ALGORYTMY

VOL. IX • NO. 15 • 1972

INSTYTUT MASZYN MATEMATYCZNYCH

A L G O R Y T M Y
Vol. IX, N° 15 1972

Adres Redakcji: Warszawa, ul. Krzywickiego 34, tel. 28-37-29

Pow. w IMM nakł. 550 pap. offset. kl. III g. 80 zam. 105/72

GP-II-1416/68

TREŚĆ
СОДЕРЖАНИЕ
CONTENTS

str.

Komitet Redakcyjny

[illegible]

Adres Redakcji: Warszawa, ul. [illegible]

[illegible]

[illegible]

ALGORYTMY, IX, No 15, 1972, 5-17

AN ATTEMPT OF MECHANIZATION OF HALL COLLECTING PROCESS

by Emanuel CZYŻO

Received January 14th, 1972

In this paper the collecting algorithm is described. The algorithm is an implementation of Hall collecting process for positive word written in generators of a nilpotent free group of a given degree of nilpotency.

1. INTRODUCTION

J.I. Cannon in the review [2] describes the present situation of application of computers in the group theory. He states that one of the most important problems in the field is the mechanization of the tedious algebra and specially the commutator calculus. The collecting process plays important part here in connection with problems involving the presentation of groups in terms of generators and relations.

The problem was studied by H. Felsch (see [2]), who has written a program for words of the form $(x_1x_2)^n$. Campbell and Lamberth (see [2]) have written a program SLIP, which is an implementation of the collecting process for words of a nilpotent group of finite class.

In this paper the similar implementation method of the collecting process with the aid of a computer is considered.

The method has been applicated in a program HALL+ (see [4]). The program is an implementation of the collecting process for positive words written in generators of a nilpotent group of a given nilpotency. The description of the method is supplemented with results obtained by means of the program HALL+.

We give also the table, which gives number of basic commutators for a given weight and number of generators, calculated from Witt's formula.

The further development of this method for obtaining an implementation of the full collecting process (i.e. for arbitrary: positive and negative words) and some similar processes is in preparation on the base of described results.

The terminology and notations of this paper is the same as in M.Hall's book [1].

2. COLLECTING PROCESS

2.1. Commutators

Let X be a finite set of letters $x_1, x_2, \ldots, x_r$. Let us define a set of strings, which we shall call c o m m u t a t o r s .

Definition

a. String c_i consisting of one letter x_i, that is $c_i = x_i$ is a commutator of weight one. The weight of the commutator c_i we shall denote $w(c_i)$,

b. if c_i, c_j are commutators then the string $c_k = (c_i, c_j)$ is a commutator and $w(c_k) = w(c_i) + w(c_j)$.

The set X is finite, hence the set of commutators of defined weight is also finite. We order commutators any way so

that: if $w(c_i) > w(c_j)$ then c_i follows c_j. Let c_s denote the commutator which takes the s-th position (according to our ordering) then $c_u \leqslant c_v$ iff $u \leqslant v$.

2.2. Collecting process

For words written in commutators we define the collecting process described by M. Hall ([1], p. 165).

For a word

$$f = c_{i_1} c_{i_2} \dots c_{i_n}$$

the subword

$$w = c_{i_1} c_{i_2} \dots c_{i_m} \quad (1 \leqslant m \leqslant n)$$

is called the c o l l e c t e d p a r t of the word f if m is the greatest index such that

$$i_1 \leqslant i_2 \leqslant \dots \leqslant i_m \quad \text{and} \quad i_m \leqslant i_k \quad \text{for } k = m+1, \dots, n.$$

Let w_0 be the collectod part of f

$$f = f_0 = w_0 v_0$$

For a word $f_1 = w_1 v_1$ with a collected part w_1 we form f_{1+1} as follows.

For

$$v_1 = c_{i_1} \dots c_{i_j} \dots c_{i_n}$$

let c_u be the smallest commutator (in the sense of defined order) in the word v_1 and let c_{i_j} is the first occurrence of this commutator in v_1.

Define

$$v_1' = c_{i_1} \dots c_{i_j} \quad c_{i_{j-1}} \quad c_k \, c_{i_{j+1}} \dots c_{i_n}$$

where

$$c_k = \left(c_{i_{j-1}}, c_{i_j}\right)$$

(notice that if w_1 is a collected part of the word then $j > 1$). We define f_{l+1} as follows

$$f_{l+1} = w_1 v_1'$$

and denote the collected part of f_{l+1} as w_{l+1}. The process terminates if for some m the collected part of f_m is equal f_m. For example if $f = x_2 x_1$, then

$$f_1 = x_1 x_2 (x_2, x_1) \text{ and } f_k = f_1 \text{ for } k \geqslant 1.$$

Observe that if $x_1, x_2, \ldots, x_r$ are generators of the group F (where F is a free group with r generators) and if

$$(c_v, c_u) = c_v^{-1} c_u^{-1} c_v c_u$$

then all words f_k (for $k = 0, 1, \ldots$) represent the same element of the group F because in this group

$$c_v c_u = c_u c_v (c_v, c_u)$$

2.3. Basic commutators

In the collecting process only commutators of special form can appear.

Definition

1. Commutators of the weight 1 are basic commutators
2. A commutator $c_k = (c_v, c_u)$ is basic of the weight n if
   a. c_v, c_u are basic of the weight less than n and $w(c_v) + w(c_u) = n$
   b. $c_v > c_u$ and if $c_v = (c_s, c_t)$ then $c_t \leqslant c_u$.

We use the previously defined order and enumeration for basic commutators. It's easy to see that in the collecting process only basic commutators appear.

The following theorem gives the explanation of the used name by these commutators.

Theorem

If F is the free group with generators $x_1, x_2, \ldots, x_r$ and if in a sequence of basic commutators $c_1, c_2, \ldots, c_t$ are those of weights 1, 2,..., n, then an arbitrary element f of F has a unique representation,

$$f \equiv c_1^{e_1} c_2^{e_2} \ldots c_t^{e_t} \mod F_{n+1}$$

The basic commutators of weight n form a basis for the free Abelian group F_n/F_{n+1}. See [1] p. 175.

2.4. Witt's formula

Let $M(r,n)$ denote the number of basic commutators of the weight n for r generators. The following Witt's formula holds

$$M(r, n) = 1/n\left(\sum_d \mu(d)\ r^{\frac{n}{d}}\right)$$

where d runs over all divisors of n and $\mu(m)$ the Möbius's function.

The formula is proved in [1] p. 169. Numbers of basic commutators for r = 2, 3,..., 9 and n = 2,3,...., 11 are given in the table 1.

3. IMPLEMENTATION OF THE COLLECTING PROCESS - PROGRAM HALL+

The program HALL+ is an implementation of the collecting process for positive words written in the alphabet

$$X = \{x_1, x_2, \ldots, x_r\} \quad \text{for } r \leq 9$$

The program is written in Gier ALGOL 4 [3].

Table 1

n/r	2	3	4	5	6	7	8	9	r/n
2	1	3	6	10	15	21	28	36	2
3	2	8	20	40	70	112	168	240	3
4	3	18	60	150	315	588	1008	1620	4
5	6	48	204	624	1554	3360	6552	11808	5
6	9	116	670	2580	7735	19544	43596	88440	6
7	18	312	2340	11160	39990	117648	299592	683280	7
8	30	810	8160	48750	209790	720300	2096640	5380020	8
9	56	2184	29120	217000	1119720	4483696	14913024	43046640	9
10	99	5880	104754	976248	6045837	28245840	107370900	348672528	10
11	186	16104	381300	4438920	32981550	179756976	780903144	2852823600	11
s	412	25486	526638	5695487	40406582	213348092	905634660	3250708221	s

The table gives numbers of basic commutators of the given weight $n \leq 11$ for a given number of generators $r \leq 9$ as calculated from Witt's formula.

The row s gives a total number of basic commutators of weight ≤ 11 for r generators

3.1. Enumeration of commutators

As observed during the collecting process only basic commutators are generated. Let us define an enumeration for these commutators as follows.

$N(c_1)$ denotes a number of a commutator c_1.

Definition

a. $N(x_i) = i$ for $i = 1,2,\ldots,9$

b. if $c_k = (c_v, c_u)$ then $N(c_k) = N(c_v)\ 10^{w(c_u)} + N(c_u)$

For example the commutator (c_2, c_1) with a number 21 is denoted by c_{21}, and the commutator $((c_2, c_1), c_3)$ with a number 213 is denoted by c_{213}.

Now, we order all commutators according to their numbers, that means $c_u \leqslant c_v$ iff $N(c_u) \leqslant N(c_v)$.

From the definition of a number of a commutator it follows that the number of the commutator of the weight n contains exactly n digits in decimal expansion, thus the assumed ordering preserves the order of weights. It can be observed that in the case of basic commutators different commutators correspond to different numbers.

3.2. Input data

The input data for the program is a word written in generators, using only digits and letters. For example we write the word $f = (x_1\ x_2\ x_3)^2$ on the paper tape in the following form x1x2x3x1x2x3 Ø. Each word is terminated by Ø.

Before we start the collecting process we define the value of the variable WEIGHTMAX. For technical reasons only commutators of weight $\leqslant 11$ are permitted.

3.3. Machine representation of a word

The word which takes part in the collecting process is represented in the array BUF[0 : 4095]. Every commutator which occurs in the word is written in four successive elements of the array.

(*)	the address of the former commutator	the number of the commutator	the weight of the commutator	the address of the next commutator

The index of the second element in (*) is an address of the commutator.

For example the word $f = x_3\, x_2\, x_1$ is represented as follows:

BUF	1	3	1	5	1	2	1	9	5	1	1	-1
index	0	1				5				9		11

We put BUF [0] = 1 and BUF [11] = -1.

It follows from the method of representation of a word, that it is possible to store words consisting at most 1024 commutators. The present version of the program does not allow words of greater length, however a simple modification of the program allows to remove this limitation if needed.

The following limitation for numbers of commutators $N(c_u) \leq 2^{39} - 1$ follows from the used way of presentation of commutators. Hence in the collecting process we can consider only commutators of weight not greater than 11.

3.4. Implementation of the rule $c_v\, c_u = c_u\, c_v$ (c_v, c_u)

Let us consider the word $f_0 = \alpha_0\, \beta_0$ where $\alpha_0 = \emptyset$ (empty word) and $\beta_0 = x_3\, x_2\, x_1$. First we find the address of the

first occurrence of a minimal commutator (c_{min}) appearing in β_0. The address is stored in PLACE. In our example $c_{min} = c_1$ and PLACE = 9. We need to transpose c_1 with the former commutator $(c_2$ in our example$)$. Transposition of two commutators consist in transposition of their numbers and weight. Links between these commutators remain unchanged.

In the case $w(c_2) + w(c_1) >$ WEIGHTMAX no new commutator is added.

In the case $w(c_2) + w(c_1) \leq$ WEIGHTMAX the new commutator $(c_2, c_1) = c_{21}$ is added. Some links must be changed too.

In our example

BUF	1	3	1	5	1	1	1	9	5	2	1	13	9	21	2	-1	(*)
index	1				5				9				13				

It is easy to observe that the ordering in $(*)$ is

$$\beta_0^1 = c_3\ c_1\ c_2\ c_{21}$$

New value of PLACE is 5. Repeating the described step we obtain

BUF	1	1	1	5	1	3	1	17	17	2	1	13	9	21	2	-1	5	31	2	9
index	1				5				9				13				17			

where $c_{min} = c_1$ is on the first position of

$$\beta_0^2 = c_1\ c_3\ c_{31}\ c_2\ c_{21}$$

We determine a new collected part $\alpha_1 = \alpha_0\ c_1$ and $\beta_1 = c_3\ c_{31}\ c_2\ c_{21}$, and we continue the collecting process, as far as we obtain $\beta_n = \emptyset$.

3.5. Output data

After terminating the process we output the resulting word in an abbreviated form on the paper tape.

For example the word $f = c_1\ c_1\ c_2\ c_2\ c_{21}\ c_{212}$ we output in form $(c1)\uparrow 2(c2)\uparrow 2(c21)\,(c212)\,\emptyset$. Each word is terminated by $\emptyset$.

4. APPLICATIONS

4.1. Using this program the collecting process for words of form

$$f(x_1,\ldots,x_r) = (x_1 \ldots x_r)^n \quad \text{for } n = 2, 3, \ldots, 9,$$

was carried out. The following table shows results of the experiment. We give for each r the computation time for n = 2 and given value of the variable WEIGHTMAX.

	n = 2	
r	Wmax	time
2	11	10 sec
3	11	8 min
4	7	24 min
5	5	5 min
6	4	4 min
7	4	16 min
8	4	40 min
9	3	3,5 min

The program does not work for greater values of WEIGHTMAX because of shortage of memory. To give better idea we present words obtained after the termination of the process for r = 2, n = 4 and WEIGHTMAX = 7 (then a↑b denotes a^b)

(c1)↑(4)(c2)↑(4)(c21)↑(6)(c211)↑(4)(c212)↑(14)(c2111)(c2112)↑
(11)(c2122)↑(11)(c21112)↑(3)(c21121)↑(11)(c21122)↑(10)(c21221)↑
(42)(c21222)↑(3)(c211121)↑(3)(c211122)↑(3)(c21121)↑(32)(c211222)
(3)(c212211)↑(37)(c212221)↑(39)(c2111211)(c2111212)↑(5)(c2111221)↑
(9)(c2111222)(c21121211)↑(12)(c21122111)↑(18)(c21122112)↑(56)
(c21122211)↑(31)(c21221211)↑(59)(c21222111)↑(35)(c21222121)↑(79)
(c21222211)↑(12)∅

and for r = 4, n = 4 and WEIGHTMAX = 3

(c1)↑(4)(c2)↑(4)(c3)↑(4)(c4)↑(4)(c21)↑(6)(c31)↑(6)(c32)↑(6)(c41)↑(6)(c42)↑(6)(c43)↑(6)(c211)↑(4)(c212)↑(14)(c213)↑(20)(c214)↑(20)(c311)↑(4)(c312)↑(14)(c313)↑(14)(c314)↑(20)(c322)↑(4)(c323)↑(14)(c324)↑(20)(c411)↑(4)(c412)↑(14)(c413)↑(14)(c414)↑(14)(c422)↑(4)(c423)↑(14)(c424)↑(14)(c433)↑(4)(c434)↑(14)∅

All computation we done on GIER computer in the Warsaw University's Centre of Numerical Calculations.

