STAT

UNCLASSIFIED
CLASSIFICATION

AIR INTELLIGENCE INFORMATION REPORT

COUNTRY OR AREA REPORT CONCERNS
USSR

DATE OF REPORT
17 Sep 1959

SUBJECT (*Descriptive title. Use individual reports for separate subjects*)
Soviet Document: "Solutions of Engineering Problems on Automatic Computers" by B. M. Kagan and T. M. Ter-Mikaelyan

STAT

2. The translation consists of two title pages, one page of translator's notes, 294 pages of text, five pages of Supplement #1: "List of Operations of the Computer M-3", four pages of Supplement #2: "List of Operations of the Computer 'Ural'", and four pages of reference, a total of 310 pages.

STAT

One Copy of orig rpt in Russian (UNCL)

STAT

B. M. KAGAN and T. M. TER-MIKAELYAN

SOLUTIONS OF ENGINEERING PROBLEMS ON AUTOMATIC DIGITAL COMPUTERS

GOSENERGOIZDAT

(STATE POWER ENGINEERING PUBLISHING HOUSE)

B.M. KAGAN and T.M. TER-MIKAELYAN

SOLUTION OF ENGINEERING PROBLEMS ON AUTOMATIC DIGITAL COMPUTERS

STATE POWER ENGINEERING PUBLISHING HOUSE

MOSKVA 1958 LENINGRAD

Remarks: 1. Some components of the formulas or/and complex symbols in the original text have been found to be outright illegible. Such components have been replaced by question marks.

2. This translation has been made by three translators in succession, which has resulted in a slight difference in the choice of English Terms. So, for example, the Russian word "razryad" has been translated at first as "order", then as "digit" and then as "column".

The book deals with questions concerning the application of automatic digital computers (ATsVM) for engineering calculations and research.

It explains the functioning principle of an automatic digital computer as well as the preparation and programming procedures for solving mathematical tasks.

Examples of the application of automatic digital computers for solving engineering problems were examined (investigation of transition processes in long-distance power lines, calculations of the stability of automatic control systems, investigations concerning the critical speed of turbogenerator rotors, calculations for a series of electric motors for lowest cost). Although all examples concern themselves with electric devices, the tasks presented have the character of general engineering problems.

The book was written for scientific workers, engineers, postgraduate students and senior students of higher educational institutions.

Authors: Boris Moiseyevich Kagan and Teodor Mikhaylovich Ter-Mikaelyan

Solution of Engineering Problems by Automatic Digital Computers.

Editor: V.M. Kurochkin Technical Editor: G.Ye. Larionov

STAT

PREFACE

The theoretical study of most problems encountered in various branches of engineering is reduced to mathematical problems the rigorous solution of which either cannot be found or is so complicated that it is difficult to use it in calculations. For the solution of these problems various approximate methods have been proposed, which make it possible to obtain the answer in numerical form. However, the great number of arithmetical operations necessary for applying the methods mentioned, made them up to recently practically inapplicable.

High-speed electronic computers which have appeared during the last 10 or 15 years considerably expanded the range of solvable problems, and at present have penetrated into the practice of not only scientific but also into engineering-technical investigations. Taking into consideration difficulties arising with engineers who turn to the computers' aid in their activities, the authors made an attempt to compose a book providing basic information on the operational principles of the automatic digital computers and on the possibilities of their employment for engineering investigations and calculations. The book aims to give the reader a sufficient guide for independent programming and formulation of engineering problems for the computers admitting a numerical method of solution.

The first four chapters of the book acquaint the reader with the design of automatic digital computers and the methods of programming mathematical problems. The lectures of Prof. A.A. Lyapunov, delivered by him at the Moscow University in 1954-1955, and also the train of ideas developed by Prof. A.A. Lyapunov and his mathematicians group during the seminar on mechanical mathematics in the MGU (Moscow State University), exerted a great influence on the contents of chapters 3 and 4 and in particular on paragraph 4-1. These chapters lay down only the foundations of the method and should not be considered as a complete course of the theory and practice of programming.

Chapters 5 to 8 describe logical schemes of the programs for a series of practical problems through the example of which the reader can gain an acquaintance with the methods of solving engineering problems on automatic digital computers. Although the majority of the questions considered here is of the nature of general engineering problems, such as the study of transient processes, calculation of the stability of dynamic systems and investigation of resonance phenomena in intricate structures, the exposition of these problems is given

a definite electrotechnical trend.

It is quite understandable that the authors were not able, even to a small degree, to reflect the diversity of engineering problems for whose solving the digital computers are presently employed. The authors hope, however, that the reader will be able to prepare himself independently and, after gaining familiarity with the methods of programming described in chapters 3 and 4 and with some examples considered in chapters 5 to 8, to solve on the computer any problem that may arise. At that it is of course assumed that the problem is mathematically formulated and a numerical method for its solution is available.

Chapter 9, which is of a reference nature, offers brief information on approximate methods for solving certain mathematical problems. The basic literature on this problem is cited at the end of the book.

In 1956, the group of employees of the LUMS of the USSR Academy of Sciences, the NII for the electric engineering industry of the State Planning Committee of the USSR, and Academy of Sciences of the Armenian SSR, under the general supervision of the Corresponding Member of the AS USSR, I.S. Bruk, and Member of the AS of the Armenian SSR, A.G. Iosif'yan, constructed the M-3 computer.

The book generalizes the experience gained by the authors who participated in the work of the group of employees of the LUMS AS USSR, the NII EP of the Gosplan USSR and the AS Armenian SSR on the designing of the M-3 computer and on conducting a series of engineering investigations on the M-3 computer in the NII EP Gosplan USSR and on the BESM computer in the ITMiVT AS USSR. Being one of the first publications of this kind, the book cannot be free of drawbacks. The authors will be thankful for all the remarks and suggestions which the readers will make.

In conclusion, the authors use this occasion to express their deep gratefulness to Prof. A.A. Lyapunov and Docent V.M. Kurochkin for a number of valuable indications which the authors took into consideration in preparing the manuscript for publication.

Chapters 2, 6, 7, 8 and paragraphs 1-1, 1-2, 1-4 were written by B.M. Kagan; chapters 3, 4, 5, 9 and paragraphs 1-6, 1-7, 1-8, 1-9 were written by T.M. Ter-Mikaelyan, and paragraphs 1-3 and 1-5 were written by the authors jointly.

The Authors

CONTENTS

STAT

FIRST CHAPTER.

Fundamentals.

1-1. Introduction

The development of technique calls for the increased capacity of individual units and machines, the intensification of technical processes, the increase of speeds, temperatures, pressures, stresses in the structural materials of machines and apparatuses, the raising of reliability, and speedy and precise operation of various devices. The solution of all these problems is impossible without a deep and all-sided study of processes taking place in the machines, apparatuses and complicated circuits.

In many cases mathematical relations describing the processes in the studied devices turn out to be very complicated due to the intricacy of a circuit, the presence of elements with distributed parameters, phenomena of saturation and other non-linearities. As a result the theoretical investigation by the conventional methods becomes practically impossible.

