MATHEMATICS

L. V. KANTOROVICH

1957

SovietRxiv

Abstract

Full Text

MATHEMATICS

L. V. KANTOROVICH

ON A CERTAIN MATHEMATICAL SYMBOLISM CONVENIENT FOR CARRYING OUT COMPUTATIONS ON MACHINES

(Presented by Academician S. L. Sobolev, 26 X 1956)

Introduction. At the present time the question of carrying out, on electronic digital machines, the most diverse kinds of mathematical work is very urgent –not only numerical computations, but also other types: analytic calculations and transformations, logical analysis, analysis and synthesis of relay circuits, processing of computational plans and programs. This raises the task of developing practically convenient and universal methods for entering tasks of this kind into machines and unified methods for executing these tasks.

For this purpose it is essential to develop the following three questions:

1) The creation of a uniform symbolism. The requirements imposed on such a symbolism are: a) universality–the possibility of using one and the same symbolism for describing mathematical tasks of different content; b) precision and unambiguity of the symbolism; c) compactness (small capacity of the record in relation to the information contained in it); d) convenience of handling it on a machine and sufficient clarity for ordinary perception.

2) The development of convenient forms for describing the arrangement of material introduced into and stored in the machine in the course of its processing.

3) The development of unified methods for processing a mathematical task, regardless of its specific content. The identification of general types of operations encountered in such processing, the clarification of their properties, and the preparation of programs that practically carry them out on machines. In other words, the development of a kind of algebra of such operations–machine algebra.

1. Schemes. Various mathematical objects (in finite number) will be denoted, independently of their nature (numbers, vectors, functions, operations, etc.), by integer numbers. As the basic notation describing a given mathematical task, process, or theory, we adopt an abstract scheme that determines the relations

and connections among the various objects. The scheme consists of a number of lines of the form:

$$k_1 = (k_1', k_1'', k_1''')$$

$$k_2 = (k_2', k_2'', k_2''')$$

$$\ldots\ldots\ldots$$

Here $k_1, k_1', \ldots$ are numbers. Each line says that some object is determined by the objects written on the right; for example, the object k_1 is determined by the objects k_1', k_1'', k_1'''.

Suppose that there is a "field" of objects of any nature in which, to certain ordered systems of its objects, called compatible, there corresponds an element of the same field, in the notation: $d = (a, b, c)$.

It should be said that it is useful to include operators among the elements of the field, so that in the group considered as a "field," the elements will be, in addition to the basic elements, also the signs $+$ and $-$; $d = (+, a, b)$ means $d = a + b$.

Let some element of the field be assigned to each number. We shall say that we have a solution of a scheme if, when the corresponding elements of the "field" are substituted into the equations of the scheme in place of the numbers, the equalities of the scheme are satisfied.

We shall call a scheme explicit if its rows admit such an ordering that: 1) the resultant elements, i.e., the numbers standing on the left in some row, are not repeated, and 2) in the text of no row (i.e., on the right) do there occur resultant elements of lower rows. (In other words, the elements of such a scheme in principle allow a certain process of successive determination.)

The elements of an explicit scheme naturally turn into a partially ordered set; namely, one element must be regarded as $>$ another if, under any arrangement of the type described above, it turns out to be lower. The minimal elements of this set, i.e., the numbers not appearing as resultant ones, we shall call basic. It is clear that if an explicit scheme is solvable for given basic elements, then it is solvable in only one way.

For example, for the function

$$y = \sqrt{x+a}(\cos\sqrt{x+a} + e^x) + \frac{e^x}{\sqrt{x+a}}$$

the notation in the form of an (explicit) scheme has the form

$1 = (+, 2, 3)$ $6 = (+, 7, 8)$ $5 = (+, 9, 10)$ $3 = (:, 10, 4)$
$2 = (\times, 4, 5)$ $7 = x$ $9 = (\cos, 4)$
$4 = (\sqrt{\ }, 6)$ $8 = a$ $10 = (\exp, 7)$

The operations should also have been denoted by numerals. Let us note the compactness of the notation in the form of a scheme—expressions that occur repeatedly are not given in expanded form, unlike ordinary formula notation.

A scheme may be regarded as a functional transformation of the group of basic elements (not necessarily including all of them; the remaining ones are regarded as parameters) into the group of resultant ones.

Subschemes are naturally singled out from a given scheme—by the same notation one specifies the dependence of certain groups of intermediate arguments on others.

Conversely, one may consider the operation of joining several schemes, with the corresponding coordination of the numbering of their elements; in particular, the successive superposition of schemes, the degree of a scheme, a cumulative operation on objects constructed according to the given scheme, etc.

It is advisable to extend the initial field of elements, including in it sets, schemes composed of its elements, and also certain operations, for example the computation of a scheme from a given system of elements, the taking of a subscheme, the joining of schemes, and the symbols corresponding to these operations. In particular, it is useful to introduce the frequently used operations of branching and iteration.

