ОБЪЕДИНЕННЫЙ ИНСТИТУТ ЯДЕРНЫХ ИССЛЕДОВАНИЙ

Лаборатория теоретической физики

Hans Jürgen Kaiser

E - 1131

TRACE CALCULATION ON ELECTRONIC COMPUTER

Дубна 1962 г.

Hans Jürgen Kaiser *

E - 1131

TRACE CALCULATION ON ELECTRONIC COMPUTER

* On leave from Institut für Theoretische Physik Technische Universität Dresden, German Democratic Republic.

A b s t r a c t

A computer programme has been devised which calculates traces of γ-matrices by algebraic means, so that the results are obtained in closed form. The programme deals with traces of the types

$$\mathrm{tr}\, \hat{a}_0 \hat{a}_1 \ldots \hat{a}_{n-1}, \quad \mathrm{tr}\, \gamma^5 \hat{a}_0 \hat{a}_1 \ldots \hat{a}_{n-1}, \quad \mathrm{tr}\, g^{\mu\nu} \hat{a}_0 \hat{a}_1 \ldots \gamma^\mu \ldots \hat{a}_k \ldots \gamma^\nu \ldots \hat{a}_{n-3}.$$

The input expression may be of a complicated arithmetical form containing parentheses ad libitum. The programme also performs multiplication of traces of the types

$$\mathrm{tr}\, \hat{a}_0 \hat{a}_1 \ldots \hat{a}_{n-1}\, \mathrm{tr}\, \hat{b}_0 \hat{b}_1 \ldots \hat{b}_{m-1}, \quad g^{\mu\nu}\, \mathrm{tr}\, \hat{a}_0 \hat{a}_1 \ldots \gamma^\mu \ldots \hat{a}_{n-2}\, \mathrm{tr}\, \hat{b}_0 \hat{b}_1 \ldots \gamma^\nu \ldots \hat{b}_{m-2}.$$

Г.Ю. Кайзер

ВЫЧИСЛЕНИЕ ШПУРОВ ИЗ ГАММА-МАТРИЦ НА ЭЛЕКТРОННОЙ ВЫЧИСЛИТЕЛЬНОЙ МАШИНЕ

А н н о т а ц и я

Составлена программа для электронной вычислительной машины для вычисления шпуров из γ-матриц алгебраическим образом, так что результаты получаются в замкнутой форме. Программа обрабатывает шпуры видов $\mathrm{tr}\, \hat{a}_0 \hat{a}_1 \ldots \hat{a}_{n-1}$, $\mathrm{tr}\, \gamma^5 \hat{a}_0 \hat{a}_1 \ldots \hat{a}_{n-1}$, $\mathrm{tr}\, g^{\mu\nu} \hat{a}_0 \hat{a}_1 \ldots \gamma^\mu \ldots \hat{a}_k \ldots \gamma^\nu \ldots \hat{a}_{n-3}$. Вводимое выражение может иметь сложный арифметический вид, включая любую комбинацию скобок. Кроме этого программа умножает шпуры видов

$$\mathrm{tr}\, \hat{a}_0 \hat{a}_1 \ldots \hat{a}_{n-1}\, \mathrm{tr}\, \hat{b}_0 \hat{b}_1 \ldots \hat{b}_{m-1}, \quad g^{\mu\nu}\, \mathrm{tr}\, \hat{a}_0 \hat{a}_1 \ldots \gamma^\mu \ldots \hat{a}_{n-2}\, \mathrm{tr}\, \hat{b}_0 \hat{b}_1 \ldots \gamma^\nu \ldots \hat{b}_{m-2}.$$

1. Introduction

The calculation of traces of Dirac γ-matrices is an elementary but often very tedious step in the evaluation of Feynman diagrams, so that it is desirable to have the work done by an electronic computer. There are in principle two means to realize that. Firstly traces can be calculated purely numerically by choosing determined values for each component of all contributing four-vectors, multiplying by the corresponding γ-matrices and then multiplying successively the resulting 4 x 4 -matrices /1/. The second way is to adapt the rules for traces reduction /2/ which are common in hand calculation to a computer programme. Both methods have their advantages and disadvantages.