The engineer-investigator who faces such difficulties can often resort to the physical simulation of the phenomenon under study. At the present time, for instance, physical models of electric transmission lines are effectively used for studying processes occurring in these complicated systems. Physical simulation is of a special significance when there is no reliable mathematical description of the phenomenon being studied. However, only certain physical phenomena can be studied with such a model, and any more or less essential modifications of the original parameters may call for the designing of a new model.

Modern computers open new possibilities. They can be divided into two large groups: a) electronic simulating analog computers, and b) electronic high-speed digital computers.

In electronic simulating analog computers, circuits are devised by means of electronic tubes, capacitors, resistors and some other elements, in which the changes in currents and voltages with time are described by the same differential equations as the phenomenon under investigation. The solution of equations is obtained in the form of oscillograms of corresponding voltages in the circuits of the simulating device. These voltages-analogs of the variable

quantities of the problem being solved vary continuously with time, if this corresponds to the initial equations. Electronic simulating devices are also called computers of continuous action. Their accuracy amounts to 5 to 10%. Electronic simulating computers are not universal devices. They are convenient for solving such problems which in mathematical respect are reduced to ordinary differential equations with constant and variable coefficients(1). Electronic analog computers have been mainly applied for investigating automatic control systems. This is because it is comparatively simple to connect an analog device with the operating equipment.

During the last years automatic high-speed digital computers (ATsVM) came into wide use. These computers operate with numerical quantities presented in digital form. The quantities cannot vary continuously in a digital computer. They vary discontinuously, assuming individual elementary arithmetical operations (addition, subtraction, multiplication and division of numbers). The control of the calculating process is carried out automatically according to a program worked out in advance. Main advantages of digital computers consist in their universality and accuracy of operation.

High-speed digital computers make it possible to solve a very wide range of problems. It is necessary only that the problem should admit of a numerical method of solution. The accuracy of these computers is high, because the calculations are usually performed up to 9th or 10th digits. The calculating accuracy with digital computers is not limited by anything and depends only on the number of orders in the numbers with which the computer operates.

Digital computers operate at an enormous speed, performing thousands and tens of thousands of operations per second. For example, the BESM computer designed under the guidance of Academician S.A. Lebedev performs 8,000 to 10,000 operations per second.

High-speed digital computers can perform not only arithmetical but also some logical operations. It is possible therefore to automate the process of calculations and to carry out the automatic selection (depending on the satisfaction of certain conditions) of one of several variants in the course of calculations. Although the calculations with these computers call for a preliminary rather labor-consuming work on the compiling of the program of calculations (so-called "programming of a problem") the labor applied is wholly paid off, if a complicated problem is being performed.

---

(1) There exist also analog computers for solving equations with partial derivatives.

Studies of losses in the grids, current distribution, and calculations of economic load distribution, are also performed with digital computers.

The use of computers for determination of economically optimum regimes of power system operation has a bright prospect. These calculations will determine the graphs of economically reasonable load distribution between individual thermal and hydropower plants of a power system, which take into account the graphs of the total load of the system, losses in the lines, fuel cost, water levels in reservoirs, natural discharges, and efficiency factors of individual units. The possibility of determination of economically reasonable load distribution with the ATsVM makes it possible to automate the dispatcher control of power systems.

The ATsVM's are used, in the field of power engineering equipment, for designing atomic reactors, for calculating the thermal balance of turbines, for selecting the optimum regime of power units operation, and for studying the strength and vibration of units.

It is known, how important it is for high-capacity generators to determine the critical rates of rotor rotation, and to establish the relation between the values of critical speeds and the structural characteristics of the rotor and bearings.

It is practically impossible to carry out any complete investigation of this problem by conventional methods in view of the large amount of labor consumption for calculations. Computers, however, make it possible to solve this problem. In a similar way, computers can be applied for calculating resonance frequencies of turbogenerator panels, operating regimes of hydrogenerator footstep bearings, for investigating temperature distribution fields, air velocities in channels, and other problems of heating and ventilation of large electrical machines.

Digital computers find application in calculations and investigations of stability and various operating regimes in complicated systems of automatic control and regulation, in investigations of dynamic processes in complicated systems of electric drive, in control and regulation circuits containing various non-linear devices, such as saturation choking coils, mercury rectifiers, etc. They can be used for calculating optimum processes in automatic control systems under certain limitations (limitation of speed, moment, etc.), for synthesis of automatic control systems with prescribed characteristics.

Of great interest is the calculation study with the ATsVM of

The performance of certain logical operations by the automatic digital computers opens new possibilities in designing the systems of automatic control and regulation. These possibilities are brought about by means of switching a computer into the system of automatic control.

High-speed computers are a powerful means for investigating and calculating complicated engineering problems in various fields of engineering. They make it possible to mathematically (numerically) simulate operating processes in devices of various kinds.

While new technique is being designed, many efforts and means are spent for constructing and investigating various physical models and the manufacture of intermediate experimental specimens. However, if differential equations describing the processes in the device being constructed are known, then it is possible, with an "ATsVM", to calculate in a short time working processes for a large number of designing variants and to choose the best one. With the aid of an ATsVM, it is possible to determine the optimum processes in operation of a complicated machinery by means of calculations. Such a way permits to reduce, in many cases, the amount of experimental studies, physical modelling and testing intermediate experimental specimens.

In the investigation of working processes it is very important to separate factors essentially affecting the process from secondary ones which may be omitted. Digital computers can be used for calculating working processes under various assumptions. The comparison of results obtained permits to determine reasonable limits for simplifying assumptions. In a number of cases the detailed investigation of the problem can then be switched to simpler devices, such as electronic computers of continuous action, calculating desks, etc.

Calculations and studies on ATsVM will be widely applied in designing electrical engineering equipment within the near future. In connection with the construction and future operation of very large electric power plants which will supply long-distance transmission lines with electric power, calculations of static and dynamic stability of systems, and the study of the effects of generator characteristics and excitation circuits of synchronous generators on their operation, will be of great importance. Solution of these problems, connected with considerable calculating difficulties, is comparatively simple for computers. In addition to stability problems, digital computers are used presently for the analysis of a number of other complicated problems in the operation of power systems.

dynamics of automatic control systems subjected to the action of random signals varying continuously. It is connected with the possibility of using the ATsVM for working out random quantities with various laws of distribution.

The progress achieved in the field of designing computers put forward the problem of using the ATsVM as an element of automatic control systems ("controlling computers") and of designing on this basis the complex automation system of technological processes, as one of the important problems at the recent stage of technical development.

In this connection, the application of universal ATsVM's has a great practical importance for engineering and scientific calculations, for designing calculations, or, in other words, for numerical simulation of controlling processes in the automation systems with controlling computers. The universal ATsVM's can, if equipped with special continuous-discrete converters and their reverse, in a number of cases be connected with actual machines and be used in experimental studies as models of controlling computers.

It is expedient to apply computers for calculations of the series of electric machines and transformers. It turns out to be possible to find optimum size and winding characteristics of electric machines and transformers starting from the given nominal data (capacity, efficiency factor, etc.) and certain criteria (minimum weight, minimum cost, etc.)

