa-Wrocław 1950:205-207.
3. WAKULICZ A., PAWLAK Z.: Bull. Acad. Polonaise des Sc., Cl.III, 1957:5, 223-226.
4. SIERPIŃSKI W.: Teoria liczb, cz. II, PWN, Warszawa 1959:304-305.

Instytut Maszyn Matematycznych PAN
Algorytmy Nº 3
© 1964.03

518.6:621-232

O ZASTOSOWANIU PEWNEJ METODY PROGRAMOWANIA WYPUKŁEGO DO SYNTEZY MECHANIZMÓW

Jan GOLIŃSKI

Pracę złożono 6.05.1963 r.

Podano zastosowanie pewnej metody programowania wypukłego do rozwiązywania zagadnień z zakresu syntezy mechanizmów.

Przedmiotem rozważań będzie zastosowanie pewnej metody przybliżonego znajdowania ekstremów dla zagadnień z zakresu syntezy maszyn i mechanizmów.

Konstruktor projektując maszynę lub mechanizm otrzymuje założenia do zadania w postaci danych, założonych ograniczeń oraz wymagań, jakie winny spełniać obliczane parametry. Ograniczenia te mogą być różne, np. mogą dotyczyć rozmiarów, sił, przyśpieszeń, warunków montażowych, technologicznych, estetycznych itd. Projektujący dokonuje zwykle kilku prób odrzucając warianty, które nie spełniają określonych warunków typu nierówności. Wreszcie, z obliczonych wariantów spełniających wszystkie złożone warunki, wybiera jeden, lub kierując się określoną zasadą ponawia obliczenia, starając się poprawić parametry projektowanej maszyny zgodnie z przyjętym kryterium.

Zadanie tego rodzaju można sformalizować następująco:
Wynik syntezy maszyny lub mechanizmu da się przedstawić w postaci wektora

$$x \equiv \left[x_i \right],$$

gdzie składowe oznaczają wielkości konstrukcyjne, parametry technologiczne, elektryczne, akustyczne itd. Jak wspomniano, konstruktor nie ma zupełnej swobody przy określaniu tych parametrów. Jest on ograniczony różnymi więzami. Poszczególne więzy można przedstawić w postaci nierówności

$$\varphi_i(x) \geqslant 0 \qquad (i = 1, 2, 3, \ldots, n) \qquad /1/$$

Zadaniem konstruktora maszyny lub mechanizmu jest wybór tego rozwiązania, które wynika z przyjętego kryterium zadania. Cel ten można opisać funkcją optimum

$$f(x) = \text{optimum}. \qquad /2/$$

Funkcja $f(x)$ może wyrażać np. ciężar urządzenia, jego sprawność, przyśpieszenie itd. Rozwiązaniem zadania jest wektor x spełniający warunki /1/ tak, aby funkcja /2/ osiągnęła właściwe extremum.

Stwierdzić należy, że tak sformułowane zadanie można rozwiązać jedną z metod badań operacyjnych. Zastosowaną metodę [3] krótko opiszemy.

Założenia metody.

E^m stanowi euklidesową m wymiarową przestrzeń wektorową, a f oraz φ_i $(i = 1, 2, 3, \ldots, n)$ rzeczywiste, ciągłe funkcje wklęsłe. Zadanie polega na znalezieniu maksimum warunkowego funkcji f w E^m ograniczonej więzami /1/. Można udowodnić [3], że rozwiązać je można przez znalezienie wektora x będącego granicą przy μ malejącym i dążącym do zera wektorów x_μ, w których funkcje G_μ, gdzie

$$G_{\mu}(x) = \mu \cdot f(x) + \frac{1}{2}\sum_{i=1}^{m} s(\varphi_i(x)) \cdot \varphi_i(x),$$

$$s(\varphi_i(x)) = \begin{cases} -\varphi_i(x) & \text{dla} \quad \varphi_i(x) < 0 \\ 0 & \text{dla} \quad \varphi_i(x) \geqslant 0 \end{cases}$$

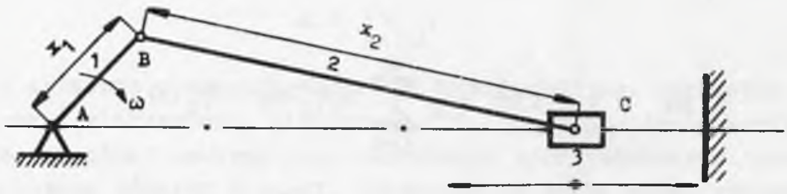
przyjmują maksimum bezwarunkowe w E^m. Problem sprowadza się w ten sposób do wielokrotnego znalezienia maksimum bezwarunkowego funkcji w E^m jedną ze znanych metod.