The numerical method gives only a point to point construction (e.g. of a scattering cross section), whereas it is often desirable to have a closed formula. The numerical approach is further not very suitable to account for known orthogonality restrictions between some four- vectors (e.g. transversal photons). The main advantage of the numerical method lies in its comparatively high speed in calculating very long traces, as it requests for each additional operator only one additional matrix multiplication, with the result that the computer time rises linearly with the length n of the trace.

On the other hand, the algebraic method gives closed expressions which often allow for explicit cancellation of terms, deals easily with orthogonality restrictions, but becomes unworkable for very long traces ($n > 18$) . As the number of permutations to be evaluated rises as $(n-1)!!$ we meet here with an exponential rise of the effort with the number of operators, if the special structure of the trace does not allow a previous reduction.

The aim of this paper is to discuss the principles of a programme of the second (algebraic) kind.

2. Principles of the programme

The general expression of a trace

$$tr\ A_{ik} \equiv \frac{1}{4} \sum_{i=1}^{4} A_{ii} \tag{1}$$

through the scalar products of the four-vectors a_μ reads /2/ (n even)

$$\begin{gathered} tr\ \hat{a}_0 \hat{a}_1 \ldots \hat{a}_{n-1} = \sum \delta_p \, (a_{i_0} a_{i_1})(a_{i_2} a_{i_3}) \ldots (a_{i_{n-1}} a_{i_n}) \\ \delta_p = \operatorname{sign} \begin{pmatrix} 0 & 1 & 2 & \ldots & n-1 \\ i_0 & i_1 & i_2 & \ldots & i_{n-1} \end{pmatrix} \qquad \hat{a} \equiv \gamma^\mu a_\mu \qquad (ab) \equiv a^\mu b_\mu \end{gathered} \tag{2}$$

where the summation runs over all pairings of the four-vectors. The sign is determined by the parity of the corresponding permutation.

The adaption of this formula to the computer is done as follows;
We relate to each scalar product (ab) (excluding those which are known to vanish) a group of four binary units on a determined place in the computer's memory. The values 0000 to 1001 then characterize the powers $(ab)^0$ to $(ab)^9$ of that scalar product in the result. The pairings are determined by the corresponding permutations of the initial operator combination. The permutations for itselves are generated successively for each length of the trace (see Appendix 1) The contribution of a permutation to the result is simply found by group – wise addition of the numbers coding the powers of each scalar product.

Example: For the trace

$$\mathrm{tr}\ \hat{a}\ \hat{b}\ \hat{c}\ \hat{d}\ \hat{e}\ \hat{a}\ \hat{e}\ \hat{b} \tag{3}$$

the permutation

$$-\ 01\ \ 23\ \ 46\ \ 57 \tag{4}$$

leads to the following product of scalar products

$$(ab)\,(cd)\,(ee)\,(ab). \tag{5}$$

Let further the places ..0001 be reserved for (ab) and 0001... for (cd) and let be $(ee) = 1$ (so that the powers of (ee) need no explicit mention). Then the result of the permutation (4) is represented by the two lines (results block):

$$\begin{cases} -1 & \\ \ldots & 0001 \quad 0010 \end{cases} \tag{6}$$

the first denoting the numerical coefficient, the second the powers 1 resp. 2 in the expression $(cd)^1\ (ab)^2$. (If necessary several lines can be reserved for the powers).

The successively arising result blocks are arranged in a results field, such that each power line shows up only one time. and the incorporation of a further block with the same powers is realized by merely adding the numerical factors.

If the given expression contains several traces, their results are in the same manner superimposed in the results field. This behaviour can be blocked by marking the different traces with different general coefficients denoted by different numbers in an additional stroke of the results block.

If the trace contains solitary γ^μ-matrices combined through $g^{\mu\nu}$ there must be introduced symbolic operators $\hat{A}\ \hat{B}$ .. for the γ-matrices bearing the indices of the first, second ... metric tensor.