We have listed only a few engineering problems from the field of electrical engineering and some adjacent fields for the solution of which digital computers are applicable.

We shall cite also several examples of ATsVM application for engineering calculations in other branches of engineering. In the field of aviation engineering the ATsVM's are used for calculating the strength, for studying the problems of vibration and flutter in aircraft construction, for determining the best shape of aircraft wings and reactive engine nozzles, for investigating the problems connected with take-off, landing, and catapulting of aircraft, for determining the take-off speed and trajectory, and for solving other problems.

Ballistic tables are calculated with the aid of ATsVM.

In the construction field the ATsVM's may be applied for calculating complicated trusses, bridges, buildings, dams, etc. As an example, the shapes of the outline of the steepest canal slopes which do not slip down have been determined by BESM computer.

With the aid of ATsVM's it is possible to calculate pressures in intricate hydraulic systems and in gas pipelines.

In the oil industry the ATsVM's are applied for solving such problems as outlining the oil strata, determination of the most effective oil refinery processes depending on the properties of initial raw materials, etc.

Of a bright prospect is the application of the ATsVM's for economic calculations and studies in the planning of production, analysis of the productional process running, determination of net costs, prices, wages, etc.

The possibility of applying universal ATsVM's for solving engineering problems does not reduce to any degree the importance of using other computing devices for certain calculations, such as analogs of electric systems, electronic computers of continuous action. For example, calculations of current distribution in intricate power systems are more fitted for models of electric grids. Computers of continuous action, analogs, are preferable for many problems in the field of automatic control and regulation, as they need no programs for a problem solution and admit the direct connection of a computer with the operating equipment. At the present time the calculation methods are being developed which make use of various combinations of ATsVM's and models of electric grids or ATsVM's and analog computers.

1-2. On numerical methods of solving mathematical problems.

The investigation of scientific and technical problems is usually reduced in mathematical respect to establishing and analysing solutions of differential equations. Although very many processes in engineering can be sufficiently precisely described by differential equations, however, the solution of these equations in an analytical way (in a "closed form") can be obtained only in very rare cases. There exist numerical (approximate) methods for solving mathematical problems which do not call for analytical solutions. In the using of these methods the solution of complicated mathematical problems is reduced to same sequence of arithmetical operations performed in a definite order. Numerical methods of solving mathematical problems are described in Chapter 9. Here we shall confine ourselves to one simple example.

Let us consider the non-linear differential equation(1) which is

(1)Equation (1-1) describes the motion of the rotor of a synchronous machine. is the angle between the vectors of e.m.f. of the idle run of a generator and the voltage of a receiving system.

well known to engineers and electricians:

$$M_o \frac{d^2\theta}{dt^2} = P_o - P_m \sin\theta \qquad (1\text{-}1)$$

where M_o, P_m and P_o are constants.

The solution of equation (1-1) cannot be presented in finite terms by elementary functions. To solve this equation one can use numerical methods, as e.g. the Euler method of finite increments ("method of successive intervals").

Let relative angular speed and acceleration be denoted as follows:

$$\Omega = \frac{d\theta}{dt}; \quad \alpha = \frac{d^2\theta}{dt^2}$$

and the following initial conditions are given:

$$t = 0 \quad (\theta)_o = \theta_o; \quad \left(\frac{d\theta}{dt}\right) = (\Omega)_o = 0$$

It is requested to solve equation (1-1), i.e. to determine $\theta = f(t)$ at $0 \leq t \leq T$.

The prescribed range of t-values is divided into sufficiently small time intervals, steps Δt, and it is assumed that acceleration is constant in each interval and equals its value at the beginning of the interval. Indices i and i+1 denote the values of the quantities at the beginning and at the end of the i+1 interval respectively. The magnitude of acceleration which corresponds to the beginning of the i+1 interval is determined by the expression

$$\alpha_i = \frac{P_o - P_m \sin\theta_i}{M_o} \qquad (1\text{-}2)$$

Then the increments of speed $\Delta\Omega_{i+1}$ and of angle $\Delta\theta_{i+1}$ during the i+1 interval will be as follows:

$$\left.\begin{aligned} \Delta\Omega_{i+1} &= \alpha_i \Delta t \\ \Delta\theta_{i+1} &= \Omega_i \Delta t + \alpha_i \frac{\Delta t^2}{2} \end{aligned}\right\} \qquad (1\text{-}3)$$

At the end of the i+1 interval the relative speed and angle will take the following values:

$$\begin{aligned} \Omega_{i+1} &= \Omega_i + \Delta\Omega_{i+1} = \Omega_i + \alpha_i \Delta t \\ \theta_{i+1} &= \theta_i + \Delta\theta_{i+1} = \theta + \Omega_i \Delta t + \alpha_i \frac{\Delta t^2}{2} \end{aligned} \qquad (1\text{-}4)$$

Finally we have one more relation:

$$t_{i+1} = t_1 + \Delta t \qquad (1\text{-}5)$$

STAT

The integration of equation (1-1) is carried out in the following way in the hand calculations. Since the values of θ_i and Ω_i in the beginning of the i+1 interval are known (for the first interval $\theta_i = \theta_0$, $\Omega_i = 0$), one determines $\sin \theta_i$ from the tables and calculate acceleration a_i by formula (1-2). Then one determines Ω_{i+1}, θ_{i+1}, t_{i+1} by formulae (1-3) to (1-5) proceeding according to a definite succession. After finishing calculations for one interval one passes over to the next one. When the acceleration value in the beginning of the next is being determined, the value of θ_{i+1} obtained for the preceding interval is substituted into (1-2), and so on. Before going over to calculations for the next interval, one has to compare the t_{i+1} value with the prescribed integration limit T, and if $t_{i+1} < T$, then the next step of integration must be carried out; otherwise calculations are stopped because the estimation has already been completed.

Thus the process of integrating equation (1-1) can be reduced to carrying out a definite sequence of arithmetical operations and one simple logical one, the comparison of the values of t_{i+1} and T. Depending on the results of this comparison the calculations either continue or discontinue.

The sequence of arithmetical and logical operations which have to be performed over the initial data and over the results of the intermediate calculations in order to obtain the answer, is called the algorithm of the solution of a mathematical problem (L. 12).

1-3. Block-diagram of the ATsVM computer.

In solving the most of mathematical problems by numerical methods, enormous number of arithmetical operations must be carried out. Still recently the performance of these calculations called for such great amount of labor that the solution of many problems was practically unrealizable. During the last one-and-half decades have been designed electronic automatic digital computers (ATsVM's) operating at an enormous speed. The process of calculations in the ATsVM is wholly automated by means of a programming control.

Fig. 1-1 shows the simplified block-diagram of the ATsVM. The computer consists of the following basic units: an arithmetical unit, a memory device and an output device, and a controlling unit.

The arithmetical unit, AU, performs operations over the numbers introduced into it. The working speed of the arithmetical units in modern computers amounts to thousands of arithmetical operations per second. One can say that the arithmetical unit is like a comptometer

STAT

operating at a gigantic speed.