4.2. Using this program coefficients b_j in the formula

$$(*) \quad e_i = b_1\binom{n}{1} + b_2\binom{n}{2} + \ldots + b_m\binom{n}{m}$$

were calculated by K. Dałek [5].

The formula $(*)$ gives exponents in the expression

$$(x_1 \ldots x_r)^n = x_1^n \ldots x_r^n\, c_{r+1}^{e_{r+1}} \ldots c_s^{e_s}\, R_1 \ldots R_t$$

where

$c_s < R_1$ and $w(R_k) > \text{Wmax}$ for $k \geqslant 1$

and m – the weight of the commutator c_i

and $b_1, b_2, \ldots, b_m$ – are non-negative integers dependent on c_i and independent on n. C.f. [1] p. 182

4.3. The new procedure COLLECTING is in preparation. The procedure will be applicable for the collecting process for any words written in basic commutators with any exponents positive or negative, in spite of the program HALL+ applicable for positive words.

5. REMARKS AND CONCLUSIONS

In this paper some basic researches of the automatization of the commutator calculus were presented. The problems in question are far from the end at yet. Up to now there have been no general methods, which would enable a very successful application of computers for solving commutator problems. Our results on the problems at present are in a stage of research work and they are limited mainly by the available equipment. The difficulties lies in the lack of an adequate computer and a suitable external language as well. A great amount of data is processed in solving the problems, so that more interesting from the grouptheoretical viewpoint results might be obtained on a very modern computer of very high standard. The author feels the design of a special purpose language would speed up investigations in the field.

A new system of processing words aided mainly to commutator problems is now in preparation. The system will consist of two sets of procedures: basic procedures and special procedures, and it will be elaborated during preparation of new algorithms.

References

[1] HALL M.: The Theory of Groups, New York, 1959.

[2] CANNON J.I.: Computers in Group Theory: a Survey, Comm. ACM 12 1969: 3-12.

[3] A Manual of Gier Algol 4, A/S Regnecentralen, Copenhagen, 1967.

[4] CZYŻO E.: Komputery w teorii grup. Proces wybierania dla słów dodatnich, Sprawozdania ZON UW, 24, 1970.

[5] DAŁEK K.: Computation of Exponents in Hall Formula. Bull. Ac. Pol. Sci. ser. math, vol. XIX, No 8, 1970: 711-718.

PRÓBA AUTOMATYZACJI PROCESU WYBIERANIA

Streszczenie

W artykule opisano proces wybierania M.Halla pozwalający dla grupy wolnej $F(x_1, \ldots, x_r)$ uzyskiwać przedstawienie elementów w postaci:

$$f = x_{\alpha_1}^{a_1} x_{\alpha_2}^{a_2} \ldots x_{\alpha_k}^{a_k} = c_1^{e_1} c_2^{e_2} \ldots c_t^{e_t} R_1 \ldots R_n$$

gdzie $c_1 = x_1, \ldots, c_r = x_r, c_{r+1} = (x_2, x_1), \ldots, c_t$

są komutatorami bazowymi wagi $\leq m$, uporządkowanymi względem wzrastających indeksów, zaś $R_1, \ldots R_n$ są komutatorami wagi $> m$, nieuporządkowanymi, $a_1, a_2, \ldots, a_k$ oraz $e_1, e_2, \ldots, e_t$ - liczby naturalne.
Podano opis realizacji procesu wybierania dla słów dodatnich na maszynę GIER w języku GIER Algol 4.
Przy użyciu programu policzono wybraną postać słów $(x_1 x_2 \ldots x_r)^n$ dla przypadków $r + n + m \leq 12$ w czasach rzędu od minut do godziny.

ПРОБА АВТОМАТИЗАЦИИ ПРОЦЕССА ВЫБОРКИ

Резюме

В настоящей работе описан алгоритм выбора. Этот алгоритм является дополнением к процессу выбора Голла для положительного слова записанного в генераторе нильпотентной свободной группы данной степени нильпотенции.

Warszawa, ul. Krzywickiego 34
A.M.

ALGORYTMY, IX, No 15, 1972, 19-63

PRZEGLĄD METOD SZUKANIA EKSTREMUM FUNKCJI WIELU ZMIENNYCH

Andrzej MULAWA

Pracę złożono 15.01.1970

Praca stanowi przegląd algorytmów szukania największej i najmniejszej wartości funkcji określonych na podzbiorach przestrzeni euklidesowej. Rozpatruje się tylko algorytmy deterministyczne wykorzystujące wartości funkcji i wartości jej gradientu - zadanych numerycznie, albo uwzględniające tylko wartości funkcji. Omówione metody są często stosowane przy optymalizacji statycznej obiektów przemysłowych. W artykule bardziej szczegółowo omawia się optymalizację statyczną kotła energetycznego.

Spis treści

Wykaz oznaczeń

$E^n \quad U^n$ - n-wymiarowe przestrzenie euklidesowe

$\in$ - element należy

$\| \underline{p} \| = \left(\sum_{i=1}^{n} p_i^2 \right)^{1/2}$ - norma euklidesowa wektora $\underline{p}$

$f(\underline{x})$ - funkcja skalarna wektora $\underline{x}^T = (x_1, x_2, \ldots, x_n)$

$\nabla f(\underline{x})$ - gradient funkcji f

$\{\underline{\xi}_i\} \{\underline{c}_i\}$

$i = 1, \ldots, n$ - układy wersorów

$\underline{h}(\underline{x})$, $\underline{g}(\underline{x})$ - funkcje wektorowe wektora $\underline{x}$

A, B, C, M - macierze kwadratowe

$\alpha \ \beta \ \delta \ \lambda$ s x - wielkości skalarne

E - macierz jednostkowa

ϵ - parametr oznaczający dokładność

l_n - przedział nieokreśloności po n eksperymentach

L_n - minimaksowy przedział nieokreśloności

F_n - n-ty wyraz ciągu Fibonacciego

i j k l r - wskaźniki współrzędnych wektora lub kroków iteracyjnych

$\underline{p}_i \ \underline{p}_i^T$ - oznacza tensorowy iloczyn wektorów, czyli diadę

1. WSTĘP

Najprostszym problemem optymalizacji jest optymalizacja statyczna w przypadku deterministycznym. Matematycznie sprowadza się ona do znalezienia ekstremum funkcji skalarnej $f(\underline{x})$, gdzie $\underline{x}$ jest wektorem n-wymiarowej przestrzeni euklidesowej E^n. W dalszym ciągu będziemy mówili tylko o minimum $f(\underline{x})$ pamiętając, że poszukiwanie maksimum $f(\underline{x})$ sprowadza się do szukania min. $\{-f(\underline{x})\}$. Poza tym $\underline{x}$ musi należeć do obszaru dopuszczalnego D. W przypadku optymalizacji statycznej obszar dopuszczalny jest zazwyczaj ograniczony nierównościami: $\underline{h}(\underline{x}) \leq \underline{x} \leq \underline{g}(\underline{x})$. Niekiedy w rozważaniach przyjmuje się, że obszar D może pokrywać się z całą przestrzenią E^n.

W ogólności zakłada się, że nie można znaleźć ekstremum $f(\underline{x})$ poprzez przyrównanie pochodnych $\frac{\partial f(\underline{x})}{\partial \underline{x}}$ do zera. Dzieje się to dlatego, że:

- $f(\underline{x})$ nie jest dana w sposób jawny, tylko możemy obliczać jej poszczególne wartości. Mówimy wówczas, że $f(\underline{x})$ jest zadana w sposób numeryczny.
- Co do ∇f to może on być także zadany numerycznie bądź w ogóle może nie być znane wyrażenie określające go.
- Wyrażenie analityczne na $f(\underline{x})$ jest tak złożone i nieliniowe, że układ równań $\nabla f = \underline{0}$ można rozwiązać tylko iteracyjnie. Proces ten może okazać się wolniej zbieżny od procesu iteracyjnego minimalizacji bezpośredniej funkcji $f(\underline{x})$.

Metody optymalizacji statycznej można podzielić na aproksymacyjne, iteracyjne, metody szukania ślepego oraz metody mieszane. W pracy będą omawiane tylko metody iteracyjne. Można je także podzielić na metody wchodzenia na wzgórze /hill climbing/ oraz metody eliminacyjne.

Klasyczną metodą eliminacyjną jest metoda ε-minimax, opisana w rozdziale traktującym o szukaniu jednowymiarowym. Jak dotąd dla przypadków wielowymiarowych metody eliminacyjne znalazły zastosowanie dla bardzo wąskiej klasy problemów [8]. Metody aproksymacyjne są ściśle związane z szukaniem jednowymiarowym. Z tego względu będę omawiał tylko metody hill climbing. Jeśli w metodzie wykorzystuje się gradient $f(\underline{x})$ bądź jego aproksymację, to metodę zalicza się do gradientowych. Jeżeli algorytm wykorzystuje tylko wartości $f(\underline{x})$, to zaliczamy go do metod szukania bezpośredniego. Pewne metody gradientowe opisane są w rozdziale trzecim tej pracy, a metody bezpośrednie w czwartym.

W zasadzie praca ta zawiera przegląd algorytmów; wyniki autora zawarte są w punkcie 2.8.

Celem przeglądu algorytmów było wybranie odpowiedniej metody optymalizacji wskaźnika jakości regulacji kotła energetycznego. Kryterium to jest dane w sposób niejawny i zależności między wartością kryterium a zbiorem parametrów (są to nastawy regulatorów), nie dają się rozwikłać. Prowadzi się do tego, że $f(\underline{x})$ jest zadana tylko numerycznie. Gradient tej funkcji nie jest dany. Pod tym kątem dokonano wyboru dwóch algorytmów: Powella-Zangwilla i Rosenbrocka. Uzasadnienie wyboru znajduje się w rozdziale 6 oraz w szczegółowych wnioskach występujących po każdym algorytmie.

Jako kryterium przyjęto liczbę obliczeń wskaźnika jakości - $f(\underline{x})$. Kryterium to jest szeroko stosowane w analogicznych przeglądach algorytmów, dlatego że proces organizacji szukania minimum $f(\underline{x})$ jest znacznie krótszy /co najmniej o rząd wielkości w praktycznych przypadkach/ niż czas obliczeń wartości $f(\underline{x})$.

Ograniczenia w naszym przypadku mają postać: $\underline{h} \leqslant \underline{x} \leqslant \underline{g}$. Rozwiązanie problemu ograniczeń następuje poprzez odpowiednie transformacje. Dokładniejsze informacje na ten temat znajdują się w rozdziale 3.

Trzeba zaznaczyć, że optymalizacja będzie przeprowadzana na modelu cyfrowym kotła. W tym celu wykorzysta się język symulacyjny Cemma 2 dla maszyny ZAM 41. Nie uwzględnia się wówczas zakłóceń stochastycznych i błędów pomiarów. Z tego względu nie rozpatrywano metod biorących pod uwagę te czynniki.

2. POSZUKIWANIE JEDNOWYMIAROWE

2.1. Wprowadzenie

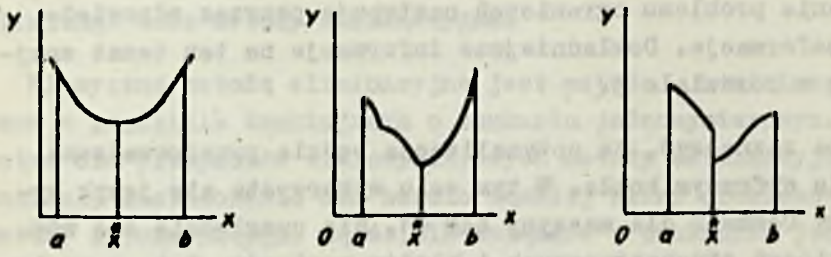
Większość algorytmów poszukiwania najmniejszej wartości funkcji wielu zmiennych stanowi sekwencję jednowymiarowych minimalizacji, dlatego na początku omówimy procedury poszukiwania minimum funkcji jednej zmiennej. Algorytmy te korzystają z pewnego pojęcia, zwanego założeniem o jednomodalności

funkcji minimalizowanej. Założenie jednomodalności jest dostatecznie ogólne, by obejmowało większość praktycznych problemów [8]. Korzystając z tego założenia nie możemy określić położenia kresu dolnego funkcji, a tylko obszar, w którym on się znajduje.