Przy rozwiązaniu zadania, przedstawionego w tym opracowaniu, zastosowano metodę najszybszego spadku. W metodzie tej za punkt początkowy wybieramy dowolny punkt $P(x_0, y_0)$ na obszarze. Kierunkiem największego spadku jest kierunek wektora gradientu $\left(-\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial y}\right)_0$. Nowy punkt P_1 określa się z danego punktu P_0, zaś proces powtarzamy tak długo, dopóki nie osiągniemy optimum w punkcie P_m. Współrzędne kolejnego punktu P_1 obliczamy z zależności

$$x_1 = x_0 - h\left(\frac{\partial f}{\partial x}\right)_0, \qquad y_1 = y_0 - h\left(\frac{\partial f}{\partial y}\right)_0.$$

Oczywiste jest, że przez zmianę znaków i kierunków nierówności, właściwość wklęsłości zmieni się na właściwość wypukłości i problem maksymalizacji przejdzie na problem minimalizacji. Fakt ten wykorzystano przy układaniu programu. Zasadę postępowania przy korzystaniu z opisanych metod i sposób programowania tego typu zadań pokażemy na bardzo prostym i powszechnie znanym mechanizmie. Zadanie to rozwiązano analitycznie i wykreślnie [2] i stąd znane jest dokładnie rozwiązanie.

Należy dokonać syntezy symetrycznego mechanizmu korbowo-wodzikowego, którego schemat pokazano na rys.1.



Rys. 1. Symetryczny mechanizm korbowo-wodzikowy
1- korba, 2- łącznik, 3- wodzik, x_1 i x_2 - odpowiednio poszukiwane długości korby i łącznika.

Więzy wynikają z następujących przyjętych warunków kinematycznych:
- prędkość V_B punktu B stała i równa 2,5 msek^{-1};
- największa wartość bezwzględna przyśpieszenia P_c punktu c nie większa od 1200 msek^{-2};
- największa wartość bezwzględna przyśpieszenia kątowego łącznika CB nie większa od 42500 sek^{-2};
- stosunek $\frac{x_1}{x_2}$ nie większy od 0,2 i nie mniejszy od 0.6.

Oznaczając przez ω prędkość kątową korby AB i korzystając ze znanych związków:

$$\omega = \frac{V_B}{x_1}$$

$$(P_c)_{max} = \frac{x_1 \cdot \omega^2}{1 + x_1 \cdot x_2}$$

$$(\varepsilon)_{max} = \omega^2 \frac{x_1}{\sqrt{x_2^2 - x_1^2}}$$

otrzymamy po przekształceniach /przyjmując jako jednostki sek.i om/:

$$\begin{aligned}
&\varphi_1(x) = 1.92 \cdot x_1 \cdot x_2 - x_1 - x_2 \geqslant 0; \\
&\varphi_2(x) = x_1^2 \cdot (x_2^2 - x_1^2) - (1.47)^2 \geqslant 0; \\
&\varphi_3(x) = x_1 - 0.2 \cdot x_2 \geqslant 0; \\
&\varphi_4(x) = 0.6 \cdot x_2 - x_1 \geqslant 0; \\
&x_1 > 0; \\
&x_2 > 0.
\end{aligned} \qquad /1'/$$

Zakładając, że np. masy obu szukanych członów są proporcjonalne do kwadratów ich długości oraz, że odpowiednimi współczynnikami są 4 i 1, otrzymamy funkcję kryterium w następującej postaci:

$$f(x) = 4x_1^2 + x_2^2. \qquad /2'/$$

Celem jest więc określenie takich parametrów mechanizmu, które, spełniając założone warunki typu nierówności, pozwalają na uzyskanie najmniejszej masy urządzenia, równej $|f(x)|$.

Zadanie opisane zależnościami /1'/ i /2'/ przygotowano do obliczeń na maszynie cyfrowej. Załączona sieć działań i program w języku SAKO [1] ilustrują przebieg obliczeń, które wykonano na maszynie ZAM-2. Wzmianki wymagają pewne trudności przygotowania zadania do maszynowego obliczenia. Istotnym jest właściwy dobór jednostek i ustalenie skali dla aktualnie wykorzystanych obliczeń.