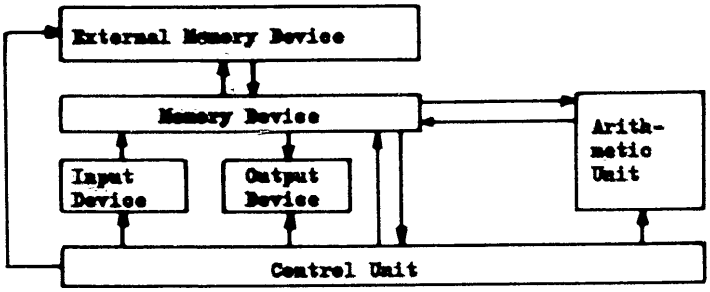


Figure 1-1.

In solving the problems with a comptometer or a desk computer the operator picks up the numbers wanted, switches in the device and then writes down on the paper the result obtained. The application of a similar method for electronic computers would make completely senseless the high speed of their arithmetical unit. The human being cannot insert numbers into an arithmetical unit at a required speed and read out results of the operations. Therefore these processes are automated by means of a so-called "memorizing device", ZU, or, briefly expressed, "memory".

The memory consists of a series of separate cells in each of which one or several numbers are stored. Cells have numbers which permit to distinguish them from one another. These numbers are called "adresses" of the ZU cells.

The numbers are transferred from the memory into the arithmetical unit of the computer ("reading of the number"), and the result obtained in the arithmetical unit is put back into the memory ("recording of the number"). The time necessary for the transfer of a number from the ZU into AU or back is called the time of addressing to the memory. This time should be commeasurable with the operational speed of the arithmetical unit for the effective utilization of the potentialities of the latter.

Circuits of the memories are devised in such a way that the contents of a cell should not change after reading out a number from it. If this number will be needed in a next calculation, it can be obtained from the same cell. However, when a new number is transferred into a memory cell, the number previously stored there is erased and is replaced by the new number.

In order to completely automate the calculating process and fully exclude the participation of a human being in the computer operation, the ATsVM's are provided with a control unit, UU, which controls the

STAT

cannot be brought about on the computer directly in this shape. It is necessary for this purpose to code it in such a shape that the computer could "read" it and fulfill.

In the following paragraphs we shall consider those operations which an automatic digital computer must carry out in order to make possible the compiling of the program for any calculating process, and shall become acquainted with the methods of coding and storing the numbers and commands in the computer.

The program and the numbers are put into the computer memory by means of the "input device". At last, the computer is provided with the "output device V" for typing the results obtained.

In modern computers the high speed of calculations (thousands operations per second) is achieved by means of constructing the memory of two units (Fig. 1-1): a) the inner high-speed memory ZU for a comparatively small number of cells (usually from 1,000 to 4,000); b) the external memory VZU which operates comparatively slow but is able to store a great quantity of numbers (several tens or even hundreds of thousands).

In this case, all data necessary for the solution of a given problem is inserted into the external memory. In the course of calculations, the parts of the program and the constants which correspond to individual steps of the problem solving are re-written into the internal memory. Thus the calculating process proper proceeds without addressing the external memory.

The next chapter will deal with the operation of elements and units of the automatic digital computer.

Although every computer possesses a series of specific (often very essential) peculiarities, the basic properties of all digital computers with the program control are nevertheless mainly the same. Therefore all considerations in the present chapter will be presented for a certain conditional computer.

1-4. Systems of counting.

In everyday life the decimal system of counting is used in which ten signs (digits) are employed for presentation of numbers: 0; 1; 2; 3; 4; 5; 6; 7; 8; 9, and any of these digits can be in every place. In other words, in the decimal system every number is represented by the sum of exponents of number 10, and coefficients with them are equal to the number of unities in the corresponding places of the decimal number. For example, the number $37406.15 = 3 \cdot 10^4 + 7 \cdot 10^3 + 4 \cdot 10^2 + 0 \cdot 10^1 + 6 \cdot 10^0 + 1 \cdot 10^{-1} + 5 \cdot 10^{-2}$.

STAT

whole calculating process and, in particular, transfers numbers from ZU into AU, switches in AU for carrying out a required operation and puts the result obtained into ZU.

As was noted in the preceding paragraph, every numerical method reduces the solution of a mathematical problem to a series of successive arithmetical and logical operations over both the numbers given in the problem conditions and those obtained in the process of calculations. As the controlling unit of the digital computer controls the whole process of calculations itself, an exact description must be carried out. Such a description of the whole calculating process is called the program of the solution of the given problem on the automatic digital computer. The automatic programming control is a basic property of the high-speed digital computers.

The program consists of separate "commands" (one says also "orders" or "instructions") which indicate which individual operation and over which numbers must be carried out by the computer at the given stage of calculations. These commands incorporate succesively all the operations which must be performed on the computer for solving the given problem by the selected numerical method. The totality of the commands necessary for the solution of a problem, written in a certain succession forms the program.

This can be illustrated by an elementary example. Assume, for instance, that it is necessary to compute a determinant of the second order

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

To do this, one has to multiply number a by d, then multiply number b by c and subtract the second product from the first. In other words, the following operations are carried out:

1	x	a	d	ad
2	x	b	c	bc
3	−	ad	bc	ad − bc

Here the first column contains the symbol of an operation which has to be carried out, the second and third contain the numbers over which these operations are performed, and finally the fourth column contains the results of the operations.

Thus the program for calculating the determinant of the second order consists of three separate commands. However, this program

In high-speed digital computers the binary system of counting is often used for presentation of numbers and commands. In this system only digits 0 or 1 can occur in each order of the binary number. In the binary system every number is represented by the sum of the exponents of number 2, and coefficients at the exponents of the number 2 may be either 0 or 1. For example, the decimal number 21.5 = $1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1}$ and it is written as follows 10101.1 in the binary system. Table 1-1 shows the first 17 decimal and binary numbers.

Table 1-1

Decimal numbers	0	1	2	3	4	5	6	7	8	9	10	11
Binary numbers	0	0	10	11	100	101	110	111	1000	1001	1010	1011

Decimal numbers	12	13	14	15	16	17
Binary numbers	1100	1101	1110	1111	10000	10001

A greater number of orders is needed for representation of binary numbers than for the same numbers in the decimal system. Nevertheless the application of the binary system makes it possible to reduce the total amount of equipment and to provide more convenience in designing digital computers, because any element having only two stable states can be employed for representation in the computer of the order of a binary number. Examples of such elements are relays, trigger circuits, etc.

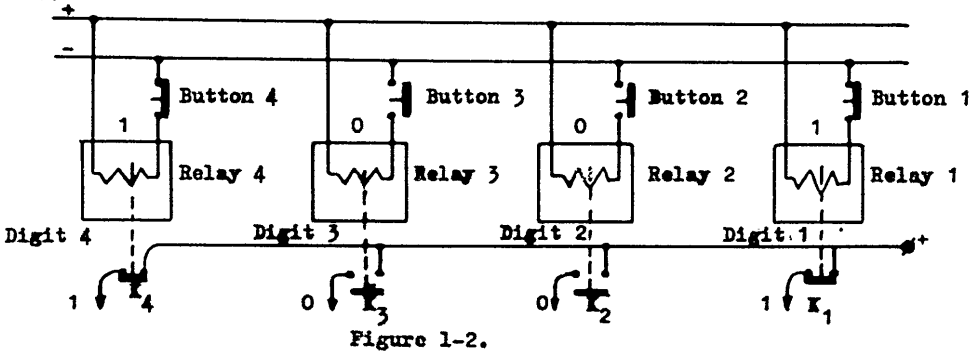


Figure 1-2.