Example : The trace

$$\mathrm{tr}\ g^{\kappa\lambda}\, g^{\mu\nu}\ \hat{a}\ \hat{b}\ \gamma^\mu\ \gamma^\kappa\ \hat{c}\ \gamma^\nu\ \hat{d}\ \gamma^\lambda \tag{7}$$

is written as

$$\mathrm{tr}\ \hat{a}\ \hat{b}\ \hat{B}\ \hat{A}\ \hat{c}\ \hat{B}\ \hat{d}\ \hat{A}$$

which can be treated according to the general scheme discussed above with the additionally programmed rules for interpreting the scalar products:

$$\ldots\ (aA)\ \ldots\ (cA)\ \ldots\ =\ \ldots\ (ac)\ \ldots \tag{8}$$

following from

$$g^{\mu\nu}\,(a\gamma^\mu)\,(c\gamma^\nu)\ =\ (ac) \tag{9}$$

and

$$\ldots\ (AA)\ \ldots\ =\ 4 \tag{10}$$

following from

$$\mathrm{tr}\ g^{\mu\nu}\gamma^{\mu}\gamma^{\nu} = 4 \tag{11}$$

In order to speed up the calculation, there have been taken two means:

1. Before going into the calculations described above, the programme tries to reduce the length of the trace by anticommuting identical operators in adjacent positi ons, as far as this can be done without generating additional terms (i.e. as far as the anticommutation involves only operators with zero scalar product) and then using the formula

$$\mathrm{tr}\ \ldots\ \hat{a}\,\hat{a}\ \ldots = (a^2)\ \mathrm{tr}\ \ldots \tag{12}$$

2. If the current permutation led to a zero scalar product the programme jumps to that permutation which as first does not contain that scalar product.

If the trace includes γ^5-matrices, they are first of all anticommuted in the first position and pair-wise cancelled, such that we must discuss as new case only that of one γ^5-matrix in front of operators a .

The γ^5-traces are handled in the following way:

The programme takes successively 4 operators from those backing the γ^5 and combines they (if they are all different as the result is zero otherwise) into a pseudo-scalar .

The residual operators are then handed over to the main programme. The election of the 4 operators is directed by special ' γ^5-permutations', generated by a method described in Appendix 2. If we e.g. apply the γ^5-permutation

$$1235 \quad 0467 \tag{13}$$

onto the trace

$$\mathrm{tr}\ \gamma^5\,\hat{a}\,\hat{b}\,\hat{c}\,\hat{d}\,\hat{e}\,\hat{f}\,\hat{g}\,\hat{h} \tag{14}$$

we find the splitting

$$\varepsilon^{\kappa\lambda\mu\nu}\, b_\kappa\, c_\lambda\, d_\mu\, f_\nu\ \ \mathrm{tr}\ \hat{a}\,\hat{e}\,\hat{g}\,\hat{h} \tag{15}$$

into pseudo-scalar and residual trace.

The pseudo-scalar isfurther ordered – thereby eventually changing sign – into rising order of the numbers coding the contributing operators and then represented as a particular line in the results block.

If the trace is multiplied from outside by a further pseudo-scalar, the product of the two pseudo-scalars will be expressed through a combination of scalar products.

Care has been taken to simplify the input of the trace. For this end it was devised especially a programme which disentagles any arithmetic expression containing parentheses ad libitum (see Appendix 3).

A simplified block diagram of the programme calculating traces is given in Fig. 1.

In another variant, the programme can be used to multiply traces, represented by results fields described above.

Thereby contractions through $g^{\mu\nu}$ e.g.

$$g^{\mu\nu} \, tr \, \hat{a} \, \hat{b} \, \gamma^{\mu} \hat{c} \, tr \, \gamma^{\nu} \hat{d} \tag{16}$$

can be carried out.

Appendix 1: The γ^{μ} - permutations

The calculation of a trace containing n γ^{μ}—matrices demands the successive generation of (n − 1) !! permutations corresponding to the possible pairings of the n operators together with the determination of the sign of these permutations.