2.2. Jednomodalność

Niech będzie dana funkcja $y = y(x)$ w przedziale $[a, b]$. Funkcja ta jest jednomodalna, gdy w $[a, b]$ znajduje się taki punkt $\overset{*}{x}$, że dla: $a \leq x \leq \overset{*}{x}$ funkcja $y(x)$ jest ściśle malejąca, a dla $\overset{*}{x} \leq x \leq b$, $y(x)$ jest ściśle rosnąca. Sam punkt $\overset{*}{x}$ może należeć zarówno do pierwszego jak i drugiego przedziału[1].

Przykłady funkcji jednomodalnych podaje rysunek 1.



Rys. 1

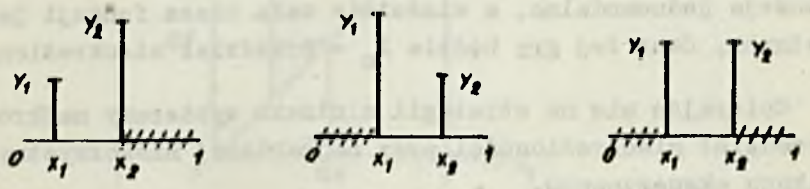
2.3. Przedział nieokreśloności

Jednomodalność jest własnością niezmienniczą względem zmiany skali. Nie naruszając ogólności będziemy rozpatrywać y

[1] Właściwie w literaturze pod pojęciem jednomodalności y rozumie się, że dla $a \leq x \leq \overset{*}{x}$ funkcja jest ściśle rosnąca, a dla $\overset{*}{x} \leq x \leq b$ ściśle malejąca. Ponieważ szukamy kresu dolnego y, zmieniliśmy to pojęcie.

w przedziale $[0,1]$. Po obliczeniu dwóch wartości funkcji jednomodalnej zawsze możemy wyodrębnić przedział, w którym minimum nie może się znajdować.

Najlepiej objaśni to rysunek 2.



Rys. 2

W dalszym ciągu każde wyliczenie wartości funkcji $y = y(x)$ będziemy nazywali eksperymentem. W ogólności po n eksperymentach w punkcie x_K będziemy mieli najlepszą aproksymację minimum funkcji f.

Oznaczmy tę aproksymację przez $y(x_K) = y_K$. Oznaczmy krańce przedziału w następujący sposób: $x_o = 0$ $x_{n+1} = 1$. Przedział, w którym minimum musi się znajdować nazwiemy przedziałem nieokreśloności i oznaczymy przez l_n. Zachodzi wówczas związek

$$l_n = x_{K+1} - x_{K-1} \tag{1}$$

W rozważaniach ponadto przyjęto, że wartości x_K są ponumerowane w ciąg rosnący. Długość l_n zależy od rozmieszczenia eksperymentów x_k oraz od eksperymentu x_K, co zapisujemy przez:

$$l_n = l_n(x_k, K) \tag{2}$$

Długość przedziału nieokreśloności zależy od położenia wszystkich eksperymentów x_k $k = 1,\ldots, n$. Powinniśmy więc napisać, że $l_n = l_n(x_1,\ldots, x_k,\ldots x_n, K)$. Notacja $l_n(x_k, K)$ została wykorzystana ze względu na ekonomię miejsca.

2.4. Minimax

W dalszym ciągu korzystać będziemy z elementarnych pojęć teorii gier. Wybór położenia eksperymentów x_k będzie naszą strategią. Wynik tych eksperymentów y_k - strategią przeciwnika. Oczywiście naszym urojonym przeciwnikiem będzie podana funkcja jednomodalna, a właściwie cała klasa funkcji jednomodalnych. Ceną tej gry będzie l_n - przedział nieokreśloności.

Opierając się na strategii minimaxu wybieramy najkrótszy przedział nieokreśloności przy najbardziej niekorzystnych wynikach eksperymentu.

$$L_n = \max_{1 \leq K \leq n} \left\{ l_n \left(x_k, K \right) \right\}$$

$$L_n^* = \min_{x_k} L_n = \min_{x_k} \max_{1 \leq K \leq n} \left\{ l_n \left(x_k, K \right) \right\} \qquad (3)$$

Min
x_k oznacza, że minimalizujemy, ze względu na cały zbiór eksperymentów $\{x_k\}$ k = 1,..., n. Patrz uwaga do wzoru (2).

Dla danej liczby eksperymentów n istnieje tylko jeden L_n^*. Strategia minimaxu jest strategią uniwersalną (niezależną od y), przy założeniu że y(x)jest funkcją jednomodalną.

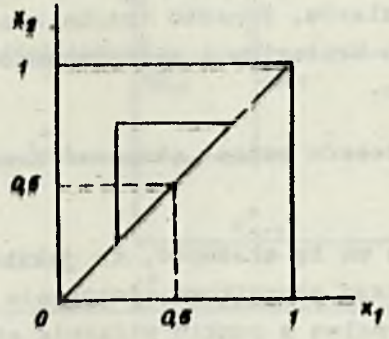
2.5. Przykład

Rozpatrzmy przypadek n = 2

$$L_2 = \max \left[\left(x_2 - x_0 \right), \left(x_3 - x_1 \right) \right] = \max \left[x_2, 1 - x_1 \right] \qquad (4)$$

Wykres warstwic tej funkcji z rysunku 3 wskazuje, że $\min L_2 = L_2^*$, zachodzi przy $x_1 = x_2 = 0{,}5$. Biorąc pod uwagę warunek $x_2 > x_1$, nie możemy osiągnąć tego minimum, ale możemy zbliżyć się do niego o wielkość ε, gdzie ε jest najmniejszą możliwą perturbacją zmiennej x.

$$L_2 \left[\frac{1}{2} + \frac{\varepsilon}{2}, \frac{1}{2} + \frac{\varepsilon}{2} \right] = \frac{1}{2} + \frac{\varepsilon}{2} \qquad (5)$$



Rys. 3

Jeżeli zwiększać teraz liczbę eksperymentów jednoczesnych, to L_n^* (n jest liczbą parzystą) oraz L_{n+1}^* związane są związkiem rekurencyjnym

$$L_{n+1}^* = L_n^* + \frac{\varepsilon}{\frac{n}{2} + 1}, \tag{6}$$

$$L_n^* = \frac{1 + \varepsilon}{\frac{n}{2} + 1}. \tag{7}$$

Wyprowadzenie tych związków można znaleźć w [8].

Przy jednoczesnym wykonywaniu eksperymentów nie wykorzystujemy informacji zawartej w każdym wyniku y. Można się spodziewać, że przy iteracyjnym wyznaczaniu L_n^*, przedział nieokreśloności będzie malał dużo szybciej ze wzrostem n.

2.6. Liczby Fibonacciego

Wśród metod iteracyjnych najefektywniejszą metodą przy przyjęciu prawa minimaxu jest metoda Kiefera [8]. Autor tej metody nazwał ją: ℓ -minimax.

Iteracyjne szukanie kresu dolnego funkcji jednomodalnej możemy uważać za proces decyzyjny n-etapowy. Na każdym etapie k;

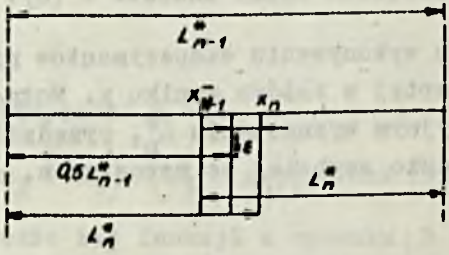
$1 < k < n$ podejmujemy decyzję, gdzie umieścić eksperyment x_k. Strategia optymalna minimalizuje l_k przy jak najniekorzystniejszych wynikach pomiarów. Ponadto trzeba uwzględnić parametr ε. Jak już wspomniano kryterium i cały schemat postępowania nazywamy ε-minimaxem.

Do tego rodzaju procesów można stosować zasadę optymalności Bellmana:

Strategia optymalna ma tę własność, że jakikolwiek byłby stan początkowy i decyzja początkowa, pozostałe decyzje muszą tworzyć strategię optymalną z punktu widzenia stanu wynikłego z pierwszej decyzji.

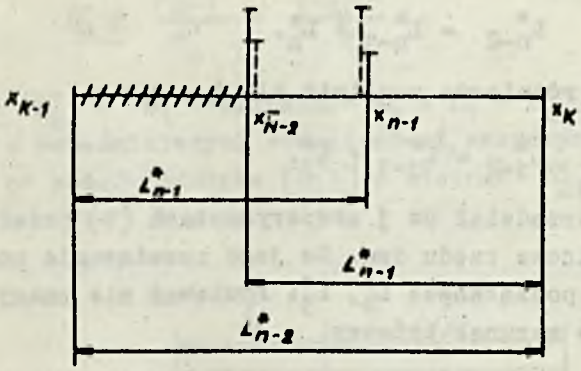
Stanem początkowym jest zawsze przedział nieokreśloności $[0,1]$. Ponadto musimy znać liczbę etapów n.

Przypuśćmy, że wykonano n - 1 pierwszych eksperymentów. Przy strategii optymalnej obszar nieokreśloności wynosi L^*_{n-1}. Wewnątrz tego przedziału znajduje się eksperyment x_{N-1} o najmniejszej wartości y_{n-1}. Gdzie należy umieścić x_n, by L_n było równe L^*_n. Z punktu 2.5 wiemy, że x_{N-1} i x_n muszą być odchylone o $\frac{\varepsilon}{2}$ od środka L^*_{n-1}. Ilustruje to rysunek 4.



Rys. 4

Przejdźmy teraz do etapu poprzedniego. Wykonano n-2 eksperymentów. L^*_{n-2} - przedział nieokreśloności przy założeniu optymalnej strategii. x_{N-2} najmniejsza wartość y uzyskana przez te n-2 eksperymentów. Jak wiadomo znajduje się on wewnątrz L^*_{n-2}. Jak umieścić eksperyment x_{n-1}? $y\left(x_{n-1}\right)$ może być mniejsze od $y\left(x_{n-2}\right)$ lub większe. Ilustruje to rys. 5

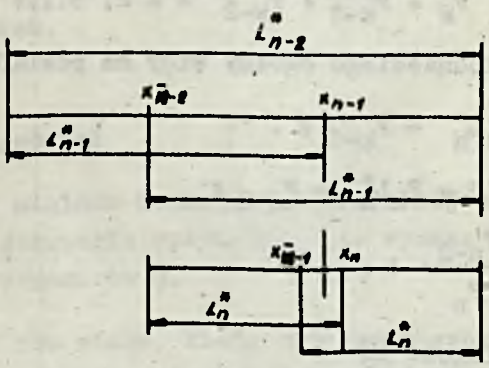


Rys. 5

W zależności od tego przedziałem nieokreśloności może być odcinek

$$\left[\bar{x}_{N-2},\ x_k\right] \quad \text{lub} \quad \left[x_{k-1},\ x_{n-1}\right].$$

Ten drugi przypadek zaznaczono na rysunku linią przerywaną. Przy strategii optymalnej oba odcinki muszą mieć długość L^*_{n-1} [1]. Wobec tego mamy wyznaczone położenie $\bar{x}_{N-1}$ i $\bar{x}_{N-2}$. Te proste zależności geometryczne między: L^*_{n-2} L^*_{n-1} L^*_n można przedstawić na rysunku 6.



Rys. 6

[1] Gdyż wynik nie jest zależny od rezultatów eksperymentu

Biorąc pod uwagę ten rysunek otrzymujemy

$$L^*_{n-2} = L^*_{n-1} + L^*_n. \tag{8}$$

Można to równanie różnicowe uogólnić dla $1 < j < n$

$$L^*_{j-2} = L^*_{j-1} + L^*_j. \tag{9}$$

L^*_j optymalny przedział po j eksperymentach (9) przedstawia równanie różnicowe rzędu dwa. Do jego rozwiązania potrzebne są dwa warunki początkowe: L^*_2, L^*_3. Ponieważ nie znamy ich, musimy wykorzystać warunek końcowy

$$L^*_{n-1} = 2 L^*_n - \varepsilon, \tag{10}$$

ale:

$$L^*_{n-2} = L^*_{n-1} + L^*_n = 2 L^*_n - \varepsilon + L^*_n = 3L^*_n - \varepsilon,$$

$$L^*_{n-3} = L^*_{n-2} + L^*_{n-1} = 5 L_n - 2\varepsilon,$$

$$L^*_{n-4} = 8 L^*_n - 3\varepsilon. \tag{11}$$

Zauważmy, że współczynniki przy L^*_n są liczbami Fibonacciego

$$F_o = F_1 = 1 \quad F_k = F_{k-1} + F_{k-2} \quad k = 2, 3, \ldots \tag{12}$$

Przy użyciu liczb Fibonacciego ogólny wzór ma postać:

$$L^*_{n-k} = F_{k+1} L^*_n - F_{k-1} \varepsilon. \tag{13}$$

Jak wiemy: $L^*_1 = 1 = F_n L^*_n - F_{n-2} \varepsilon$

$$L^*_n = \frac{1}{F_n} + \frac{F_{n-2}}{F_n} \varepsilon. \tag{14}$$

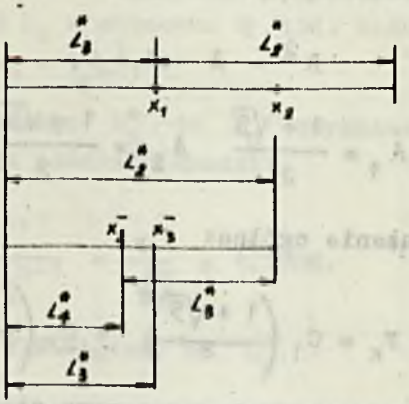
Obliczmy teraz długość L^*_2

$$L^*_2 = F_{n-1} L^*_n - F_{n-3} \varepsilon \tag{15}$$

Po prawej stronie tej zależności mamy wielkości znane. Po zastosowaniu wzorów (9) i (14) otrzymujemy:

$$L_2^* = \frac{F_{n-1}}{F_n} + \frac{(-1)^n}{F_n}\varepsilon. \qquad (16)$$

Mając L_2^* i L_1^* obliczamy $L_3^* = L_1^* - L_2^*$ itd. Oczywiście, zgodnie z wcześniejszymi rozważaniami, eksperymenty x_1 i x_2 są odległe od końców odcinka $[0,1]$ o odcinek L_2^*, jak na rysunku 7.