It can be assumed that the closed state of a relay represents unity and the disconnected state represents zero. In the circuit of the relay-contact the presence of voltage can represent unity and its absence - zero. For example, having four relays (Fig. 1-2) and

STAT

assuming that the state of relay #1 represents a digit of the first binary order, relay #2 - of the second binary order, and so on, any integer from 0 to 15 can be represented in the binary form by means of buttons KN. Figure 1-2 shows relays #1 and #4 switched in, which corresponds to the binary number 1001 (decimal number 9).

The addition of two numbers in the binary system, as well as in the decimal system, can be performed by columns. At that the following rules are observed in the addition process in each order: 0 + 0 = 0; 0 + 1 = 1 + 0 = 1; 1 + 1 = 0 + unity of transfer into a higher order. For instance, the operation of addition of two numbers 23 + 25 = 48 when represented in the binary system is carried out in the following way:

$$\begin{array}{r} 10111 \\ +\ 11001 \\ \hline 110000 \end{array}$$

An important advantage of the binary system consists in the extreme simplicity of its multiplication table:

$$\begin{array}{l} 0 \times 0 = 0 \\ 0 \times 1 = 0 \\ 1 \times 0 = 0 \\ 1 \times 1 = 1 \end{array}$$

As an example, the operation of multiplication 6 x 5 = 30 looks as follows in the binary system of recording:

$$\begin{array}{r} \times\ 110 \\ \quad 101 \\ \hline \quad 110 \\ \quad 000 \\ \quad 110 \\ \hline 11110 \end{array}$$

Thus the operation of multiplication is reduced to the operations of shift and addition. At that partial products are obtained by shifting the multiplicand to the left by the number of orders which corresponds to the number of non-zero orders of the multiplier.

Besides the binary and decimal systems, are employed also the octal and hexadecimal systems of counting whose bases are the numbers 8 and 16 respectively. In the octal system any digit from 0 to 7 can occur in every order. The decimal number 3011 looks as follows in the octal recording: $5703 = 5 \cdot 8^3 + 7 \cdot 8^2 + 0 \cdot 8^1 + 3 \cdot 8^0$.

In the hexadecimal system 15 digits are employed for representation of numbers, and new symbols are introduced to denote figures larger than 9. For example, ten can be denoted by 0; eleven by $\overline{1}$;

STAT

STAT

twelve by $\overline{2}$; thirteen by $\overline{3}$; fourteen by $\overline{4}$; and fifteen by $\overline{5}$. The decimal number 3011 will be recorded in the hexadecimal system in the following way: $\overline{123} = \overline{1} \cdot 16^2 + \overline{2} \cdot 16^1 + 3 \cdot 16^0$.

The base of any system of counting recorded in the same system is represented by 10 (the number two in the binary system is 10; the number eight in the octal system is 10 and so on).

Octal and hexadecimal numbers are easily converted into binary and vice versa, binary numbers are simply converted into octal or hexadecimal. It is explained by that, that the bases of the octal and hexadecimal systems are integer exponents of the number 2: $8 = 2^3$; $16 = 2^4$. In order to convert an octal number into the binary form it is sufficient to replace each digit of the octal number by the corresponding three-order binary number. In the same way for the conversion from the hexadecimal to the binary system, each digit of the hexadecimal number is replaced by the four-order binary number. For instance, the octal number 5703 looks as follows in the binary system:

$$\underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{000}_{0}\ \underbrace{011}_{3}$$

and the hexadecimal number $\overline{123}$ in the binary system is written in the following way:

$$\underbrace{1011}_{1}\ \underbrace{1100}_{2}\ \underbrace{0011}_{3}$$

In the conversion from the binary to the octal (or hexadecimal) system, the groups by three (or four) binary orders are replaced by the corresponding digits of the octal (or hexadecimal) number, beginning consecutively from the lower orders for the integral part of the number and from the higher orders for the fractional part.

The considered numbers 101 111 000 011 and 1011 1100 0011 can be looked upon as octal and hexadecimal numbers respectively in which a digit of each order is recorded in the binary system. These forms of number recording are called binary-octal and binary-hexadecimal systems. They are called also binary-coded systems.

The binary-decimal system or number presentation is also employed in computers. In this system each digit of the decimal order is written in the form of the corresponding four-order binary number, as e.g. the number 952 in this system looks as follows: 1001 0101 0010.

At the present time the binary system is the basic system of counting used in most computers. The binary system is employed in

STAT

STAT

computers for representing and storing numbers and commands and for performing arithmetic operations. The octal and hexadecimal systems are employed for compiling the programs of calculations for the shorter and more convenient recording of binary numbers, since these systems need no special operations for being converted to the binary system.

The constant quantities ("constants") - initial conditions, coefficients and so on, necessary for the solution of a problem, are inserted into the computer in the octal (hexadecimal) or binary-decimal systems. In the latter case the conversion of the binary-decimal numbers into binary ones is performed by the computer according to a special program. The conversion from the decimal into binary-decimal system is performed outside of the computer on a tape punching device (perforators, etc.).

The results of calculations are obtained from the computer in the octal (hexadecimal) or decimal systems, and as intermediate systems used within the data output unit, the binary coded varieties of these systems are employed. The conversion of data from the binary system into the binary-decimal one is performed by the computer according to a special program.

1-5. Computers with floating and fixed decimal point.

Each digit of a binary number is presented on an automatic digital computer by some technical device, for example, a relay. The computer contains only a limited number of such devices and therefore it will operate only with numbers of a limited length, i.e. with numbers containing a certain amount of digits. The number of digits is selected once by the accuracy requirements for solving a problem and secondly by technical considerations.

Principally, a digital computer may provide any desired accuracy by a corresponding increase of the amount of digits used for representing the number. However, a too great increase of the digit number leads to an increase of the volume of the equipment and to great constructional difficulties. Usually 30-40 binary digits are used in digital computers for representing numbers.

The range of numbers which may be represented in a computer by a given amount of digits depends essentially on the accepted mode of number representation. Two types of number representations are used in computers: a) with a fixed decimal point; b) with a floating decimal point.

In computers with a fixed decimal point, which separates the full

STAT

number from the fraction, the decimal point is fixed between certain digits and is kept unchanged during all computing operations. Usually the decimal point is located before the left hand digit, i.e. before the top digit. In this case, the computer will accept and process only numbers with a modulus smaller than one.

For presenting the sign of a number a special digit is used, placed, for example, left of the decimal point. Thereby the sign "plus" is presented in the sign digit by a zero, while a one is used for the "minus".