Korzystny jest wypadek, kiedy jednostki są tak dobrane,że wszystkie parametry są wielkościami bliskich sobie rzędów. W praktyce technicznej zdarza się to jednak rzadko. Należy wówczas szukać drogi pozwalającej na dokonanie obliczeń w takiej skali, która nam zapewni wystarczającą dokładność i która jednocześnie nie przekracza zakresu maszyny.

Załączony przykładowo program składa się z 3 rozdziałów:

Rozdział 0: jest krótkim rozdziałem wprowadzającym dane.
Rozdział 1: w rozdziale tym dokonuje się wszystkich obliczeń związanych z zadaniem.
Rozdział 2: jest rozdziałem wyprowadzającym, drukującym wyniki.

Dla obszerniejszych zagadnień technicznych /większa ilość zmiennych i warunków/ nie udaje się na ogół pomieścić całości obliczeń w jednym rozdziale /ograniczeniem wielkości rozdziału jest pojemność pamięci szybkiej/. Pamiętać należy, że rozbicie zadania na rozdziały przedłuża czas obliczenia. Przykładowo podano czas potrzebny do obliczenia omawianego przykładu na maszynie ZAM-2:

obliczenie - 1 minuta
drukowanie wyników - 2 minuty.

Podkreślmy, że przedstawiony przykład, nadzwyczaj prosty technicznie, ma jedynie na celu pokazanie, jak należy korzystać z jednej z metod programowania nieliniowego.

Użyteczność metody jest szczególnie wyraźna przy większych zagadnieniach technicznych z większą ilością zmiennych i ograniczeń.

Załączony program ma uniwersalne zastosowanie do wszystkich zadań tego typu. 'Układ' programu i jego sieć działań pozostaje w zasadzie ta sama. Zmianie ulegają jedynie dołączane podprogramy.

Objaśnienia do arkusza wydawniczego.

X_1, X_2 - oznaczają poszukiwane długości członów,

F - oznacza wartość funkcji w punkcie $/X_1, X_2/$,

E_1, E_2, E_3, E_4 - oznaczają warunki. Niespełnienie warunków wskazuje gwiazdka. Przy zmianie DEL /μ ze wzoru /1//, następuje drukowanie wartości E w danym punkcie.

Program jest tak zbudowany, że wyszukiwanie ekstremum odbywa się niejako automatycznie. Kolejna zmiana wartości kroku H i współczynnika DEL /μ w wyrażeniu na $G_\mu(x)$/ następuje samoczynnie przez sprawdzenie warunków:

- Warunek sprawdzający czy należy zmienić krok gradientu:
  $WP_1 \times W_1 + WP_2 \times W_2 > 0.0001$,

gdzie:

W_1 i WP_1 wartości pochodnych cząstkowych $\frac{\partial G_\mu(x)}{\partial X_1}$ w dwóch kolejno po sobie następujących krokach.

W_2 i WP_2 wartości pochodnych cząstkowych $\frac{\partial G_\mu(x)}{\partial X_2}$ w dwóch kolejno po sobie następujących krokach.

tzn. jeżeli suma iloczynów wartości gradientu w dwóch kolejnych "krokach" jest większa od określonej tolerancji, wówczas wykonany zostanie kolejny rozkaz. W przeciwnym razie ulega zmianie 'krok'. /Zmiana 'kroku' gradientu następuje przez podzielenie przez dwójkę/.