Rys. 7

Przy dalszym analogicznym postępowaniu uzyskamy strategię ε - minimaksu.

2.7. Złoty podział

Szukanie minimum funkcji jednomodalnej za pomocą ε-minimaksu jest strategią optymalną, ale wymaga znajomości z góry liczby eksperymentów n.

Niekiedy nie wiemy, kiedy skończymy szukanie. Sytuacja taka zachodzi wówczas, gdy porównamy najwyższą wartość y_k z sąsiednimi y_{k-1} i y_{k-2} i stwierdzimy, że różnica jest mniejsza od pewnego parametru δ. Aby umożliwić sobie takie postępowanie, trzeba stosować złoty podział.

Rozważmy równanie różnicowe

$$F_k = F_{k-1} + F_{k-2} , \tag{17}$$

Jest to równanie liniowe jednorodne o stałych współczynnikach, z warunkami początkowymi $F_0 = F_1 = 1$. Rozwiązanie tego równania jest jawnym wyrażeniem na k-ty wyraz ciągu Fibonacciego.

Równanie charakterystyczne tego równania ma następującą postać:

$$\lambda^2 - \lambda - 1 = 0,$$

$$\lambda_1 = \frac{1+\sqrt{5}}{2} \qquad \lambda_2 = \frac{1-\sqrt{5}}{2} . \tag{18}$$

A oto rozwiązanie ogólne:

$$F_k = C_1 \left(\frac{1+\sqrt{5}}{2}\right)^k + C_2 \left(\frac{1-\sqrt{5}}{2}\right)^k \tag{19}$$

Z warunków początkowych wyznaczamy stałe C_1 i C_2

$$\begin{cases} C_1 + C_2 = 1, \\ C_1 \left(\frac{1+\sqrt{5}}{2}\right) + C_2 \left(\frac{1-\sqrt{5}}{2}\right) = 1, \end{cases}$$

$$C_1 = \frac{1+\sqrt{5}}{2\sqrt{5}} \qquad C_2 = \frac{\sqrt{5}-1}{2\sqrt{5}},$$

$$F_k = \frac{1}{2\sqrt{5}} \left(1+\sqrt{5}\right) \left(\frac{1+\sqrt{5}}{2}\right)^k + \frac{1}{2\sqrt{5}} \left(\sqrt{5}-1\right) \left(\frac{1-\sqrt{5}}{2}\right)^k \tag{20}$$

Oznaczmy teraz $\tau = \frac{1+\sqrt{5}}{2} = 1{,}6180....$, zauważmy, że $\frac{1-\sqrt{5}}{2} = -\frac{1}{\tau}$. Wówczas

$$F_k = \frac{\tau^{k+1} - (-\tau)^{-(k+1)}}{\sqrt{5}} \tag{21}$$

Równanie to nazywa się równaniem Lucasa:

Dla dużych k mamy $F_k \approx \frac{\tau^{k+1}}{\sqrt{5}}$,

natomiast[1] $L_2^* \approx \frac{F_{k-1}}{F_k} = \frac{1}{\tau}$ (22)

Jeśli teraz $L_1 = 1$ podzielimy w stosunku τ, to uzyskamy złoty podział odcinka o długości 1. Postępowanie to możemy kontynuować, dzieląc L_2 w stosunku τ, itd. Końcowy odcinek po n eksperymentach $L_n = \frac{1}{\tau^{n-1}}$. (23)

Stosunek najmniejszego L_n^* do L_n uzyskanego przy szukaniu za pomocą złotego podziału wynosi:

$$\frac{L_n}{L_n^*} = \frac{\tau^{n+1}}{\sqrt{5}\,\tau^{n-1}} = \frac{\tau^2}{\sqrt{5}} = 1.1708. \quad (24)$$

L_n jest tylko o 17% większy od L_n^*.

W ten sposób niewiele tracąc pozbyliśmy się głównej wady szukania za pomocą liczb Fibonacciego.

2.8. Minimalizacja funkcji n-modalnych

Przy optymalizacji funkcji $f(x)$ w przedziale nieokreśloności $g \leq x \leq h$ metodą liczb Fibonacciego zakłada się jednomodalność funkcji celu. Podobnie inne znane algorytmy opierają się na tym założeniu. Wówczas przez porównanie dwóch wartości funkcji $f(x_1)$ i $f(x_2)$ można zmniejszyć odcinek $[g, h]$. W ten sposób można powiedzieć, że metody minimalizacji za pomocą szukania bezpośredniego (direct search) sprowadzają się do porównywania i eliminacji.

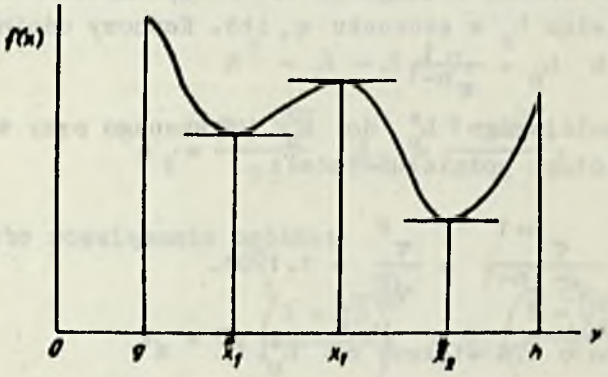
Warto zastanowić się czy metoda eliminacji stosuje się do funkcji n-modalnych. Pod tym pojęciem rozumiem takie funkcje, których przedział $[g, h]$ daje rozbić się na n przedziałów:

[1] Przy pominięciu wyrazu z ε, który zazwyczaj jest o rząd wielkości mniejszy od L_1

$$g \leq x < x_1 \cdots x_{n-1} \leq x \leq h, \quad n > 1,$$

takich, że w każdym z nich f(x) jest jednomodalna. Można wówczas znaleźć ekstrema tej funkcji przez zastosowanie algorytmu Fibonacciego albo złotego podziału w każdym z tych podprzedziałów.

Rozpatrzmy przypadek funkcji dwumodalnej. Przedstawia ją rysunek 8.



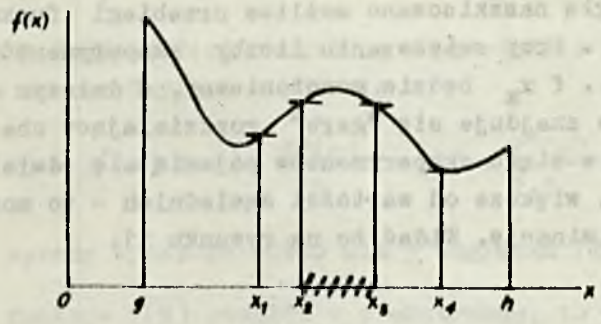
Rys. 8

$\overset{*}{x}_1$, $\overset{*}{x}_2$ - punkty lokalnie optymalne

Zauważmy, że przy czterech obliczeniach wartości funkcji można rozbić przedział nieokreśloności tak, by w dwóch pozostałych do analizy odcinkach, zmiennej niezależnej, f (x) była jednomodalna.

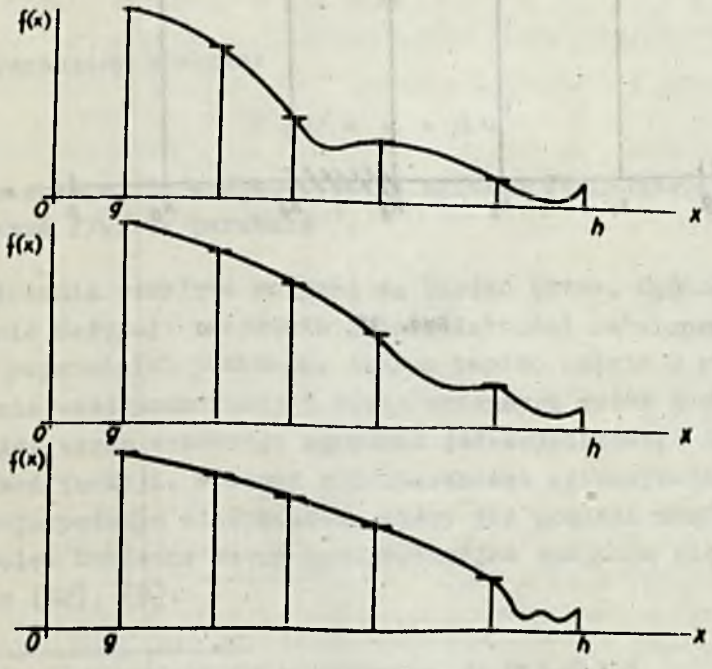
Widać to z rysunku nr 9.

Zakreskowano tutaj tę część pierwotnego przedziału, która podlega eliminacji. Teraz w każdym z pozostałych przedziałów można zastosować optymalny algorytm minimalizacji - metodę Fibonacciego. W ogólnym przypadku nie możemy przypuszczalnie wyznaczyć optymalnego algorytmu według metody minimaxu jak w rozdziale 2.4.



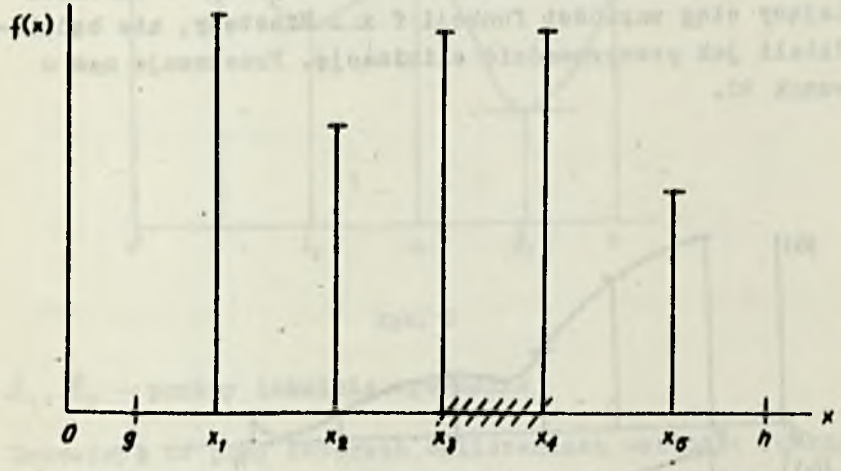
Rys. 9

Przypuśćmy, że w wyniku eksperymentów uzyskano monotonicznie malejący ciąg wartości funkcji f x . Niestety, nie będziemy wiedzieli jak przeprowadzić eliminację. Przekonuje nas o tym rysunek 10.



Rys. 10

Linią ciągłą naszkicowano możliwe przebiegi funkcji dwumodalnej f x . Przy zwiększeniu liczby eksperymentów, gdy ciąg f x_1 ... f x_k będzie monotoniczny, w dalszym ciągu nie wiadomo gdzie znajduje się "garb" rozdzielający oba minima. Kiedy jednak w ciągu eksperymentów pojawią się dwie wartości funkcji f, większe od wartości sąsiednich - to można przeprowadzić eliminację. Widać to na rysunku 11.



Rys. 11

2.9. Interpolacja kwadratowa

Rozważmy funkcję f(x) określoną w przedziale [a, b] rozwijalną w szereg Taylora w punkcie c

$$f(x) = f(c) + f'(c)\,(x-c) + \frac{f''(c)}{2}(x-c)^2 + \frac{f'''(c)}{2.3}(x-c)^3 + \ldots$$

**** są to wyrazy wyższego rzędu niż 3 względem /x-c/

Jeśli funkcja f(x) posiada w c ekstremum, to $f'(c) = 0$.

$$f(x) = f(c) + \frac{f''(c)}{2}(x-c)^2 + \ldots$$

W dostatecznie bliskim sąsiedztwie c, możemy pominąć wyrazy rzędu wyższego niż drugi oznaczając

$$\alpha = f(c) \qquad \beta = \frac{f'(c)}{2},$$

$$u = x - c.$$

Otrzymujemy wówczas:

$$f(u) = \alpha + \beta u^2 \qquad (25)$$

Wynika z tego, że w dostatecznie bliskim sąsiedztwie c możemy uważać f/u/ za parabolę [1].