If, for example, the number of binary digits of a computer is equal to 34, whereby one digit is used for recording the sign, then the digit lattice will have the form shown by Figure 1-3. In this case the computer will accept numbers ranging from $-(1 - 2^{-33})$ to -2^{-33}, from $+2^{-33}$ to $+(1 - 2^{-33})$ and the number zero. A number with a modulus smaller than 2^{-33}, but greater than zero may not be presented in the computer and is shown as a zero (the number moves from the digit lattice of the computer to the right). A number having a modulus greater than $(1 - 2^{-33})$ is also not presented by the computer since the number moves out of the digit lattice to the left (the so-called overflow of digits).

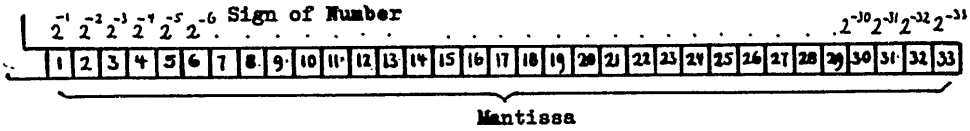


Figure 1-3.

When such numbers appear during a computation process, then its top digits (digits left of the decimal point) are lost and the result is false. This is avoided in automatic devices in which an automatic device interrupts the work of the computer in case a number greater or equal to one appears.

When programming problems for computers with a fixed decimal point, it is necessary to introduce special scale factors, whereby all basic, intermediate or final magnitudes become smaller than one.

The error of the computed result for adding and subtracting operations depends on the absolute accuracy, while multiplying and dividing operations depend on the relative accuracy of number representation of a computer. The absolute accuracy of number representation is determined by the number of digits used. With a fixed decimal point, the relative accuracy is the lower, the smaller the number. Therefore, the scale factor must not be too large, since it

necessary to shift the mantissa towards left by three digits and to reduce the order of the number by three units. As a result, the same number is obtained, but only in a normalized form $2^{11} \times 0.1011000$.

When multiplying numbers in computers with a fixed decimal point, the order of the number is added while the mantissa is multiplied. Analogously, the divisor is subtracted from the order of the dividend, the mantissa of the dividend is divided by the mantissa of the divisor and the result is normalized (partially). When adding and subtracting numbers on computers with a floating decimal point, the orders of the numbers are preliminarily equalized. The order of the smallest number is made equal to the order of the largest number of digits, equal to the difference of the orders of the numbers, after which the mantissa is added (or subtracted). The order of the sum (or the difference) obtained equals to the order of the larger number. All these operations are performed automatically in the arithmetic block of the computer.

The memory device of the computer will store normalized and not normalized numbers. As we will see in the following chapter, the commands belong to the latter.

1-6. Coding of commands.

As it was said before, the digital computers are designed to perform arithmetic operations with numbers. Thereby, it is necessary that these numbers are stored in the computer itself for an effective use of the speed with which the arithmetic block performs those operations. Consequently, the memory device of the computer must not only store the basic initial numbers of the problem but also the results of intermediate computations used during the course of the problem solution. For example, in case a differential equation has to be solved,

$$y' = f(x, y);$$
$$x_o \leq x \leq X;\ y(x_o) = y_o$$

and Euler's formula is used,

$$y_{i+1} = y_i + hf(x_i, y_i),$$

then the computer must store the numbers x_o, y_o, X, step h, and in addition the values x_i and y_i, obtained during the preceding step of integrating.

With the example, listed in paragraph 1-3, it was necessary to store in the computer not only the elements of the determinant a,b,c

The smallest, by modulus, normalized number, which is representable on our computer, will be $2^{-31} \cdot \frac{1}{2} = 2^{-32}$. Numbers having a modulus smaller than 2^{-32}, cannot be represented and will be shown as being equal to zero. The largest, by modulus, normalized number will be $2^{+31}(1 - 2^{-35}) \approx 2^{31}$. Consequently, only numbers within the range of $-2^{+31}(1 - 2^{-35})$ to -2^{-32}, of $+2^{-32}$ to $+2^{+31}(1 - 2^{-35})$ and the number zero may be represented on the type of computer under consideration. Since 2^{31} and 2^{32} are approximately equal to 10^9, the operational range of the computer will be the interval from -10^9 to -10^9 and from $+10^9$ to $+10^9$, or in other words, numbers are represented with an accuracy of up to 9 decimal places. Such a range seems to be adequate for the majority of occurring practical problems.

The range of numbers with which a computer with a floating decimal point will operate is determined by the number of digits placed under the order category, while the accuracy depends on the number of digits under the mantissa category. Actually, with the same number of mantissa digits, the accuracy of a computer with a floating decimal point is higher than the one of a computer with a fixed decimal point. This is explained by the fact that the relative accuracy of number representation is decreased with reduced numbers in case a fixed decimal point is used, while it is kept constant for all numbers within the operating range of a computer with a floating decimal point because of the normalization.

In this way, the number representation system with a floating decimal point permits to obtain a greater working range, a higher accuracy of computations (at an equal number of mantissa digits) and simplifies programming operations compared to the fixed decimal point system. However, the floating decimal point system requires additional computer equipment for representing and processing of orders. Therefore the floating decimal point system is applied in larger computers, while the fixed decimal point system is used in smaller computers.

If not normalized numbers are formed during a computing process, then the computer (with a floating decimal point) will normalize them automatically(1). In case the top k digits of the mantissa are equal to zero, then the normalization operation consists of shifting the mantissa towards left by k digits (in order to fulfill requirement 1-7) with a corresponding reduction of the order of the number by k units. For example, for normalizing the binary number $2^{14} \times 0.0001011$, it is

(1) In case the normalization operation is not required, then the computer is stopped.

reduces the accuracy of the computations.

In computers with a floating decimal point, all numbers are represented in the following way:

$$X = 2^p \cdot A, \quad |A| < 1 \qquad (1\text{-}6)$$

whereby p - is a full number (positive, negative or zero), designated by the order of the number X;

A - is the mantissa of the number X.

The magnitude of the order is determined by the position of the decimal point in the number. Provided the number A satisfies the inequation

$$|A| > \tfrac{1}{2} \qquad (1\text{-}7)$$

which says that X is a "normalized number". The afore-mentioned inequation means that with a normalized number, the first digit after the decimal point is always a one. In computers with a floating decimal point, the order - and the mantissa A are presented separately. Thereby, the digital lattice of the computer is divided into two parts: one part of the digits is used for presenting the order p and the rest for presenting the mantissa A. It is necessary to take into consideration that with normalized numbers not only mantissa A but also the order p might have both signs. For example, the number $\frac{1}{8}$ is represented in the normalized form as $2^{-2}\ \frac{1}{2}$, etc. Consequently it is necessary to reserve one digit for representing the sign of the order and the sign of the mantissa.

In the following, some conventional computer was considered which has a floating decimal point and a digital lattice of 42 digits. If it is assumed that the digital lattice computer is of the type as shown by Figure 1-4, then the left six digits are reserved for presenting the order (one of them for its sign) and the other 36 for the mantissa (one of them for its sign).