Let e.g. be n = 6. Then the 5!! = 15 permutations may be arranged as follows:

+ 01 23 45	+ 01 35 24
− 01 24 35	:
+ 01 25 34	
− 01 34 25	+ 05 14 23

The block diagram of an algorithm doing the work for any n is given in Fig.2. (The shortcut, possible if a scalar product vanishes is not indicated). $A_0 \cdots A_p \cdots A_{n-1}$ denote the addresses containing the $0^{th} \ldots p^{th} \ldots (n-1)^{th}$ figure of the permutation; [A] means the contents of A.

Appendix 2: The γ^5 —permutations

If the trace contains a γ^5—matrix in front of n γ^{μ}—matrices, one must construct $\binom{n}{4}$ permutations determining the different possibilities to extract 4 operators out of the n for combination with the γ^5 into a pseudoscalar.

In the case n = 6 they read:

+ 0123 45
− 0124 35
+ 0125 34
+ 0134 25
:
+ 2345 01

The γ^5 —permutations are generated by the algorithm described in Fig.3.

Appendix 3: Decomposition of parenthetic expressions

The input expression may consist of operators, numerical factors, general coefficients, pseudoscalars, and arithmetical operators +, −, (, and). The programme accepts all combinations of arithmetical operators which make sense.

The input expression is first treated by an initial transformation, which includes translation of the numerical factors into the binary system, replacing the " − " by + (−1), and supplementing parentheses around solitary groups multiplying parenthetic expressions (e.g. a (b + c) → (a) (b + c)).

The result of the transformation is then successively worked up according to the block diagram Fig.4.

To each group between two subsequent arithmetical operators it is related a binary digit in the "scale", which contains in the initial state everywhere "1" and may extend over several addresses if necessary. At each entry the search for "1" in the scale starts at far left. At each crossing arithmetical operators the "pin", a binary "1" surrounded by zeros is shifted by one place to the right. Furthermore at each parenthesis, some counters are operated, necessary to determine the range of the parenthesis considered (these counters are not explicitely shown in Fig.4).

The method used has the advantage, to demand no memory space for the accumulation of intermediary results. Each additional arithmetical operator requires only one further bit in the scale.

References

1. G.I.Kopylov. Preprint JINR D-821 (1961).
2. J.M.Jauch, F.Rohrlich: The Theory of Photons and Electrous. Addison-Wesley, 1955, p. 436.

Received by Publishing Department
on November 24, 1962.