Założenia przyjęte powyżej są bardzo ostre. Ogólność ich znacznie ustępuje założeniu jednomodalności omówionemu przez nas w poprzednich punktach. Jednak bardzo często w programach szukania ekstremum funkcji wielu zmiennych można przeznaczyć na każdy wyraz sekwencji szukania jednowymiarowego tylko parę obliczeń funkcji. Wówczas z konieczności aproksymujemy funkcję najprostszym wielomianem, który już posiada minimum, czyli parabolę. Dokładne wzory interpolacyjne znajdują się na przykład w [12], [5].

---

[1] Postępowanie opiera się na założeniu, że $f''(c) \neq 0$

2.10. Podsumowanie

Szukanie położenia minimum funkcji jednej zmiennej przeprowadza się albo za pomocą aproksymacji tej funkcji przez funkcję analityczną zazwyczaj będącą wielomianem, albo metodą eliminacji. Metoda eliminacji przez złoty podział lub liczby Fibonacciego jest metodą optymalną w sensie ε -minimaksu, pojęcia wywodzącego się z teorii gier. Dzięki niej na przykład już po 6 eksperymentach można 13 razy zmniejszyć przedział nieokreśloności. W praktycznych algorytmach optymalizacji wielowymiarowej najczęściej stosuje się aproksymację funkcji jednowymiarowej przez wielomian stopnia drugiego. Metoda ta wymaga tylko trzech wyliczeń funkcji, której minimum się poszukuje. Oba podejścia będą sprawdzane przy optymalizacji nastaw regulatorów kotła energetycznego.

3. METODY GRADIENTOWE

3.1. Metoda stycznych równoległych

3.1.1. W p r o w a d z e n i e

Przypuśćmy, że funkcja $f(x_1, \ldots, x_n) = f(\underline{x})$ jest aproksymowana w pobliżu minimum, formą kwadratową:

$$f(\underline{x}) \cong \Phi(\xi) + \frac{1}{2} \sum_{i=1}^{n} \cdot \sum_{j=1}^{n} (x_i - \xi_i)(x_j - \xi_j) f_{ij}(\xi),$$

$$f_{ji}(\underline{x}) = \frac{\partial^2}{\partial x_i \, \partial x_j} f(x).$$

W zwartej formie wzór powyższy przedstawia się następująco:

$$f(\underline{x}) \cong f(\xi) + (\underline{x} - \xi)' \, \underline{A} \, (\underline{x} - \xi).$$

Zakładamy, że A - macierz kwadratowa tej formy, o elementach $a_{ij} = 0{,}25 \, f_{ij}(\xi)$, jest macierzą dodatnio określoną. Założenie to będzie obejmowało swym zasięgiem cały rozdział 3.

Pomijamy więc wyrazy wyższego rzędu niż drugi w rozwinięciu $f(\underline{x})$ wokół punktu $\underline{\xi}$ na szereg Taylora. Dla takiego przypadku, zresztą często spotykanego w praktyce, efektywne metody iteracyjne podali: Shah Buehler Kempthorne [8] oraz Powell [4]. Procedury te nazwano: "Partan" - skrót do słów: parallel tangent - styczne równoległe.

3.1.2. Uzasadnienie i opis metody dla przypadku dwuwymiarowego

Rozpatrzmy formę kwadratową dwu zmiennych

$$f(x_1, x_2) = f(\underline{x}) = \frac{1}{2} a_{11} (x_1 - \xi_1)^2 + a_{12}(x_1 - \xi_1)(x_2 - \xi_2) + \frac{1}{2} a_{22} (x_2 - \xi_2)^2$$

Warstwice otrzymujemy przez równania $f(x_1, x_2) = \text{const}$. Są one w tym przypadku elipsoidami koncentrycznymi. Jeżeli punkty $\underline{x}_1$ i $\underline{x}_2$ leżą na prostej przechodzącej przez minimum, to styczne do warstwic w punktach $\underline{x}_1$ x_2 - π_1 π_2 są równoległe. Przedstawia to rysunek 12.

Dowód tego faktu można np. znaleźć w [8].

Opierając się na tym możemy podać skuteczny algorytm poszukiwania punktu $\underline{\xi}$:

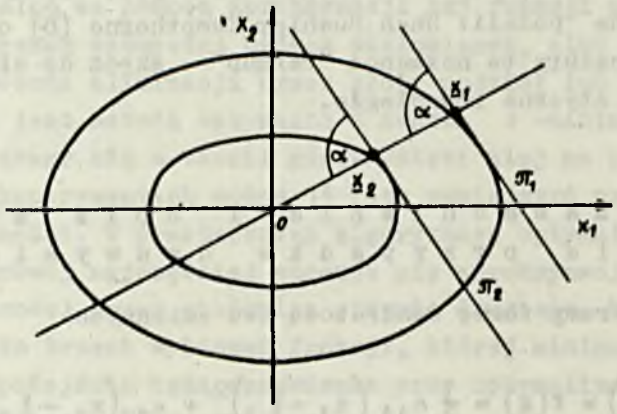
1. W początkowej aproksymacji $\underline{x}_0$ obliczamy styczną π_0.

2. Obliczamy minimum $f(\underline{x})$ wzdłuż jakiegokolwiek kierunku nie leżącego w π_0. Otrzymujemy punkt $\underline{x}_2$.

3. Przez punkt $\underline{x}_2$ prowadzimy prostą $\pi_2 \parallel \pi_0$.

4. Szukamy min. $f(\underline{x})$ wzdłuż π_2. Otrzymujemy punkt $\underline{\xi}_2$.

Uzasadnienie oznaczeń wynika z uogólnienia algorytmu na przypadek wielowymiarowy.



Rys. 12

5. Prowadzimy prostą π_3 przez $\underline{x}_2$ i $\underline{x}_3$.

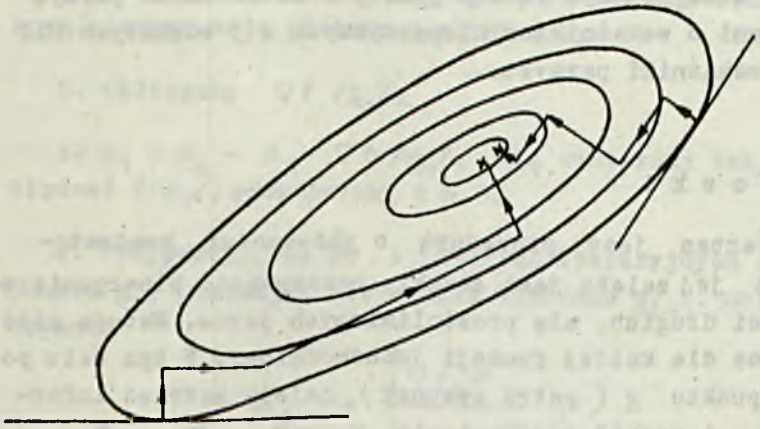
Najniższy punkt na tej prostej jest szukanym minimum. Zauważmy, że kierunek pierwszy, który wybieramy dowolnie może być kierunkiem - ∇f.

Przedstawia to rysunek 13,

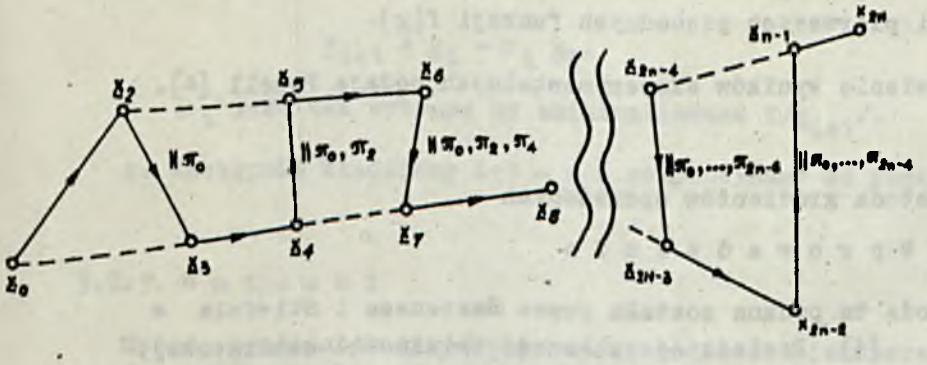
Jak z niego widać, wariant Partan pokonuje wydłużony grzbiet funkcji $f(\underline{x})$ w ciągu 3 iteracji. Klasyczny wariant najszybszego spadku w analogicznej sytuacji wpadłby w charakterystyczne oscylacje, co także pokazuje rysunek 13.

3.1.3. P r z y p a d e k w i e l o w y m i a r o w y

Sekwencją poszukiwań jednowymiarowych, do jakich sprowadza się Partan, dla przypadku n-wymiarowego podaje schemat podany na rysunku 14:



Rys. 13



Rys. 14

Analogicznie jak w przypadku dwuwymiarowym rozróżniamy dwa rodzaje otrzymywania kierunków, w których będziemy prowadzili szukanie jednowymiarowe:

a. prowadzenie stycznych równoległych

b. kierunki przyspieszonego szukania wzdłuż grzbietu.

Kierunki pierwsze uzyskujemy prowadząc styczne przez punkty o wskaźnikach parzystych: P_2 P_4..., P_{2n-2}. Kierunki z drugiej rodziny otrzymujemy, łącząc punkty o wskaźnikach parzystych z punktami o wskaźnikach nieparzystych o 3 większych niż analogiczne wskaźniki parzyste.

3.1.4. W n i o s k i

Metoda Partan jest procedurą o zbieżności kwadratowej. Główną jej zaletą jest szybkie pokonywanie hiperpowierzchni w postaci długich, ale prostoliniowych jarów. Metoda może być adaptowana dla każdej funkcji jednomodalnej. W tym celu po osiągnięciu punktu $\underline{x}$ (patrz rysunek) należy zatrzeć informacje $\underline{x}_0$, π_0, i zrobić podstawienie, $x_i \rightarrow x_{i-1}$ $\pi_i \rightarrow \pi_{i-1}$ $i = 1,\ldots, 2k$ i kontynuować proces aż do osiągnięcia kryterium końca iteracji.

Niestety, metoda ta wymaga złożonego algorytmu i długiego czasu obliczeń na uzyskanie płaszczyzn równoległych. Poza tym zbieżność jest gwarantowana, tylko w przypadku dokładnej znajomości pierwszych pochodnych funkcji $f(x)$.

Omówienie wyników eksperymentalnych podaje Powell [4].

3.2. Metoda gradientów sprzężonych

3.2.1. W p r o w a d z e n i e

Metoda ta podana została przez Hestenesa i Stiefela w 1952 r. [1]. Posiada ona własność zbieżności kwadratowej, przy czym znajduje minimum formy kwadratowej w liczbie kroków iteracyjnych nie większej od liczby zmiennych. Procedura Hestenesa i Stiefela po raz pierwszy zastosowana została do minimalizacji tego rodzaju funkcji, nadaje się jednak do każdej funkcji, której gradient jest łatwo dostępny.

3.2.2. A l g o r y t m

a. Przypuśćmy, że startujemy od punktu $\underline{x}_0$, będącego początkową aproksymacją minimum funkcji f /$\underline{x}$/,

b. Obliczamy $\nabla f\ /\underline{x}_0/$,

c. $\underline{x}_1 = \underline{x}_0 - \lambda_1 \nabla f\ /\underline{x}_0/$, λ_1 wybieramy tak, by zminimalizować $f/x_1/$, podstawiamy i = 1,

d. Przypuśćmy, że po i krokach iteracyjnych doszliśmy do punktu $\underline{x}_i$, posuwając się wzdłuż kierunku $\underline{s}_{i-1}$. Obliczamy współczynnik

$$\beta_{i-1} = \frac{|\nabla f_i|^2}{|\nabla f_{i-1}|^2} .$$

e. Określamy nowy kierunek poszukiwań

$$\underline{s}_i = \nabla f_i + \beta_{i-1}\ \underline{s}_{i-1}.$$

f. Krok jest wykonywany w kierunku:

$$\underline{x}_{i+1} = \underline{x}_i - \alpha_i \underline{x}_i.$$

α_i jest tak wybrane by zminimalizować $f/\underline{x}_{i+1}/$.

g. Następnie kładziemy i+1 = i i rozpoczynamy od punktu d.

3.2.3. W n i o s k i

Metoda gradientów sprzężonych jest jedną z najskuteczniejszych, gdy rozporządzamy dokładną wartością: $\nabla f(x)$. W większości wypadków jest ona trochę mniej szybka niż metoda Davidona. W paru przypadkach [9] [19] okazała się jednak bardziej stabilna. Poza tym trzeba uwzględnić, że posiada ona znacznie prostszy algorytm organizacji procesu szukania [2].

3.3. Metoda Davidona

3.3.1. W p r o w a d z e n i e

Przypuśćmy, że naszym zadaniem jest znalezienie minimum funkcji $f(\underline{x})$, przy czym możemy analitycznie obliczyć gradient tej funkcji $\nabla f(\underline{x}) = \underline{g}(\underline{x})$. Przy założeniu, że $f(\underline{x})$ posiada tylko jeden punkt stacjonarny będący minimum, zadanie minimalizacji sprowadza się do rozwiązania układu równań:

$$\underline{g}(\underline{x}) = \underline{0}. \tag{26}$$

W ogólności równania te mogą być nieliniowe i aby je rozwiązać trzeba posłużyć się metodami iteracyjnymi. Jedną z takich metod jest metoda Newtona. Opiera się ona na następującym równaniu rekurencyjnym

$$\underline{x}_{i+1} = \underline{x}_i - A_i^{-1} \quad \underline{g}_i \tag{27}$$

$\underline{x}_i$ -ita aproksymacja rozwiązania układu równań nieliniowych, a

$$\underline{g}_i = \underline{g}(\underline{x}_i) \tag{28}$$

oznacza gradient w punkcie $\underline{x}_i$

A_i - macierz Jakobiego funkcji wektorowej $\underline{g}(x)$

$$[a_{jk}] = \left[\frac{\partial g_j}{\partial x_k} \right]$$

Niestety, algorytm Newtona posiada słabe strony. Najważniejsza z nich to trudność wyznaczania macierzy A_i. W praktyce można określić $\underline{A}$ tylko numerycznie, co zwiększa tu bardzo czas obliczeń.