- Warunek sprawdzający czy należy zmienić DEL

$$0.0001 > H^2 \times \left(WP_1^{\,2} + WP_2^{\,2}\right)$$

Istnieje tutaj wyraźna interpretacja geometryczna. Jeżeli kwadrat wypadkowej, otrzymany z sumowania kwadratów kroku gradientowego wzdłuż obu osi współrzędnych, jest mniejszy niż określona tolerancja, wówczas wykonuje się następny rozkaz. W przeciwnym wypadku ulega zmianie DEL przez podzielenie przez dwójkę. Tak zrealizowane sterowanie przyśpiesza obliczenie ekstremum i jednocześnie zapobiega przypadkowi, że po kolejnym 'kroku' znajdziemy się daleko poza obszarem dopuszczalnych decyzji. Po każdej zmianie DEL program powoduje sprawdzenie warunków, zaś 'wydawnictwo' przez drukowanie gwiazdek sygnalizuje ich niespełnienie. Jeżeli to niespełnienie mieści się w narzuconych tolerancjach, zadanych programem, wówczas warunek traktujemy jako spełniony. Ostatnie obliczone wartości $(X_1, X_2) = (0.827, 1.96)$ stanowią poszukiwane współrzędne, dla których funkcja kryterium przyjmuje wartość 6.572. Dla tego punktu warunki E_1, E_3 i E_4 są spełnione, natomiast niespełnienie $E_2 = -0.004$, co mieści się w tolerancji = -0.006, z góry określonej w programie.

Literatura

1. ŁUKASZEWICZ L., MAZURKIEWICZ A.: System Automatycznego Kodowania SAKO. Warszawa. Ossolineum 1963.

2. ODERFELD J.: Programowanie w budowie maszyn, Archiwum Budowy Maszyn, 1962:9,4.

3. PIETRZYKOWSKI T.: On a Method of Approximative Finding Conditional Maximum, Algorytmy, 1962:1,1,9.

ON APPLICATION OF A CERTAIN CONVEX PROGRAMMING METHOD FOR SYNTHESIS OF MECHANISMS

Summary

The paper deals with a synthesis of plane mechanisms under certain conditions of velocity and acceleration, with minimum weight of the mechanism. The method used shows the possibility of reducing the constrained maximum to the unconditional maximum of a certain function.

```
ROZDZIAL:0
SKALA DZIESIETNA PARAMETROW:4
USTAW SKALE DZIESIETNIE:4
CZYTAJ:X1,X2, DEL,H1
STOP NASTEPNY
PISZ NA BEBEN OD 0:X1,X2,DEL,H1
IDZ DO ROZDZIALU:1


ROZDZIAL:1
CALKOWITE:ADRES
CZYTAJ Z BEBNA OD 0:X1,X2,DEL,H1
USTAW SKALE DZIESIETNIE:4
SKALA DZIESIETNA PARAMETROW:4
ADRES=4
3X)H=H1
(WP1, WP2,E1,E2,E3,E4)=W(X1,X2,DEL)
4X)(W1,W2,E1,E2,E3,E4)=W(X1+H×WP1,X2+H×WP2,DEL)
GDY BYL NADMIAR:1N,INACZEJ NASTEPNY
GDY WP1×W1+WP2×W2 >0.0001:NASTEPNY,INACZEJ 2X
GDY 0.0005 >H×(ABS(WP1)+ABS(WP2)) :1X,INACZEJ NASTEPNY
X1=X1+H×WP1
X2=X2+H×WP2
F=F(X1,X2)
PISZ NA BEBEN OD ADRES:X1,X2,F,E1,E2,E3,E4,DEL
ADRES=ADRES+8
GDY ADRES >100:NASTEPNY,INACZEJ 3X
PISZ NA BEBEN OD 0:X1,X2,DEL
IDZ DO ROZDZIALU:2
2X)H=H/2
GDY 0.0005 >H×(ABS(WP1)+ABS(WP2)) :NASTEPNY,INACZEJ 4X
1X)DEL=DEL/2
GDY DEL=0:NASTEPNY,INACZEJ 3X
2N)ADRES=ADRES+7
KA=999.
PISZ NA BEBEN OD ADRES:KA
IDZ DO ROZDZIALU:2
1N)TEKST:
NADMIAR
```

```
LINIA
SKOCZ DO2N
PODPROGRAM:(W1,W2,E1,E2,E3,E4)=W(X1,X2,DEL)
E1=1.92xX1xX2-X1-X2
W1=0
W2=0
GDY E1 >0:1A,INACZEJ NASTEPNY
W1=(1.92xX2-1)xE1
W2=(1.92xX1-1)xE1
1A)E2=X1*2 x(X2*2-X1*2)-1.47*2
GDY E2> 0:1B,INACZEJ NASTEPNY
W1=W1+2xX1 x(X2*2-2xX1*2)xE2
W2=W2+2xX1*2xX2xE2
1B)E3=X1-0.2xX2
GDY E3>0:1C,INACZEJ NASTEPNY
W1=W1+E3
W2=W2-0.2xE3
1C)E4=0.6xX2-X1
GDY E4>0:1D,INACZEJ NASTEPNY
W1=W1-E4
W2=W2+0.6xE4
1D)GDY X1 >0:1E,INACZEJ NASTEPNY
W1= W1+X1
1E)GDY X2>0:1F,INACZEJ NASTEPNY
W2=W2+X2
1F)W1=W1+8xX1xDEL
W2=W2+2xX2xDEL
WROC
PODPROGRAM:F(X1,X2)
F=4xX1*2+X2*2
WROC

ROZDZIAL:2
CALKOWITE:ADRES,I,J
SKALA DZIESIETNA PARAMETROW:4
USTAW SKALE DZIESIETNIE:4
BLOK(3):E
```

```
TABLICA(3):TOL
-0.003
-0.006
-0.001
-0.001
LINII 2
SPACJI 6
TEKST:
      X1      X2      F      E1      E2      E3      E4      DEL
*)LINIA
CZYTAJ Z BEBNA OD ADRES:DELP,X1,X2,F,*E,DEL
GDY DEL=999.:3,INACZEJ NASTEPNY
DRUKUJ(4.3):X1,X2,F
*)GDY E(1)>TOL(1):11,INACZEJ NASTEPNY
SPACJI 3
TEKST:
*
SKOCZ DO1
11)SPACJA 4
1) POWTORZ:1=0(1)3
GDY DELP=DEL:2,INACZEJ NASTEPNY
DRUKUJ(4.6):DEL
3)LINII 2
TEKST:
E:
DRUKUJ(5.3):E(0),E(1),E(2),E(3)
LINII 2
GDY DEL=999.:4,INACZEJ NASTEPNY
2) POWTORZ:ADRES=3(8)91
GDY KLUCZ 1:4,INACZEJ 5
4)STOP NASTEPNY
IDZ DO ROZDZIALU:0
5)IDZ DO ROZDZIALU:1
KONIEC:0
```