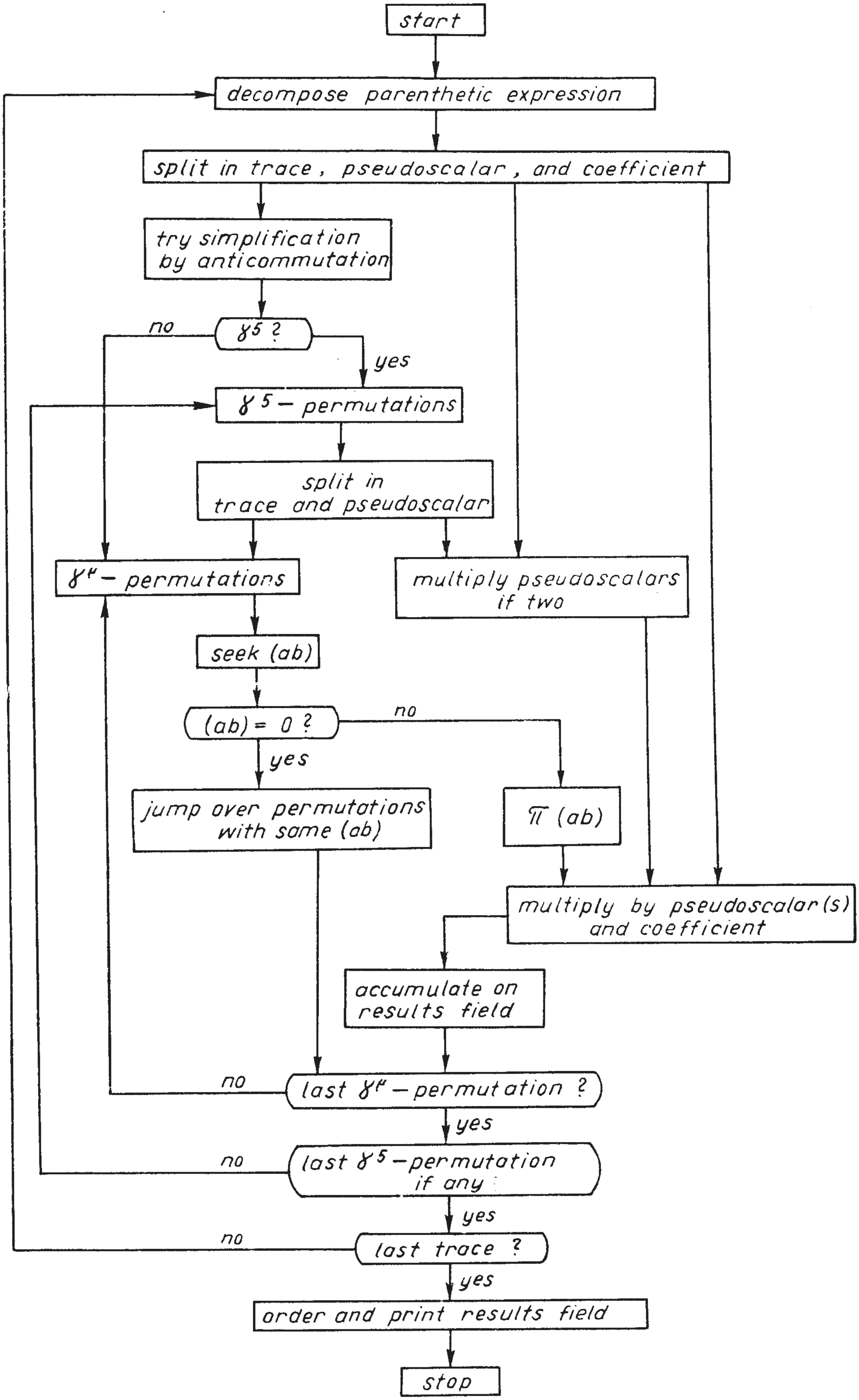
start
decompose parenthetic expression
split in trace, pseudoscalar, and coefficient
try simplification by anticommutation
γ^5 ?
no
yes
γ^5 – permutations
split in trace and pseudoscalar
γ^μ – permutations
multiply pseudoscalars if two
seek (ab)
(ab) = 0 ?
no
yes
jump over permutations with same (ab)
π (ab)
multiply by pseudoscalar(s) and coefficient
accumulate on results field
no
last γ^μ – permutation ?
yes
no
last γ^5 – permutation if any
yes
no
last trace ?
yes
order and print results field
stop

Fig. 1.