Poza tym algorytm Newtona często nie jest zbieżny do rozwiązania problemu[1]. Z tych względów nie możemy go uważać za dobrą praktyczną receptę. Idea tego algorytmu była jednak genezą nowoczesnych metod poszukiwania tzn. quasi-newtonowskich [13].

[1] Warunki zbieżności metody Newtona podane są w monografii Ostrowskiego [11]. Zależą one od początkowej aproksymacji $\underline{x}^0$ rozwiązania układu $\underline{g}(x) = 0$. Musi ona być dostatecznie bliska właściwemu rozwiązaniu, co w praktyce jest trudne do osiągnięcia [21].

Główną cechą tych metod jest rekurencyjne wyznaczanie macierzy Jacobiego. Jedną z metod quasi-newtonowskich szczególnie skuteczną jest metoda Davidona [7].

3.3.2. Klasa metod quasi-newtonowskich

$\underline{x}_i$ jest aproksymacją rozważanego układu równań. Załóżmy ogólnie, że

$$\underline{x}_{i+1} = \underline{x}_i + t_i \underline{p}_i \tag{29}$$

gdzie t_i oznacza pewien skalar. Wybór jego będzie przedyskutowany dalej i zależy od poszczególnej metody.

Przypuśćmy, że macierz B_i jest aproksymacją macierzy Jacobiego po i-tym kroku iteracyjnym. Wówczas przez analogię do algorytmu newtonowskiego możemy zapisać, że

$$\underline{p}_i = - B_i^{-1} \underline{g}_i. \tag{30}$$

Załóżmy następnie, że wektor $\underline{x}$ jest funkcją zmiennej skalarnej t

$$\underline{x} = \underline{x}_i + t \underline{p}_i. \tag{31}$$

$\underline{g}$ jest także funkcją t. Wówczas:

$$\frac{d g_j}{dt} = \sum_{k=1}^{n} \frac{\partial g_j}{\partial x_k} \frac{dx_k}{dt} \qquad j = 1, 2, \ldots, n \tag{32}$$

W postaci wektorowej powyższy układ przedstawia się następująco

$$\frac{d\underline{g}}{dt} = A \underline{p}_i . \tag{33}$$

Jeśli mamy do dyspozycji dobrą aproksymację $\frac{d \underline{g}}{dt}$, to otrzymujemy warunek, który musi spełniać macierz Jacobiego. Ponieważ chcemy aproksymować tę macierz w punkcie $\underline{x}_{i+1}$, więc i w tym punkcie trzeba przybliżyć wartość $\frac{d\underline{g}}{dt}$.

Przypuśćmy, że rozwijamy $\underline{g}(x_{i+1})$ na szereg Taylora

$$\underline{g}_{i+1} = \underline{g}(t_i - s_i) + s \frac{d\underline{g}}{dt} + \underline{r} \cdot s_i p_i \tag{34}$$

Przy założeniu, że:

$$\lim_{\|p_i s_i\| \to 0} \frac{\underline{r}\, p_i\, s_i}{\|p_i s_i\|} = 0 \tag{35}$$

$\|\ldots\|$ norma euklidesowa

pomijamy wyrazy wyższego rzędu i wówczas:

$$\underline{g}_{i+1} - \underline{g}(t_i - s_i) = s_i \, A \, p_i \tag{36}$$

Metody quasi-newtonowskie wykorzystują to równanie do stopniowej aproksymacji macierzy A

$$\underline{g}_{i+1} - \underline{g}(t_i - e_i) = s_i \, B_{i+1} \, p_i \tag{37}$$

Przy założeniu, że pominięte wyrazy rozwinięcia są dostatecznie małe, macierz B_{i+1} będzie dobrym przybliżeniem A.

Wprowadzimy jeszcze kilka oznaczeń. W trakcie algorytmu nie będziemy wyznaczali macierzy $\underline{B}$, następnie odwracali ją by podstawić do wzoru(30). Zamiast tego będziemy od razu obliczać

$$H_i = -B_i^{-1} \tag{38}$$

oznaczmy także różnicę

$$\underline{y}_i = \underline{g}_{i+1} - \underline{g}(t_i - s_i) \tag{39}$$

otrzymujemy z (30) (37) (38) (39)

$$p_i = H_i \underline{g}_i, \tag{40}$$

$$H_{i+1} \underline{y}_i = -s_i \, \underline{p}_i. \tag{41}$$

Równania powyższe nie definiują jednoznacznie macierzy H_{i+1}. Wybór równania macierzowego określa metodę. Równanie to musi być tak dobrane, by były spełnione równości (39)(40)(41). Szczególnie efektywna metoda powstaje gdy równanie ustalimy w postaci:

$$H_{i+1} = H_i - \frac{H_i \underline{y}_i \underline{y}_i^T H_i}{\underline{y}_i^T H_i \underline{y}_i} - s_i \frac{\underline{p}_i \underline{p}_i^T}{\underline{p}_i^T \underline{y}_i} \tag{42}$$

Łatwo sprawdzić, że zachodzi (41):

$$H_{i+1}\, \underline{y}_i = H_i \underline{y}_i - \frac{H_i \underline{y}_i \underline{y}_i^T H_i}{\underline{y}_i^T H_i \underline{y}_i} y_i - s_i \frac{\underline{p}_i \underline{p}_i^T}{\underline{p}_i^T \underline{y}_i} y_i =$$

$$= H_i \underline{y}_i - H_i \underline{y}_i - s_i \underline{p}_i = - s_i \underline{p}_i .$$

3.3.3. A l g o r y t m D a v i d o n a

Podamy teraz algorytm Davidona w zwartej formie. Proces iteracyjny rozpoczynamy mając do dyspozycji $\underline{x}_0$ - początkową aproksymację minimum oraz macierz H_0. Macierz ta może być macierzą jednostkową.

Przypuśćmy, że wykonano i-ty krok, w wyniku czego otrzymano i-tą aproksymację $\underline{x}_i$ $\underline{g}_i$ oraz H_i.

1. Obliczamy kierunek poszukiwania jednowymiarowego:

$$\underline{p}_i = H_i \underline{g}_i . \tag{43}$$

2. Wyznaczamy minimum f wzdłuż linii $\underline{x}_i + t\underline{p}_i$. Otrzymujemy t_i

3. Wykonujemy krok wzdłuż linii $\underline{x}_i + t\underline{p}_i$ o długości t_i. Wówczas $\underline{x}_{i+1} = \underline{x}_i + t_i \underline{p}_i$ (44)

4. Wyznaczamy macierz H_{i+1}: $H_{i+1} = H_i + C_i + B_i$, (45)
gdzie [1]:

$$C_i = - \frac{t_i \, p_i \, p_i^T}{p_i^T \, y_i} \tag{46}$$

$$B_i = - \frac{H_i \, y_i \, y_i^T \, H_i}{y_i^T \, H_i \, y_i} \tag{47}$$

5. Następnie kładziemy $i = i + 1$ i powtarzamy od pkt 1

3.3.4. W n i o s k i

W pracach [7], [13] udowodniono, że algorytm Davidona posiada własność zbieżności kwadratowej [2], czyli wyznacza minimum: $f = \underline{x}^T A \underline{x}$, w skończonej liczbie kroków. Odpowiada to rozwiązywaniu liniowych równań. Poza tym na wielu przykładach [9] [10] przekonano się, że algorytm ten przewyższa pozostałe w wypadku, gdy $g(\underline{x})$ jest łatwo dostępne. Z tego względu w tego rodzaju przypadkach metodę Davidona stosuje się najczęściej. Gdy gradient f nie jest dostępny bezpośrednio, algorytm często zawodzi a nawet staje się niestabilny [19].

---

[1] jako s_i przyjęto t_i. Możliwe jest postępowanie bardziej ogólne.

[2] Według Fletchera i Powella [7] własność zbieżności kwadratowej posiada algorytm, który znajduje z dokładnością do błędów zaokrągleń, minimum funkcji $f(\underline{x}) = \underline{x}^T A \underline{x} + \underline{b}\underline{x} + c$ (A macierz dodatnio określona), wykonując skończoną liczbę liniowych minimalizacji. Określenie to zostało przyjęte w serii publikacji dotyczących minimalizacji funkcji wielu zmiennych [4] [6] [9] [10].

Poza tym termin - zbieżność kwadratowa - występuje w następującym przypadku [20]: Przypuśćmy, że rozważamy proces iteracyjny $x_0, x_1, \ldots, x_n, x_{n+1}$ rozwiązywania metodą Newtona równania $f(x) = 0$ przy warunkach $f'(x) < 0$ $x_0 < r$ $f(r) = 0$ $f''(x) > 0$.

Proces ten posiada własność zbieżności kwadratowej, gdyż zachodzi związek:

$$|x_{n+1} - x_n| \leq k_1 |x_n - x_{n-1}|^2, \text{ gdzie } k_1 = \max_{x_0 \leq \theta \leq r} \left[\frac{f''(\theta)}{f'(\theta)} + 0{,}5 |\varphi''(0)| \right]$$

oraz $\varphi(x) = x - f(x)/f'(x)$

W tej pracy termin - zbieżność kwadratowa - pokrywa się z określeniem Fletchera i Powella.

4. METODY SZUKANIA BEZPOŚREDNIEGO

4.1. Metoda konfiguracji

4.1.1. W p r o w a d z e n i e

Autorami metody są: Hook i Jeeves [8]. Procedura opiera się raczej na intuicyjnych wnioskach niż na zależnościach matematycznych. Należy do algorytmów szukania bezpośredniego, czyli nie wykorzystuje gradientu funkcji $f(\underline{x})$. Pewne zgrubne przybliżenie ∇f uzyskuje się w tym wypadku metodą prób i błędów.

4.1.2. O p i s a l g o r y t m u

Przed rozpoczęciem iteracji ustalamy:

a. pierwszą aproksymację minimum

$$\underline{x}_1 = \sum_{k=1}^{m} \xi_{1k}\, x_1^k$$

$\left\{ \xi_{1k} \right\}$ $1 = 1, \ldots, n$ układ wersorów,

b. układ wektorów zaburzeń

$\underline{\delta}_1^k = \delta_1^k \underline{\xi}_1$ $\quad \delta_1^k$ – pewien z góry dany skalar,

c. k – parametr będący liczbą naturalną.

Ustalamy zmienne bieżące

$i = 1 \qquad \underline{t}_{ij} = \underline{t}_{10} = \underline{x}_1$

$j = 0$

$k = 0$

1. Obliczamy $\underline{t}_{ij}$ przy $j + 1 = j$:

$$\underline{t}_{ij} = \begin{cases} \underline{t}_{ij-1} + \underline{\delta}_j & \text{gdy } f\left(\underline{t}_{ij-1} + \underline{\delta}_j\right) < f\left(\underline{t}_{ij-1}\right), \\ \underline{t}_{ij-1} - \underline{\delta}_j & \text{gdy } f\left(\underline{t}_{ij-1} - \underline{\delta}_j\right) < f\left(\underline{t}_{ij-1}\right) < f\left(\underline{t}_{ij-1} + \underline{\delta}_j\right), \\ \underline{t}_{ij-1} & \text{gdy} \\ f\left(\underline{t}_{ij-1}\right) < \min & \left[f\left(\underline{t}_{ij-1} + \underline{\delta}_j\right),\ f\left(\underline{t}_{ij-1} - \underline{\delta}_j\right)\right]. \end{cases}$$

Obliczenia powtarzamy dla j = 2,..., n. W wyniku otrzymujemy $\underline{t}_{in}$

2. Kładziemy $\underline{x}_{i+1} = \underline{t}_{in}$

3. Sprawdzamy czy $f\left(\underline{x}_{i+1}\right) < f\left(\underline{x}_i\right)$. Jeśli tak, to skok do pkt 5. Jeśli nie, to modyfikujemy wektory zaburzeń:

$\delta_l^k = \dfrac{\delta_l^{k+1}}{2^{k+1}}$ dla l = i,..., n. Następnie kładziemy:

$$\underline{x}_{i+1} = \underline{x}_i \qquad k+1 = k.$$

4. Sprawdzamy czy k ≤ K. Jeśli tak to kontynuujemy od punktu 1. Jeśli nie to stop. $\underline{x}_i$ = szukane minimum.

5. i+1 = i, $t_{io} = 2\underline{x}_i - \underline{x}_{i-1}$. Rozpoczynamy od punktu 1.

4.1.3. W n i o s k i

Metoda konfiguracji była sprawdzana wielokrotnie, a wyniki publikowano. Autorzy zastosowali ją do optymalizacji układów reaktora jądrowego. Poza tym uproszczony wariant tej metody użyto w regulatorze ekstremalnym "Opcon" firmy Westinghouse Corporation [14]. Regulator pomyślnie zdał egzamin przy optymalizacji kolumn destylacyjnych.