With five digits one may write numbers from 0 to $2^5 - 1 = 31$, whereby the order p will comprise full values from -31 to +31. In the normalized form, the mantissa may assume the value of $\frac{35}{0.100....0}$ to $\frac{35}{0.111....1}$, i.e. of 2^{-1} to $2^{-1} + 2^{-2} + \ldots + 2^{-35} = 1 - 2^{-35}$.

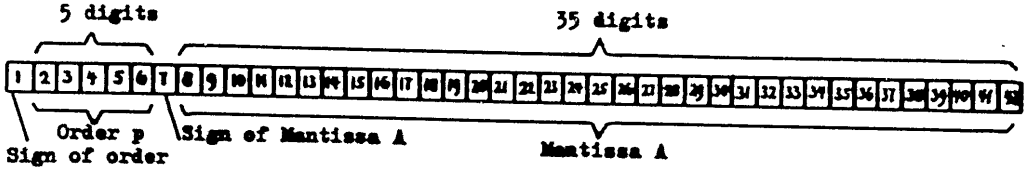


Figure 1-4.

and d, but also the products ad and bc for having the possibility of subtracting the second product from the first one. Therefore, each command must contain information concerning the kind of operation to be performed and on which numbers and where the answer must be placed. the program for solving the problem, i.e. the sequence of commands which describe completely the entire computation process, must also be stored in the computer. Consequently, commands must be added in such a form that they are easily placed into the computer.

For example, it will be necessary to subtract number b from number a. Numbers a and b are stored in certain cells of the memory, which may be called α and β. Further, it will be necessary to place the difference a - b into some cell of the memory, for example into cell γ. It is necessary to emphasize that α, β, γ are the numbers of memory device cells, i.e. the numbers by means of which all memory cells are numbered.

For the operation under consideration a corresponding command will be used containing the following data: "subtract from the number stored in cell α, the number stored in cell β and place the result into cell γ". Schematically, this command may be written in the following form:

—	α	β	γ
subtract	a	b	a - b

Here, in the first box, there is the symbol for the operation which the computer has to perform and the two following boxes contain the numbering of the cells in which the numbers are stored, and, finally, in the last box, there is the number of that cell into which the result is to be placed. In case it were necessary to divide number a by number b, then the schematic representation of such a command would appear as follows:

:	α	β	γ
divide	a	b	a : b

We already pointed out that α, β, γ are ordinary numbers. Assuming, for example that α designates the third cell, the binary number of which is 11, β is the fifth cell and its binary number is 101, and finally γ is the tenth cell whose binary number is 1010. Then the command will appear in the following form:

:	11	101	1010

We have still to devise a code for this operation which replaces the symbol ":" and which is convenient to be stored in a computer.

A binary number, for example,10, may serve as such a code. Now, the command being considered will have the following form:

STAT

10	11	101	1010

(1-8)

and is represented in the form of the following arrangement of zeros and ones:

10111011010.

In this way, when we use some numeral "code of operations" instead of the conventional symbols +, -, x, : (of course for each operation a different one), then all comands appear as ordinary binary numbers.

If α, β and γ were the designations for the seventeenth, thirty second and thirteenth cell, i.e. if their binary numbers were 10001, 100000 and 1101, then the preceding command in numerical recording would appear in the following way:

10100011000001101. (1-9)

The number of digits in the numerical recording of the command was changed with the change of the cell numbers. This circumstance leads to considerable inconveniencies in deciphering such a recording. For example, in the first case, the cell represents the third and fourth digits, but in the second case it stands for the fourth thru eighth digits. Consequently, it is necessary to indicate anew for each command what the digits of its numerical recording actually represent. Therefore, an equal number of digits is selected for representing commands, in such a way that all commands have a constant length. Thereby, it is necessary to be guided by two considerations.

First, as mentioned above, the control device steers the entire calculating process automatically and therefore the program for the solution must be stored in the computer. Since each command is already given in the form of a binary number, it is only necessary to provide the control device with a special memory for storing the program of the computer. The cells of this memory device could store all commands of the program. However, constructionally it is more advantageous to place the program into the same memory device which is already in the computer for storing numbers. Thus we must take into consideration the length of the memory cells of our conventional computer, which is equal to 42 digits, when composing a numerical command code.

Second, any cells of the memory may be taken as α, β and γ, and

consequently for each of these cells a number of digits must be taken which is adequate to represent the highest number of the cell. For example, if our basic computer contains thousand cells, then not less then ten digits must be used for representing each of the numbers α, β and γ .

Actually, this number of binary digits is required for recording all numbers from 0 to 1000 while α , β and γ can, of course, accept all these values. Exactly in the same way we must take a number of digits for the operation code, which permit to represent an adequate number of different operations to be performed by the computer.

Based on these considerations, each command of the program will be stored in a separate cell and consequently, each command will be represented by 42 digits. Thereby, those six digits are taken for the operation code which serve for representing the order of the number (the convenience of such a selection becomes evident in Chapter 3). The 36 digits remaining in the cell are separated into three groups of 12 digits each, representing corresponding numbers of cells α , β and γ . These three groups are called correspondingly the first, second and third address of the command.

In this way, we may say that the command consists of a code of operations to be performed by the computer, addresses of numbers with which these operations are performed, and the address at which the result is recorded. The schematic arrangement of the digits in the command is shown by the following table:

Code of Operation 6 digits	IA 12 digits	IIA 12 digits	IIIA 12 digits

whereby, for example, IA is read as "first address".

In case the number of some cell does not require all 12 digits, then in the given case the free digits are filled with zeros. The binary number obtained in this way is called code of command. For example, the codes of the afore mentioned commands (1-8) and (1-9) will have the follewing numbers:

000010	000000000011	000000000101	000000001010
000010	000000010001	000000100000	000000001101
6 digits	12 digits	12 digits	12 digits

Now the entire program may be represented in the form of a sequence of numbers and may be placed into the memory device of the com-

puter. This is one of the most essential properties of computers with program control. In Paragraph 1-8 it will be shown in which way the control device of the computer "distinguishes" the cells in which the program commands are stored from the cells containing the ordinary numbers.

Five digits are used for representing the modulus of a number order while one digit is used for the sign of the order. Since the presence of a zero or a one corresponds to the different signs in the sign order, the number orders are actually represented by six-digit numbers. With six binary digits it is possible to record $2^6 = 64$ different numbers - 0, 1, 2, ..., 63.

These same six digits are used for representing the operational code. Consequently, the computer will not perform more than 64 different operations (each operation must have its individual code). Modern computers usually perform a smaller number of operations, since the constructional difficulties in the design of the control device will grow with an increase of the number of possible operations. Thus, not any binary number, even if it has the established number of digits (in our example 42), may be considered as the code of some command.

The afore-mentioned examples show that the command codes, which appear as numbers, may be non-normalized. Consequently, the memory will store normalized and non-normalized numbers.

As it was shown above, each address is the number of that cell in which a number is stored required for performing a given command. Consequently, the possibility must be provided for indicating in each address the numbers of any cell of the memory device. In other words, in the computer under consideration, the operational memory may not have more than $2^{12} = 4,096$ cells, because 12 digits are used for representing each address.