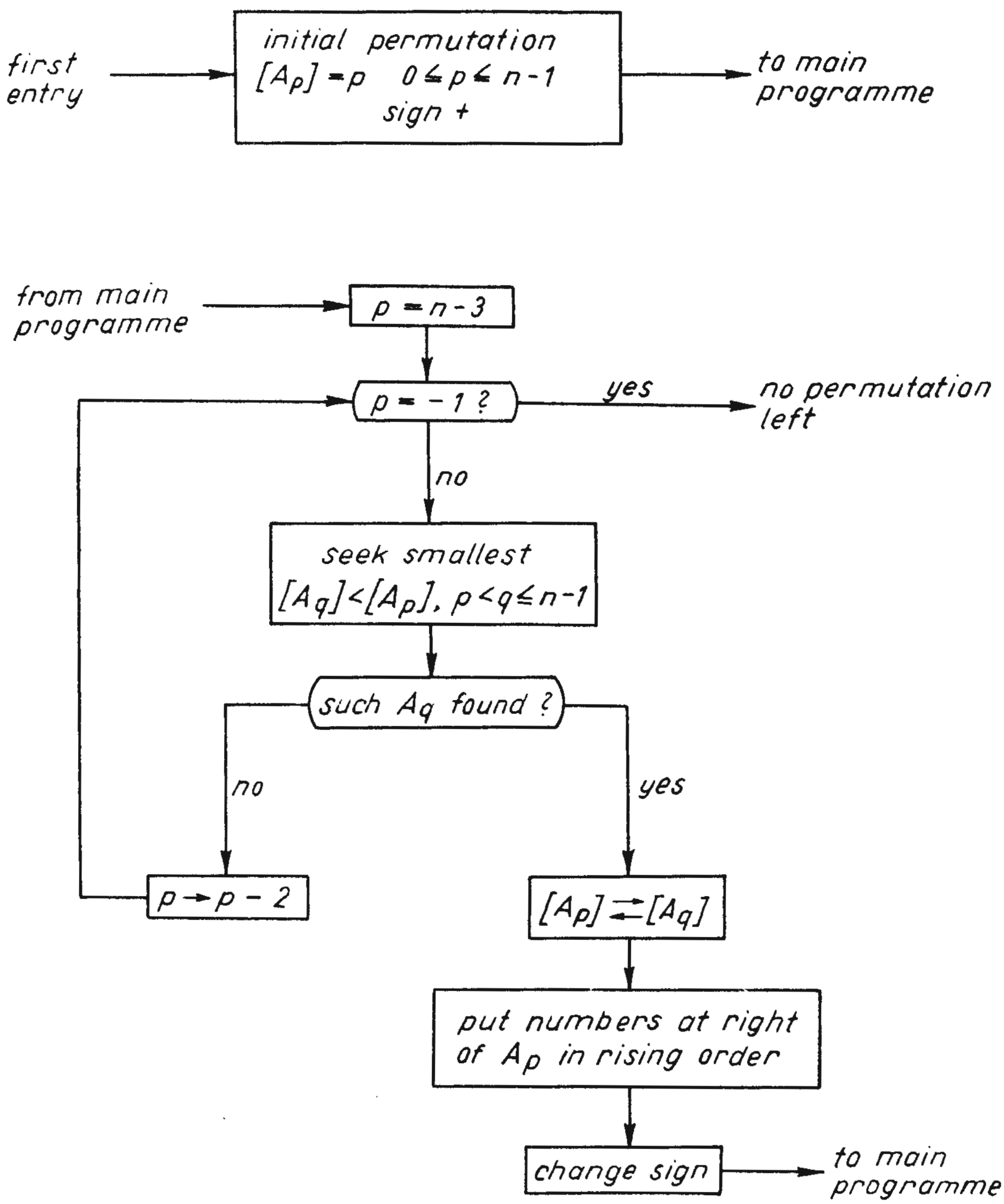
first entry
initial permutation
[A_p] = p   0 ≤ p ≤ n-1
sign +
to main programme
from main programme
p = n-3
p = -1 ?
yes
no permutation left
no
seek smallest
[A_q] < [A_p], p < q ≤ n-1
such A_q found ?
no
yes
p → p - 2
[A_p] ⇄ [A_q]
put numbers at right of A_p in rising order
change sign
to main programme

Fig. 2.