Doświadczenia i analiza algorytmu wskazują, że przeprowadza on skuteczne poszukiwanie wzdłuż silnie krzywoliniowych wąwozów. Wadą procedury jest jej powolność. Poza tym w literaturze podano nieregularne powierzchnie takich typów, dla których algorytm nie zdaje egzaminu [8].

4.2. Metoda Rosenbrocka

4.2.1. W p r o w a d z e n i e

Metodę Rosenbrocka [3] można uważać za modyfikację popularnej metody Gaussa-Seidla. Różnica polega na tym, że w metodzie Gaussa-Seidla układ współrzędnych pozostaje niezmienny w ciągu całego procesu iteracyjnego, a w algorytmie Rosenbrocka po każdym kroku iteracyjnym dokonujemy obrotu układu współrzędnych.

4.2.2. O p i s a l g o r y t m u

Przypuśćmy, że wyjściowym układem współrzędnych jest zbiór wektorów ortogonalnych $\{\underline{e}_i\}$, $i = 1, \ldots, n$. Przypuśćmy, że długości tych wektorów są znormalizowane $\|\underline{e}_i\| = 1$. Oznaczmy ten zbiór przez $\{\underline{e}_i\} \equiv \{\underline{\xi}_i^0\}$, $i = 1, \ldots, n$. W każdym kierunku przeprowadzamy eksperymenty, które dostarczają nam informacji o charakterze zmian funkcji $f(\underline{x})$.

W tym miejscu rozchodzą się drogi metody Rosenbrocka i metody Davisa Swanna Campeya [10]. W tej drugiej procedurze oblicza się za pomocą interpolacji parabolicznej przybliżoną lokalizację minimum funkcji $f(\underline{x} + \lambda\, \underline{\xi}_i)$ przy zmianach λ. Otrzymujemy wówczas wartość $\lambda = d_i$.

W metodzie Rosenbrocka dla każdego kierunku $\underline{\xi}_i$ wyznacza się parametr d_i. Na początku ustalamy $d_i = 0$. Pierwszy eksperyment polega na wykonaniu kroku o długości $e_i = e_i^0$ w kierunku $\underline{\xi}_i$. Jeśli krok był pomyślny, to w następnej iteracji $d_i = d_i + e_i$, $e_i = e_i^0 \alpha$, $\alpha > 1$. Jeśli krok był niepomyślny to $e_i^1 = \beta e_i^0$ $0 < \beta < 1$.

Parametry e_i^0, α, β są ustalane przed startem programu. Wybór dokładny opiera się na intuicji. Ustalanie wielkości d_i przerywamy, gdy dla wszystkich kierunków po kroku pomyślnym nastąpił krok niepomyślny.

1. Po wyznaczeniu zbioru parametrów $\{d_i\}$ można przystąpić do obrotu układu współrzędnych. Rotacja ta opiera się na procedurze Grama-Schmidta [1]. $\{\underline{\xi}^0_i\}$ jest to wyjściowy zbiór wersorów ortogonalnych.

2. Pierwszy krok polega na wyznaczeniu zbioru wektorów $\{\underline{A}_i\}$

$$\underline{A}_1 = d_1 \underline{\xi}^0_1 + d_2 \underline{\xi}^0_2 + \ldots + d_n \underline{\xi}^0_n ,$$

$$\underline{A}_2 = d_2 \underline{\xi}^0_2 + \ldots + d_n \underline{\xi}^0_n ,$$

$$\vdots$$

$$\underline{A}_n = d_n \underline{\xi}^0_n .$$

3. Po wyznaczeniu układu wektorów $\{\underline{A}_i\}$ $i = 1,\ldots,n$ określamy nowe wersory $\underline{\xi}^1_1$ ortogonalne względem siebie

$$\underline{B}_1 = \underline{A}_{11},$$

$$\underline{\xi}^1_1 = \frac{\underline{B}_1}{\|\underline{B}_1\|} \quad \text{normalizacja},$$

$$\underline{B}_2 = \underline{A}_2 - \underline{A}_2 \underline{\xi}^1_1 \underline{\xi}^1_1 ,$$

$$\underline{B}_n = \underline{A}_n - \sum_{j=1}^{n-1} \underline{A}_n \underline{\xi}^1_j \underline{\xi}^1_j .$$

Następnie procedurę się powtarza od pkt 1.

4.2.3. W n i o s k i

Metoda Rosenbrocka i zbliżona do niej metoda Davisa Swanna Compeya (w skrócie DSC) była wielokrotnie sprawdzana w praktyce [8] [9] [10]. Dużą zaletą obu metod jest to, że w procesie szukania nie wykorzystują one wartości pochodnych

funkcji f. Poza tym eksperymenty wskazały na dużą niezawodność tych metod. Bardzo rzadko natrafiono na przypadki, gdy algorytm Rosenbrocka nie odnajdywał minimum. Metoda DSC okazała się pod tym względem słabsza.

Jeśli uniwersalność jest zaletą procedury Rosenbrocka, to szybkość stanowi jej słabą stronę. Zwykle trzeba wykonać dużą liczbę wyliczeń $f(\underline{x})$, by znaleźć się w minimum, dużą w porównaniu z metodami Powella lub Davidona. Trzeba jednak podkreślić, że metoda Rosenbrocka szczególnie skutecznie pokonuje nieregularności w postaci parabolicznie zakrzywionych jarów.

Przykładem takiego ukształtowania powierzchni może być funkcja Rosenbrocka:

$$f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2 .$$

4.3. Metoda Powella-Zangwilla

4.3.1. W p r o w a d z e n i e

W 1964 roku Powell [5] podał bardzo efektywną metodę szukania bezpośredniego, wykorzystującą pojęcie kierunków sprzężonych. Autor podał dowód zbieżności kwadratowej tego algorytmu. W dowodzie tym tkwił błąd, który zauważył Zangwill. Zmodyfikował on algorytm Powella tak, że ten nowy algorytm zwany procedurą Zangwilla posiada własność zbieżności kwadratowej [6].

4.3.2. M e t o d a P o w e l l a

Niech $\underline{\xi}1, \ldots, \underline{\xi}n$ będzie układem wersorów wyjściowych, a niech $\underline{x}_0$ będzie początkową aproksymacją minimum $f(\underline{x})$.

1. Dla r = 1, 2,..., n obliczyć λ_r minimalizujące

$$f\left(\underline{x}_{r-1} + \lambda_r \underline{\xi}r\right)$$ i położyć $\underline{x}_r = \underline{x}_{r-1} + \lambda_r \underline{\xi}r$

2. Dla r = 1, 2,..., n-1 zastąpić $\underline{\xi}r$ przez $\underline{\xi}r+1$ i zastąpić $\underline{\xi}n$ przez $\underline{x}_n - \underline{x}_0$.

3. Wybrać minimalizujące $f\left(\underline{x}_n + \underline{x}_n - \underline{x}_o\right)$ i zastąpić $\underline{x}_o$ przez $\underline{x}_o + \left(\underline{x}_n - \underline{x}_o\right)$ i skoczyć do punktu 1.

Metoda Powella opiera się na następujących twierdzeniach i założeniach: Niech $f(\underline{x}) = \underline{x}' A \underline{x} + \underline{b} \underline{x} + \underline{c}\, x \in E^n$ będzie dodatnio określoną formą kwadratową. Forma taka na pewno posiada jedno minimum w punkcie $\underline{x}^* = -0.5\, A^{-1} \underline{b}$. Dwa kierunki $\underline{p}$ i $\underline{q} \in E^n$ nazywamy wzajemnie A-sprzężonymi, gdy $\underline{p}' A \underline{q} = 0$. Zachodzą wówczas dwa twierdzenia:

Twierdzenie 1. Jeżeli $\underline{q}_1, \ldots, \underline{q}_m$ • $m \leq n$ są zwrotami wzajemnie sprzężonymi, to minimum funkcji kwadratowej $f(\underline{x})$ w przestrzeni m-wymiarowej zawierającej punkt $\underline{x}_o$, można osiągnąć drogą tylko jednokrotnego szukania minimum $f(\underline{x})$ wzdłuż zwrotów $\underline{q}_1, \ldots, \underline{q}_m$.

Twierdzenie 2. Jeżeli $\underline{x}_o$ jest lokalnym minimum $f(\underline{x})$ wzdłuż zwrotu $\underline{q}$ oraz $\underline{x}_1$ jest także lokalnym minimum wzdłuż zwrotu $\underline{q}$, to kierunek $\underline{x}_1 - \underline{x}_o$ jest kierunkiem sprzężonym do $\underline{q}$.

Stosując te dwa twierdzenia, Powell wyprowadził swój algorytm minimalizacji funkcji kwadratowej. Nie wziął jednak pod uwagę, że zwroty $\underline{q}_1, \ldots, \underline{q}_m$ $m \leq n$ mogą wyznaczać tylko podprzestrzeń E^m przestrzeni E^n i znalezione minimum będzie odpowiadało najmniejszej wartości $f(\underline{x})$ w E^m, a nie w całej przestrzeni E^n. Przykładem funkcji, której algorytm Powella nie może zminimalizować, jest:

$$f(x, y, z) = (x - y + z)^2 + (-x + y + z)^2 + (x + y - z)^2.$$

4.3.3. A l g o r y t m Z a n g w i l l a

Należy tak zmodyfikować algorytm Powella, by zwroty sprzężone $\underline{q}_1 \ldots, \underline{q}_n$ mogły być rozpięte na całej przestrzeni E^n, co oznacza, że zbiór $\underline{q}_1$ i = 1,..., m musi być zbiorem wektorów liniowo niezależnych.

Niech $\underline{c}_r$, r = 1,..., n będzie początkowym zbiorem wersorów

1. Niech będzie dany punkt $\underline{x}_n^0$ oraz zbiór n wersorów $\{\underline{\xi}_r^1\}$ $r = 1,\ldots, n$. Obliczyć λ_n^0 minimalizujące $f_1\left(\underline{x}_n^0 + \lambda_n^0\ \underline{\xi}_n^1\right)$ i położyć $\underline{p}_{n+1}^0 = \underline{p}_n^0 + \lambda_n^0\ \underline{\xi}_n^1$. Ustalamy $t = 1$ i rozpoczynamy iterację od $k = 1$.

2. Iteracja k: $\underline{p}_{n+1}^{k-1}$ $\underline{\xi}_r^k$, $r = 1,\ldots, n$ są dane. Znaleźć α minimalizujące $f\left(\underline{p}_{n+1}^{k-1} + \alpha\ \underline{o}_t\right)$.

3. Określić

$$t \leftarrow \begin{cases} t+1 & \text{gdy } 1 \leqslant t \leqslant n, \\ 1 & \text{gdy } t = n. \end{cases}$$

Jeżeli $\alpha \neq 0$ to $\underline{p}_o^k = \underline{p}_{n+1}^{k-1} + \alpha\ \underline{o}_t$. Jeżeli $\alpha = 0$ powtórz procedurę od 2. Jeśli kroki 2 i 3 będą powtarzane n razy pod rząd, to $\underline{p}_{n+1}^{k-1}$ jest szukanym minimum.

4. Dla $r = 1,\ldots, n$ oblicz λ_r^k minimalizujące

$$f\left(\underline{p}_{r-1}^k + \lambda_r^k\ \underline{\xi}_r^k\right)$$ i zdefiniuj

$$\underline{p}_r^k = \underline{p}_{r-1}^k + \lambda_r^k\ \underline{\xi}_r^k .$$

Niech

$$\underline{\xi}_{n+1}^k = \left(\underline{p}_n^k - \underline{p}_{n+1}^{k-1}\right) \Big/ \left\| \underline{p}_n^k - \underline{p}_{n+1}^{k-1} \right\| .$$

Określić λ_{n+1}^k minimalizujące $f\left(\underline{p}_n^k + \lambda_{n+1}^k\ \underline{\xi}_{n+1}^k\right)$ i położyć $\underline{p}_{n+1}^k = \underline{p}_n^k + \lambda_{n+1}^k\ \underline{\xi}_{n+1}^k$. Następnie zdefiniować $\underline{\xi}_r^{k+1} = \underline{\xi}_{r+1}^k$ $r = 1,\ldots, n$, a potem skoczyć do 2 zastępując k+1 przez k.

Zauważmy, że kroki 2 i 3 przeszukują wartości $f(\underline{x})$ zawsze wzdłuż tych samych kierunków. Zapewniają one nastąpienie momentu końca procesu. Punkty dalsze to organizacja procesu szukania według metody Powella. Ponieważ przestrzeń E^n jest rozpięta na zbiorze $\underline{o}_i$ $i = 1,\ldots, n$, więc zbieżność dla funkcji kwadratowej jest zapewniona.

Uzasadnienie wyboru tej metody jako podstawowego algorytmu optymalizacji kotła energetycznego podaję w podsumowaniu.

5. OGRANICZENIA

5.1. Wprowadzenie

Dla metod gradientowych skutecznym sposobem optymalizacji z ograniczeniami jest metoda gradientu z projekcją. Metoda ta nie daje się adaptować na bezpośrednie metody szukania. Można wówczas pozbyć się ograniczeń przez odpowiednie transformacje zamieniające problem z ograniczeniami w przestrzeni X na problem bez ograniczeń w przestrzeni U.