2. Simplifications and transformations of schemes

We shall give several concrete operations for processing abstract schemes that arise in various uses of schemes.

1) To determine whether a given scheme is explicit, and also to find its elements from known basic ones, special programs for ordering a scheme are used. The most convenient is ordering by logical dependence, introducing the minimal number of numbers needed to find the given quantity.

2) **Simplification operation.** If the relations defining several numbers are identical (i.e., the right-hand sides of the corresponding lines are identical), then only one of all these lines should be retained, and in the remaining lines the eliminated numbers should be replaced by the retained one.

3) If there are identities connecting variables, i.e., arbitrary or to some extent arbitrary numbers of objects, then they can be written as schematic identities—rules for replacing one definite kind of subscheme by another. A program can be compiled for transforming schemes in accordance with these rules.

As an example, one may point to transformation rules connected with the properties of various operators (commutativity, associativity, distributivity, etc.). The distributivity of the operators $+$, $\times$, for example, is written in the form of the schematic equality $S_1 = S_2$, where

$$S_1 \begin{cases} 0 = (\times, 1, 4) \\ 4 = (+, 2, 3) \end{cases} \qquad S_2 \begin{cases} 0 = (+, 4, 5) \\ 4 = (\times, 1, 2) \\ 5 = (\times, 1, 3) \end{cases}$$

With the use of such programs on the basis of the given laws of operations (schematic identities), schemes can be simplified (the detection of additional identities among numbers), some operator can be raised or lowered, an operator can be eliminated if a rule is given for its interchange with others and its value for the reference elements, certain arguments can be raised upward as far as possible, the notation of a scheme can be shortened by introducing designations for repeatedly occurring schemes, and so on.

If some associative operator is applied repeatedly, it is useful to combine such numbers into a system (vector), which corresponds to the introduction of signs of the type $\sum a_i$. This facilitates simplifications within such a group.

Let us note that the application of this kind of transformation program in schematic notation is convenient because the added and replacing lines can be written in arbitrary order, and also because of the possibility of carrying out the transformation locally, i.e., considering the nearest logical connections of the given numbers without involving the complete formulas by which they are expressed.

3. **Analytic transformations.** Programs of general schematic transformations can be used in carrying out various analytic calculations on machines.

Thus, from the scheme of a function one computes the scheme of its derivative (it suffices to lower the operator by means of the rules for its interchange with functional operators). The schematic notation of the derivative can be used for further differentiation and other transformations and for obtaining the numerical value of the derivative. The same methods and programs can also be used in other cases: simplification of a rational function (division upward), separation of the real and imaginary parts, etc. They can also be used for transformation, simplification, reduction to canonical form, and other kinds of mathematical problems, for example: a) transformation of computational plans for more convenient computation by them: raising upward expressions depending on many parameters, raising complex operations, eliminating repeated computation of expressions, etc.; b) transformation of logical and set-theoretic formulas: moving quantifiers outward, eliminating certain operations (negation, complementation); c) analysis of relay-contact circuits; d) formal analysis of deductive theories (finding parallelisms; parts admitting convenient construction; etc.).

4. **Computations.** Schematic symbolism can be applied to the performance of various kinds of computations on a machine (although, formally, computation cannot be opposed to transformation, since it too represents a certain transformation of information).

A universal program can be constructed for computing the values of an explicit scheme in a certain field. This program, relying on information about the basic elements, their content and arrangement, and also on known machine algorithms (programs) that establish (from the information about the elements) the compatibility of their groups and its value in the field, makes it possible to construct the defining information for all elements of the scheme and information (references) about their arrangement.

Possible areas of application of such a general computational process are:
a) carrying out computations directly according to a mathematical plan written in the form of an explicit scheme, without actually compiling a detailed working program ("prorab" —a foreman program, operating according to the computational plan);
b) the objects of computation—the output data—may be taken to be not the results themselves, but programs; thus, such a computational process can generate a computation program according to a given plan, i.e., in principle, a programming program can be constructed in this way;
c) for computation, more precisely, for simplifying transformations, of expressions containing variables and numbers;
d) computation of the truth values of logical expressions; this can be done more economically than in the primitive way—over all combinations;
e) various computations in topology, group theory, and the theory of structures.

It should be regarded as feasible in the very near future on machines, alongside ordinary numerical computations, to carry out fully automatically a definite mathematical method for solving a problem, with the performance of all necessary numerical and analytical expansions, as well as logical analysis. Alongside its use in approximate computations and applied mathematics, machine analysis should also find systematic application in a number of so-called theoretical areas of mathematics.

Leningrad Branch of the V. A. Steklov Mathematical Institute

Received 24 IX 1956

Note: Figure translations are in progress. See original paper for figures.