	X_1	X_2	F	E_1	E_2	E_3	E_4	DEL
	+0.608	+1.990	+5.440	*	*			+0.400000
E:	-0.274	-0.833		+0.210		+0.586		
	+0.660	+1.982	+5.673	*	*			
	+0.702	+1.960	+5.809	*	*			
	+0.689	+1.903	+5.516	*	*			
	+0.712	+1.886	+5.590	*	*			
	+0.711	+1.822	+5.342	*	*			
	+0.724	+1.818	+5.401	*	*			
	+0.727	+1.796	+5.342	*	*			
	+0.730	+1.793	+5.344	*	*			
	+0.724	+1.790	+5.326	*	*			
	+0.730	+1.789	+5.329	*	*			
	+0.730	+1.787	+5.326	*	*			

	X_1	X_2	F	E_1	E_2	E_3	E_4	DEL
	+0.804	+1.841	+5.977		*			+0.200000
E:	+0.198	-0.387		+0.436		+0.300		
	+0.806	+1.859	+6.056		*			
	+0.794	+1.866	+6.006		*			
	+0.795	+1.877	+6.049		*			
	+0.788	+1.881	+6.022		*			
	+0.788	+1.888	+6.049		*			
	+0.784	+1.890	+6.031		*			
	+0.785	+1.894	+6.050		*			
	+0.782	+1.896	+6.037		*			
	+0.782	+1.897	+6.044		*			
	+0.780	+1.901	+6.048		*			
	+0.811	+1.920	+6.318		*			+0.100000
E:	+0.255	-0.168		+0.427		+0.341		

	X_1	X_2	F	E_1	E_2	E_3	E_4	DEL
	+0.808	+1.926	+6.317		*			+0.100000
E:	+0.253	-0.168		+0.422		+0.348		

	+0.808	+1.929	+6.335		*			
	+0.806	+1.930	+6.325		*			
	+0.802	+1.938	+6.331		*			
	+0.801	+1.939	+6.326		*			
	+0.800	+1.941	+6.328		*			
	+0.808	+1.946	+6.398		*			+0.050000
E:	+0.265	−0.115	+0.419	+0.360				
	+0.811	+1.948	+6.430		*			
	+0.813	+1.950	+6.444		*			
	+0.813	+1.950	+6.450		*			
	+0.816	+1.953	+6.489		*			+0.025000
E:	+0.295	−0.058	+0.427	+0.354				
	+0.819	+1.954	+6.505		*			
	X_1	X_2	F	E_1	E_2	E_3	E_4	DET.
	+0.822	+1.957	+6.534		*			+0.012500
E:	+0.310	−0.029	+0.431	+0.352				
	+0.823	+1.957	+6.542		*			
	+0.825	+1.958	+6.555		*			+0.006250
E:	+0.318	−0.015	+0.433	+0.350				
	+0.826	+1.959	+6.565		*			+0.003125
E:	+0.321	−0.009	+0.434	+0.350				
	+0.827	+1.960	+6.572					+0.000781
E:	+0.324	−0.004	+0.435	+0.349				