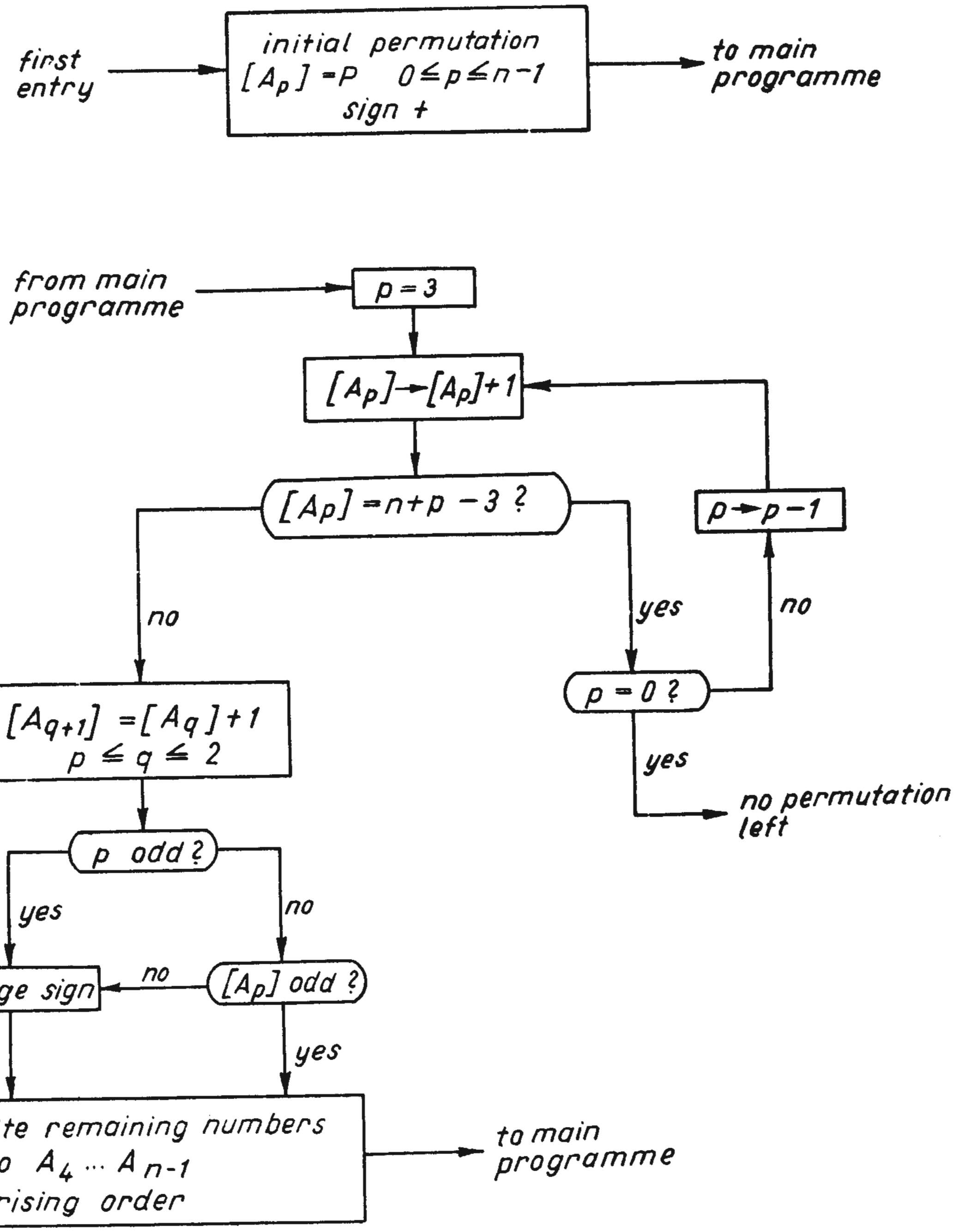
first entry
initial permutation
[A_p] = p   0 ≤ p ≤ n-1
sign +
to main programme
from main programme
p = 3
[A_p] → [A_p]+1
[A_p] = n+p-3 ?
no
yes
p → p-1
no
p = 0 ?
yes
no permutation left
[A_{q+1}] = [A_q]+1
p ≤ q ≤ 2
p odd ?
yes
no
change sign
no
[A_p] odd ?
yes
write remaining numbers
into A_4 ··· A_{n-1}
in rising order
to main programme

Fig. 3.