5.2. Transformacje

Rozpatrzmy problem minimalizacji funkcji $f(x) = f(x_1, x_2, x_3)$ przy ograniczeniach $0 \leq x_1 \leq x_2 \leq x_3$. Rozważmy dla tego przypadku transformacje:

$$x_1 = U_1^2,$$
$$x_2 = U_1^2 + U_2^2,$$
$$x_3 = U_1^2 + U_2^2 + U_3^2.$$

Wówczas $f(x_1, x_2, x_3) = \Phi(U_1, U_2, U_3)$, z tym, że szukamy minimum funkcji $\Phi(U_1, U_2, U_3)$ bez żadnych ograniczeń w przestrzeni U.

W problemach programowania nieliniowego mogą być użyteczne następujące przekształcenia:

1. $x_1 = U_1^2$,
2. $x_1 = e^{U_1}$,
3. $x_1 = |U_1|$,
4. $x_1 = \sin^2 U_1$,

$$5.\ x_i = \frac{e^{U_i}}{e^{U_i} + e^{-U_i}} .$$

Pierwsze trzy zapewniają, że x_i jest nieujemne, podczas gdy ostatnie dają się stosować gdy $0 \leqslant x_i \leqslant 1$. W praktyce najczęściej spotykamy ograniczenia w postaci $g_i \leqslant x_i \leqslant h_i$, co oznacza, że obszar dopuszczalny sprowadza się do n-wymiarowego sześcianu.

Za pomocą transformacji

$$x_i = g_i + \left(h_i - g_i\right) \sin^2 U_i,$$

$$x_i = g_i + \left(h_i - g_i\right) \frac{e^{U_i}}{e^{U_i} + e^{-U_i}},$$

pozbywamy się ograniczeń. W pierwszym z nich mamy do czynienia z rozwiązaniem okresowym. W przestrzeni U^n pojawi się ciąg minimów. W praktyce jednak kroki algorytmu szukania nie są tak duże, by prowadziły do skoków od jednego minimum do drugiego. Wszystkie wymienione transformacje są tego rodzaju, że otoczenie punktu $x \in E^n$ (E^n - przestrzeń euklidesowa) przechodzi w otoczenie punktu $u \in U^n$. Przy czym U^n jest to przestrzeń otrzymana przez transformację przestrzeni E^n.

Ogólnie rzecz biorąc, transformacje stosuje się w problemie optymalizacji nie tylko by pozbyć się ograniczeń. Proces szukania jest najłatwiejszy, gdy warstwice wokół minimów są zbliżone do okręgów. Przez odpowiednie zmiany skal można przekształcić np. warstwice nieregularne na kołowe.

Większość metod szukania zakłada, że $f(\underline{x})$ w pobliżu minimum ma szybko zbieżny szereg Taylora. Należy więc wybierać takie zmienne niezależne, przy których założenie to jest spełnione. Ilustruje to przykład [8]:

$$y = \varphi(p, t) = y_o \left[1 - a\left(p - p_o\right)^2 - b\left(t - t_o\right)^2\right],$$

$$y = \psi(\pi, t) = y_0 \left[1 - a\left(e^{\pi} - p_0\right)^2 - b\left(t - t_0\right)^2\right],$$

$$e^{\pi} = p.$$

Wygodniej rozpatrywać jest problem w przestrzeni p, t niż w przestrzeni π, t, gdyż w tej pierwszej otrzymujemy bardziej regularne wyrażenie na y.

5.3. Przykłady

Powróćmy do zastosowania transformacji, by pozbyć się ograniczeń. Opiszemy kilka przykładów napotykanych w literaturze [9] [15].

Obliczmy minimum $f = -x_1 x_2 x_3$ przy ograniczeniach:

$$0 \leq x_1 \leq 42,$$
$$0 \leq x_2 \leq 42,$$
$$0 \leq x_3 \leq 42,$$
$$0 \leq x_1 + 2x_2 + 2x_3 \leq 72.$$

Rozwiązanie: $f = 3,456$ przy $x_1 = 24$, $x_2 = 12$, $x_3 = 12$.

Możliwą transformacją jest układ

$$x_1 = X_1,$$
$$x_2 = X_2,$$
$$x_3 = \frac{1}{2}\left(X_3 - X_1 - 2X_2\right),$$
$$0 \leq X_1 \leq 42,$$
$$0 \leq X_2 \leq 42,$$
$$0 \leq X_3 \leq 72.$$

Następnie pozbywamy się ograniczeń przez zastosowanie transformacji

$$X_i = g_i + \left(h_i - g_i\right) \sin^2 Y_i.$$

Zastosowanie transformacji nie spełnia jeszcze warunku $x_3 > 0$. Przy $x_1 > 0$ $x_2 > 0$ program szukający minimum nie może wyjść poza ograniczenia.

Przydatność tego rodzaju transformacji badano w [8]. Okazało się, że przez ich stosowanie problem szukania znacznie się skrócił dzięki metodzie Powella, która właśnie może działać tylko przy braku ograniczeń.

5.4. Ograniczenia przy metodach gradientowych

Dla metod gradientowych istnieją trzy efektywne sposoby pokonywania problemu ograniczeń:

a. funkcja kary [16] [18],
b. metoda zygzakowata [18],
c. metoda gradientu z projekcją [16].

Metoda zygzakowata [18] jest metodą bardzo powolną. Z tego względu prawie nie stosuje się jej w praktycznych algorytmach. Metoda gradientu z projekcją wymaga dość skomplikowanego algorytmu i działa pewnie przy liniowych ograniczeniach. Metoda funkcji kary może być stosowana zarówno do metod gradientowych jak i do procedur szukania bezpośredniego. Wadą tej metody jest jej pewna nieokreśloność w doborze stałych charakteryzujących algorytm.

6. PODSUMOWANIE

Powszechnie przyjętym kryterium wyboru algorytmu optymalizacji statycznej jest czas pracy procedury. Zamiennym wskaźnikiem jakości jest ilość obliczeń funkcji $f(\mathbf{x})$. Wynika to z tego, że wyliczanie wartości $f(\mathbf{x})$ trwa przeważnie dużo dłużej niż organizacja procesu szukania.

W literaturze podano w pozycjach [2] [9] [10] wyniki eksperymentów z różnymi programami przy szukaniu ekstremum hiperpowierzchni:

a. Parabolicznej doliny Rosenbrocka (1960):

$$f = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2,$$

$$x_0 = (-1{,}2;\ 1{,}0).$$

b. Wąwozu hiperbolicznego Fletchera i Powella (1963):

$$f = 100\left[\left(x_3 - 10\theta\right)^2 + (r - 1)^2\right] + x_3^2,$$

$$x_1 = r \cos 2\pi\theta,$$

$$x_2 = r \sin 2\pi\theta \qquad \underline{x}_0 = (-1{,}0{,}0).$$

c. Funkcji Powella (1962):

$$f = \left(x_1 + 10x_2\right)^2 + 5\left(x_3 - x_4\right)^2 + \left(x_2 - 2x_3\right)^4 + 10\left(x_1 - x_4\right)^4,$$

$$\underline{x}_0 = (3;\ -1;\ 0;\ 1).$$

d. Funkcji o zmiennej liczbie argumentów Powell-Fletcher (1963):

$$f = \sum_{i=1}^{n}\left[E_i - \sum_{j=1}^{n}\left(A_{ij}\sin\alpha_j + B_{ij}\cos\alpha_j\right)\right]^2,$$

A_{ij} B_{ij} są liczbami pseudoprzypadkowymi z przedziału $[-100;\ 100]$.

Warunki początkowe wybrano w postaci $\overset{*}{\alpha}_i + \delta_i$, $i = 1,\ldots, n$ gdzie α_i reprezentowane są przez liczby rzeczywiste z przedziału $[-\pi, \pi]$ a δ_i przez liczby pseudoprzypadkowe z przedziału $\left[\frac{-\pi}{10}, \frac{\pi}{10}\right]$.

Eksperymenty te wskazują generalnie na przewagę algorytmów posiadających własność zbieżności kwadratowej. Spośród tej klasy programów wyróżnić jeszcze można algorytm Davidona i Powella-Zangwilla. Są one w przeważającej liczbie przypadków szybsze od pozostałych.

Jak zaznaczono we wstępie celem opracowania był wybór odpowiedniej metody optymalizacji regulacji kotła energetycznego $f(\underline{x})$. Kryterium wyboru stanowi liczba kroków iteracyjnych algorytmu aż do chwili znalezienia ε-nowego otoczenia strefy minimum $f(\underline{x})$. Ponieważ o funkcji celu kotła energetycznego nie wiele wiadomo (poza tym, że jest ciągła) więc do optymalizacji wybrano metodę Powella, która nie musi aproksymować gradientu funkcji f. Poza tym studia Broydena i Barda wskazują na przypadki, w których algorytm Davidona jest niestabilny [19].

Eksperymenty wykazały jednocześnie duże zalety procedury Rosenbrocka. Chociaż algorytm ten nie posiada własności zbieżności kwadratowej i jest metodą powolną, to jednak bardzo pewną. We wszystkich rozpatrywanych przypadkach osiągano za pomocą tego programu minimum funkcji wielu zmiennych. Opierając się na tym, postanowiono sprawdzić przydatność algorytmu Rosenbrocka przy optymalizacji kotła energetycznego, w wypadku gdyby zawiodła metoda Powella-Zangwilla. Z metod nie uwzględnionych w tym opracowaniu na uwagę zasługuje jeszcze metoda simpleksu [14], adaptowana do przypadków nieliniowych. Jak sprawdzono metoda ta bardzo szybka dla dwóch i trzech zmiennych niezależnych, zupełnie zawodzi przy zwiększeniu liczby argumentów funkcji $f(\underline{x})$ [9].

Literatura

[1] BECKMAN F.S.: The Solution of Linear Equations by the Conjugate Gradient Method. Mathematical Methods for Digital Computers, Wiley 1960.

[2] FLETCHER R., REEVES C.M.: Function Minimization by Conjugate Gradients. The Computer Journal, V. 7, N° 2, 1964.

[3] ROSENBROCK H.H.: An Automatic Method for Finding the Greatest or the Least Value of a Function. The Computer Journal, Vol. 3, 1960.

[4] POWELL M.J.D.: An Iterative Method for Finding Stationary Values of a Function of Several Variables. The Computer Journal, Vol. 5, 1962-1963.

[5] POWELL M.J.D.: An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives. The Computer Journal, Vol. 7, 1964-1965.

[6] ZANGWILL W.: Minimizing a Function Without Calculating Derivatives. The Computer Journal, Vol. 10, 1967.

[7] FLETCHER R., POWELL M.J.D.: A Rapidly Convergent Descent Method for Minimization. The Computer Journal, V. 6, 1963-1964.

[8] WILDE D.: Optimum Seeking Methods. Prentice Hall, 1964.

[9] BOX M.J.: A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems. The Computer Journal, Vol. 9, 1966-1967.

[10] FLETCHER R.: Function Minimization without Evaluating Derivatives - a Review. The Computer Journal, Vol. 9, 1966-1967.

[11] OSTROWSKI A.M.: Solution of Equations and Systems of Equations. Academic Press, 1960.

[12] BOOTH A.D.: Numerical Methods. 1957.

[13] BROYDEN G.G.: A Class of Methods for Solving Nonlinear Simultaneous Equations. Mathematics of Computation, Vol. 19, 1965.

[14] IWACHNIENKO A.G. i in.: Promyślennaja kibernetika, 1966.

[15] BOX M.J.: A New Method of Constrained Optimization and a Comparison with Other Methods. The Computer Journal, Vol. 8, 1965.

[16] KELLEY H.J.: Method of Gradients. Leitmann Optimization Techniques, 1963.

[17] HADLEY G.: Nonlinear Programming, 1965.

[18] KAGAN B.M., TER-MIKAELJON T.M.: Rešenije inżynieryjnych zadač, 1964.

[19] BARD Y.: On a Numerical Instability of Davidon - Like Methods. Mathematics of Computation, July 1968.

[20] BELLMAN R., KALABA R.: Quasilinearization and Nonlinear Boundary-Value Problems. The RAND Corporation, 1965.

[21] BROYDEN C.: A Class of Methods for Solving Nonlinear Simultaneous Equations. Mathematics of Computation, vol. 19, 1965.

ПРОСМОТР МЕТОДОВ ПОИСКА ЭКСТРЕМУМА ФУНКЦИЙ МНОГИХ ПЕРЕМЕННЫХ

Резюме

Работа является просмотром алгоритмов поиска найбольшей и наименьшей величины функций определённых на подмножествах Евклидова пространства. Рассмотрены лишь детерминистские алгоритмы, использующие величины функции и величины её градиента, численно заданные или учитивающие только величины функции. Обсуждённые методы применяются часто для статической оптимизации промышленных объектов. Более детально рассмотрена в статье статическая оптимизация котла электростанции.

A REVIEW OF METHODS OF SEARCHING MANY VARIABLE FUNCTIONS OF EXTREMUMS

Summary

The paper is a review of algorithms of searching the biggest and the smallest functions value determined on subsets of Euclidian space. Only deterministic algorithms are considered that are making use of the function value and its gradient value given numerically, or taking into account only the function values. The discussed methods are often used to static optimisation of industrial objects. The paper considers the static optimation of a power boiler more in detail.

Warszawa, ul. Krzywickiego 34
K.M.

Cena zł 40,-