T H E O R Y   O F   P R O G R A M M I N G

Instytut Maszyn Matematycznych PAN
Algorytmy N° 3

519.14:518.6

ON THE APPLICATION OF GRAPH THEORY TO DETERMINE THE NUMBER OF MULTISECTION LOOPS IN A PROGRAM

by Alfred SCHURMANN
Received 16th April 1963

A certain model of the pathes of a program is presented. The model loops are the graph cycles. The algorithm and the theorem is given determining the number of loops which pass through two parts of the program. Final conclusions contain some notes on the application of theorem 2.

1. INTRODUCTION

Because of the limited capacity of the computer working store the program should be divided into sections. There should be as little pathes as possible from one section to another. Cases where statements of one loop belong more than to one section should be avoided as much as possible.

Programs considered in this paper do not modify their own structure. All possible loops can be determined in such a program before its performance. In connection with this, during the segmentation of a program, the place may be found where the number of loops is the smallest.

2. THE REPRESENTATION OF PATHES OF A PROGRAM

A program statement can be divided into operation and decision statements [2]. A label may be assigned to the statement. Operational statements are performed linearly /i.e. successively, as they appear in the program/. Decision statements may be regarded as functions depending on input data, the results of which are labels. Labels are function values. The decision statement will be further considered as the function

$$ds(Y) = \begin{cases} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{cases}$$

where Y is an unknown quantity depending on statements as well as on input data of the program, $\alpha_1, \alpha_2, \ldots, \alpha_N$ are fixed labels of the program. There exist N pathes from the decision statement to the labels $\alpha_1, \alpha_2, \ldots, \alpha_N$. The pathes to be executed, during the operational time of the program, are determined by the value of the variable Y. However, this variable /otherwise operational statements and data/ does not change the pathes of the program. The pathes leave the decision statements and enter the labels.

Let us denote labels and decision statements by $X_1, X_2, \ldots, X_n$ successively as they appear in the program.

The pathes of the program are described by the directed graph $G = (X, \Gamma)$, where X is the ordered set $\{X_1, X_2, \ldots, X_n\}$ and Γ the following transformation

$$\Gamma X_{i-1} = \begin{cases} \{X_i\}, & \text{if } X_{i-1} \text{ is a label} \\ \{\alpha_1, \alpha_2, \ldots, \alpha_N\}, & \text{if } X_{i-1} \text{ is a decision statement } ds(Y) = \begin{cases} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{cases} \end{cases}$$

where the label α_k is a node X_{ik} of the set $\{X_1, \dots, X_n\}$.

To every path of the program corresponds a path in the graph G, and inversely. The given model of program pathes omits the operational statements, and it does not show us the number of operational statements between nodes X_e and X_{e+1}. This information is not needed in further consideration.

Definition

If $X_{k1}, \dots, X_{ki}$ is a path of a program, and $X_{k1} = X_{ki}$, then the path is a loop of the program. In the graph theory such a path is called cycle.

The graph G may be described by the so-called connection matrix $[C_{ij}]$. The element C_{ij} of this matrix is defined as follows:

$$C_{ij} = \begin{cases} 1, & \text{if in graph } G \text{ exists the arc}(X_i, X_j) \\ 0, & \text{if the arc}(X_i, X_j) \text{ does not exist.} \end{cases}$$

Example:

Given program:

```
begin real J,B,M,R; integer I;
A : I := 0;
   read I,B;
B : M := M + sin(B + I);
   I := I + 1;
   print M;
   if I < 100 then go to B;
   if J = 1.4 then go to B1;
   R := (M + R) × J;
   if R ≥ 50.6 then go to A;
   print R;
B1 : end
```

The nodes of the program are the following:

X_1 = A; X_2 = B; X_3 = <u>if</u> I < 100 <u>then</u> <u>go</u> <u>to</u> B;

X_4 = <u>if</u> J = 1,4 <u>then</u> <u>go</u> <u>to</u> B1;

X_5 = <u>if</u> R ≥ 50.6 <u>then</u> <u>go</u> <u>to</u> A; X_6 = B1.