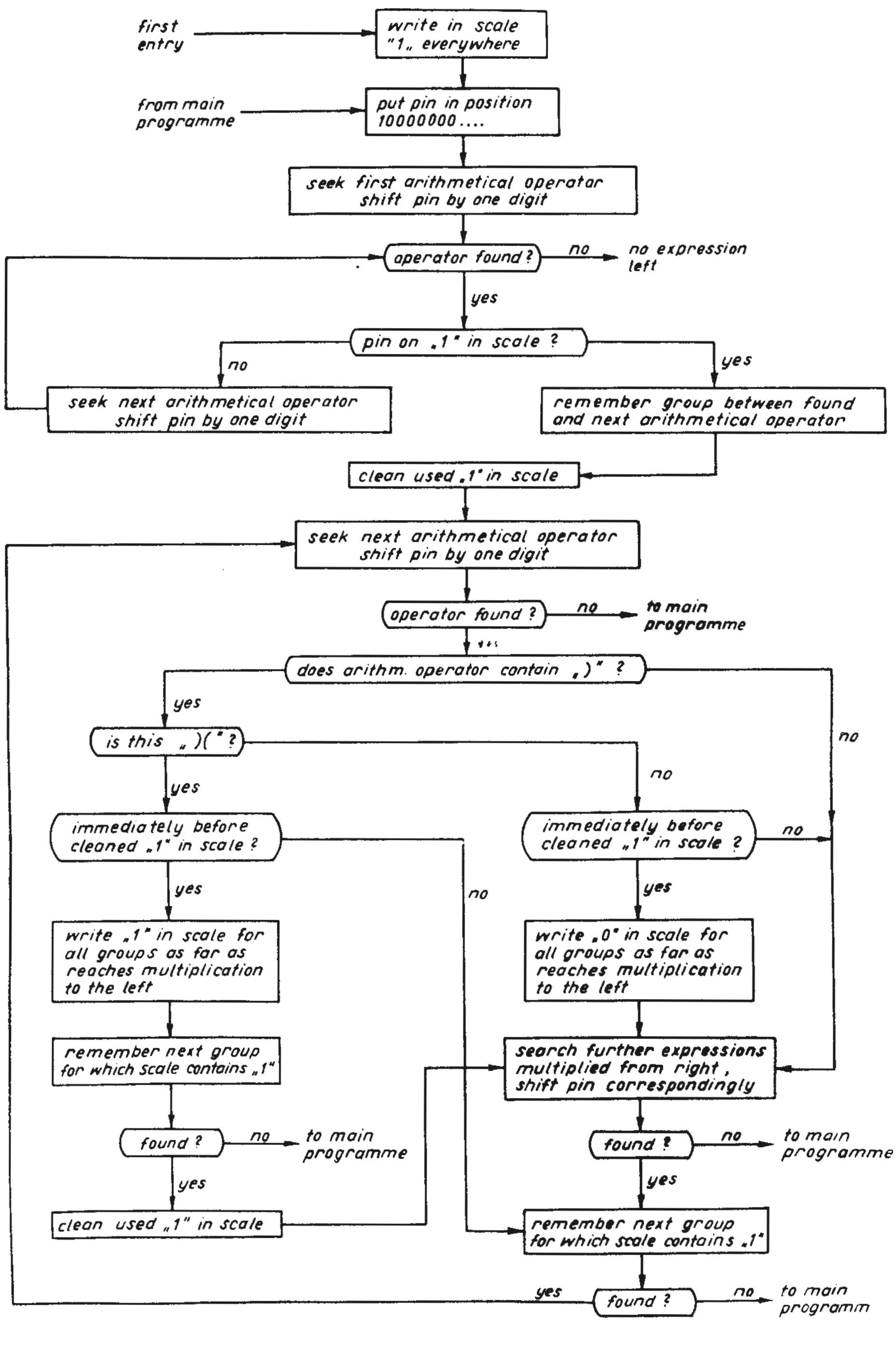
first entry
write in scale "1„ everywhere
from main programme
put pin in position 10000000....
seek first arithmetical operator shift pin by one digit
operator found?
no
no expression left
yes
pin on „1" in scale ?
no
yes
seek next arithmetical operator shift pin by one digit
remember group between found and next arithmetical operator
clean used „1" in scale
seek next arithmetical operator shift pin by one digit
operator found ?
no
to main programme
does arithm. operator contain „)" ?
yes
no
is this „)(" ?
yes
no
immediately before cleaned „1" in scale ?
immediately before cleaned „1" in scale ?
no
no
yes
yes
write „1" in scale for all groups as far as reaches multiplication to the left
write „0" in scale for all groups as far as reaches multiplication to the left
remember next group for which scale contains „1"
search further expressions multiplied from right, shift pin correspondingly
found ?
no
to main programme
found ?
no
to main programme
yes
yes
clean used „1" in scale
remember next group for which scale contains „1"
yes
found ?
no
to main programm

Fig.4.