The corresponding graph is of the following form

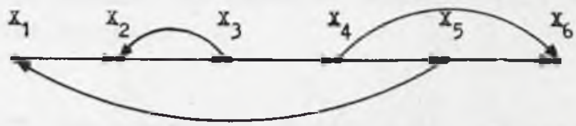


Fig. 1

The connection matrix of the graph is the following

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3. THE NUMBER OF MULTISECTION LOOPS.

Let X_i be a node of graph G. We divide the program between nodes X_i and X_{i+1}. The part of the program denoted by X_1, X_2, ..., X_i will further be called section A, the part of the program denoted by X_{i+1}, X_{i+2}, ..., X_n - section B. Assume the section B to be located in the computer internal store, and section A in the external one. The internal store can contain one section only. If, during the operation time of a program, section A is called by section B, section A is transferred to the internal store. Section B calls section A if and only if there exists the arc(X_k, X_j) for $k > i$ and for $j \leq i$. When the arc(X_k, X_j) does not belong to the loop, section B calls section A through the path ..., X_k, X_j, ... only once. The number of the above mentioned calls from section B to A is small and it may therefore be omitted.

In case the $\mathrm{arc}(X_k, X_j)$ belongs to the loop, section A is repeatedly called by section B and inversely. We further shall try to determine the number of such repeated calls equalling the number of arcs (X_k, X_j) belonging to a loop.

Definition

It will be said that a multisection loop passes between nodes /vertexes/ X_i and X_{i+1} if there exists an $\mathrm{arc}(X_k, X_j)$, where $k > i$ and $j \leq i$, belonging to a cycle of the graph G.

From the above there follows immediately the conclusion:

Corollary.

The number of multisection loops is not greater than

$$\sum_{k=i+1}^{n} \sum_{j=1}^{i} c_{kj}$$

It is easy to prove that an inverse theorem is not true /ref.to fig. 2/.

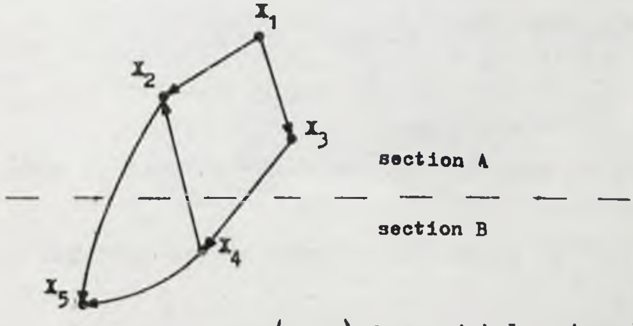


Fig. 2. $c_{42} = 1$, $\mathrm{arc}(X_4, X_2)$ does not belong to a loop.

PROBLEM.

Determine the number of multisection loops in program G.

Use will be made of an algorithm for finding the path from a to b [1]. Such an algorithm for directed graphs will be described as follows:

do not pass twice the same path in the same direction; first follow the direction of the oriented path. When reaching X_e do not follow the path that enters the vertex X_e if another choice is possible.

The algorithm

Find the element C_{kj} equalling 1 in the connection matrix of the graph G, beginning with $k = i+1$ and $j = 1$. Then check the existence of a path from vertex X_j to X_k according to the algorithm of finding the path from a to b. If such path exists there is a multisection loop. If the above-mentioned path does not exist, the arc (X_k, X_j) does not belong to a loop. The number of multisection loops is to be found by repeating these actions for all $k > i$ and $j \leq i$.

In the graph theory [1] the theorem given below is known.

Theorem 1

If G is a graph and $[C_{ij}]$ its connection matrix, the element p_{ij}^{α} of the matrix $P_{\alpha} = [C_{ij}]^{\alpha}$ equals the amount of different pathes from X_i to X_j of the length α, where $[C_{ij}]^{\alpha}$ is the matrix $[C_{ij}]$ to power α.

The length of path L is the number of arcs of which the path L consists.

Path L is the proper one if none of its vertexes is twice repeated.

Lemma

The path from X_i to X_j exists in graph G if, and only if, the element d_{ij} of matrix $D = P_1 + P_2 + \ldots + P_{n-1}$ is different than zero.

Proof

First let us prove the existance of a path from X_i to X_j if $d_{ij} \neq 0$. Acoording to the definition of d_{ij} it follows that

$$d_{ij} = p^1_{ij} + p^2_{ij} + \ldots + p^{n-1}_{ij} \qquad /1/$$

where the element p^r_{ij} of the matrix P_r is non-negative.

Thus, if $d_{ij} \neq 0$, there exists such r that $p^r_{ij} \neq 0$. This, using theorem 1, indicates the existence of a path of the length r from X_i to X_j.

And inversely, if in graph G a path exists from X_i to X_j, then there is a proper path from X_i to X_j of the length $\leq n - 1$. Thus, from theorem 1 and from /1/ one obtains $d_{ij} \neq 0$.

It results from the lemma that there exists a multisection loop in graph G, if and only if $C_{kj} = 1$ and $d_{jk} \neq 0$, where $k > i$ and $j \leq i$.

Thus, one obtains

Theorem 2.

The number of multisection loops in graph G equals

$$\sum_{j=1}^{i} \sum_{k=i+1}^{n} C_{kj} \operatorname{sign}(d_{jk}) \qquad /2/$$

CONCLUSIONS.

In section 2, X_1 has been determined as an arbitrary vertex of graph G. Thus, the described algorithm and theorem 2 permit to determine the number of loops which pass through an arbitrary place of the program. This information is necessary while dividing a program into sections.

In order to make use of the given algorithm one should investigate the average time of the program operation realizing the above algorithm.

Theorem 2 seems to be more useful than the algorithm when applied so as to divide programs.

The zero-one matrix $\left[c_{ij}\right]$ may be regarded as a boolean one. The calculation of the arithmetical sum /2/ may be performed using matrix $D' = \left[d'_{ij}\right]$, where $d'_{ij} = \operatorname{sign}\left(d_{ij}\right) > 0$, instead of matrix D.

From the definition of matrix D' and because of the fact that $p_{ij}^{\alpha} \geqslant 0$, one obtains

$$d'_{ij} = \operatorname{sign}\left(p_{ij}^{1} + p_{ij}^{2} + \dots + p_{ij}^{n-1}\right) =$$

$$= \operatorname{sign}\left(p_{ij}^{1}\right) v \operatorname{sign}\left(p_{ij}^{2}\right) v \dots v \operatorname{sign}\left(p_{ij}^{n-1}\right),$$

where the operator v is defined as follows

$$\operatorname{sing}\left(p_{ij}^{s}\right) v \operatorname{sign}\left(p_{ij}^{s+1}\right) = \begin{cases} 1, \text{ if } \operatorname{sign}\left(p_{ij}^{s}\right) = 1 \text{ or } \operatorname{sign}\left(p_{ij}^{s+1}\right) = 1 \\ 0, \text{ if } \operatorname{sign}\left(p_{ij}^{s}\right) = 0 \text{ and } \operatorname{sign}\left(p_{ij}^{s+1}\right) = 0 \end{cases}$$

Thus, the matrix D' is a logical sum of the boolean matrix $P'_{\alpha} = \left[p'^{\alpha}_{ij}\right]$ where $p'^{\alpha}_{ij} = \operatorname{sign}\left(p_{ij}^{\alpha}\right)$. This indicates that matrix D' is obtained by means of performing boolean operations on matrices $\left[C_{ij}\right]$ and P'_{α} . The product C_{kj} sign$\left(d_{jk}\right)$ in /2/ may be also treated as a logical product. In this case the sum /2/ is the number of true values of the logical product $C_{kj} \wedge d'_{jk}$.

Operations performed on boolean matrices are very fast if to each bit of a computer word corresponds one matrix element.

In connection with this the average time of program execution realizing the theorem 2 should be relatively short.

References

1. BERGE C.: Théorie des graphes et ces applications /Teorija grafov i jejo primienienija/ Izdat.Inostr.Literat., Moskva 1962.

2. KARP R.M.: A Note on the Application of Graph Theory to Digital Computer Programming, Information and Control 3, 1960:2, 179-190.

Cena zł 32,40

W. D. N. — Zam. 57/o/64. Nakł. 550 egz.