Besides the operational memory, computers usually have an external memory (VZU on Figure 1-1) containing tens of thousands of cells. We cannot place the numbers of these cells into the addresses of the aforementioned commands, but the contents of these cells may be transcribed into the operational memory by special operations. In this way an increase in the number of memory cells is provided without increasing the length of the command code.

Computers built in the afore-mentioned manner are called "computers with a three-address control system". There are also single-, two- and four-address computers.

STAT

In the single-address system, each command contains only the operation code and the address. Three commands are required in the general case for representing an arithmetic operation on two numbers with subsequent recording of the result by the memory. For example, the operation of adding numbers a and b, placed accordingly in cells and with transfer of the result to cell τ, is recorded in the following way:

M	α
N	β
R	τ

Here, M, N, R are the code of operation: M is the transmission of the entry of cell α to the register of the arithmetic block; N is the addition of the number located in the register with the number stored in cell β; R is the transfer of the number from the register to cell τ of the memory. The single-address programming system is used, for example, in the computer "Ural" (see Paragraph 2-12).

In the two-address system, each command contains the code of operation and the addresses of the first and second numbers. The result of the operation is recorded in one of the addresses, for example, in the second. Such a system is used, for example, in the computer "M-3" (refer to Paragraph 2-11).

Another version of the two-address system is also used, in which the first address is used as in the single-address computers, while the second address indicates the number of the cell in which the subsequent command is stored.

Finally, in the four-address command system, the first three addresses are used in the same way as in the three-address commands, while the fourth address indicates in which cell the next command is stored. This problem will not be considered here in further detail.

1-7. Some operations to be performed by digital computers.

The four arithmetic operations, addition, subtraction, division and multiplication of numbers are the basic operations performed by digital computers. But by means of these operations the automatic digital computers perform still a number of other operations with which we will get acquainted in this and the subsequent paragraphs.

The operation of logical multiplication is very suitable for programming (it is designated by the symbol $\wedge$) and which is defined in the following. In one digit the logic operation is performed by the same rules applicable also for the ordinary multiplication, i.e.

STAT

$$1 \wedge 1 = 1; \quad 1 \wedge 0 = 0 \wedge 1 = 0 \wedge 0 = 0.$$

The n-digit number $\gamma_n \ldots \gamma_1$, is the result of the logical multiplication of two n-digit binary numbers $\alpha_n \ldots \alpha_1$ and $\beta_n \ldots \beta_1$, in which each digit γ_i is the result of the logical multiplication of the figures α_i and β_i standing in one and the same digit of factors. In this way, the logical multiplication is an operation performed by digits. This means that the result of the operation in each digit does not depend on the value of the other digits of the factors. For example,

$$\begin{array}{r} 1010 \\ \wedge\ 1101 \\ \hline 1000 \end{array}$$

If it is necessary for some reason to separate some k or j digit of binary number $\alpha_n \alpha_{n-1} \ldots \alpha_1$, than it is only necessary to multiply logically this number by the n-valued number, whereby the digits k and j consist of ones while zeros stand in the remaining digits.

$$\begin{array}{ccccccccccccc} \alpha_n & \alpha_{n-1} & \ldots & \alpha_{k+1} & \alpha_k & \alpha_{k-1} & \ldots & \alpha_{j+1} & \alpha_j & \alpha_{j-1} & \ldots & \alpha_1 \\ \wedge\ 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\ \hline 0 & 0 & \ldots & 0 & \alpha_k & 0 & \ldots & 0 & \alpha_j & 0 & \ldots & 0 \end{array}$$

Frequently, it is necessary to use the operations of shifting numbers to the right or left by some number of digits. In case an n-valued number $\ldots_n \ldots \ldots_1$, is shifted to the right by k digits, then zeros will appear in the first k digits and the following number will be obtained

$$\underbrace{00 \ldots 0}_{k} \alpha_n \alpha_{n-1} \ldots \alpha_{k+1}.$$

When shifting the same number by k digits to the left this number is formed

$$\alpha_{n-k} \alpha_{n-k-1} \ldots \alpha_1 \underbrace{00 \ldots 0}_{k}$$

Frequently, the necessity arises to have a number, located in one cell, placed into another cell, for which the operation of transferring a number from one cell into another cell is used. Sometimes it is necessary to transfer only the modulus of this number to another cell; or the number itself is transferred, but with reversed sign. It is possible to transmit a number from any cell of the memory device to the same cell. This, for example, is used for reversing the sign of a number; thereby it is sufficient to perform the transfer of the number with a reversed sign from the cell in which it was originally stored into the same cell.

STAT

For reproducing the numbers from the memory device of the computer the printing command is used, according to which the computer prints by means of the output device the number which is stored in a particular cell of the memory.

Finally, let us consider the operation of taking the integral or the fractional part of some number a being located in the cell α. The integral part of a non-negative number a is called such an integral number [a] (read "ant'ye" a), for which the difference

$$a - [a]$$

is a non-negative number, smaller than one. For example,

$$[3.7] = 3; \quad [6] = 6; \quad [3/4] = 0.$$

The difference a - [a] is called a fractional part of the number a and is designated $\{a\}$. For example,

$$\{3.7\} = 0.7; \quad \{6\} = 0; \quad \{3/4\} = 3/4.$$

Provided the number a is negative, then under integral part is understood such an integral negative number [a] that the difference is

$$a - [a] = \{a\}$$

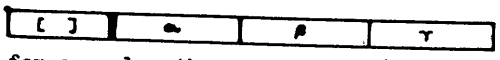
is a positive number smaller than 1. The number $\{a\}$ is called the fractional part of a negative number. For example:

$$[-5] = -5; \quad [-3.7] = -4;$$

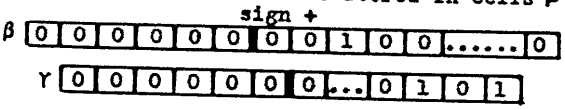
$$\{-3.7\} = 0.3;$$

$$[-0.1] = -1; \quad \{-0.1\} = 0.9.$$

Thereby, the fractional part $\{a\}$ is recorded in some cell as a non-normalized number $2^0 \cdot \{a\}$, i.e., a zero is written into the digits of the order of cell β and the number $\{a\}$ into the digits of the mantissa. The integral part of the number is recorded in the last digits of some other cell γ. In this way the command will have the following form:



If, for example, the number a = $5\frac{1}{4}$ or a = 101.01 in binary recording, is located in cell α then after performing this operation, the following numbers will be stored in cells β and γ



As a conclusion of the subject paragraph it is necessary to review briefly still one more extremely important operation, the meaning of which becomes obvious in Chapter 3. As we have shown in Paragraph 1-5,

STAT

for the addition of two normalized numbers $2^p \cdot A$ and $2^q \cdot B$, the computer will preliminarily equalize their orders and then add their mantissas, i.e. considering that $p > q$, it performs the following operations:

$$2^pA + 2^qB = 2^pA + 2^{q+p-p}\,B = 2^pA + 2^p \frac{B}{2^{p-q}}.$$

In case a non-normalized number is obtained in the result (numbers A and B might have different signs), then the computer will automatically normalize the sum obtained.

The operation of basic addition is also used in computers, whereby the orders and the mantissas of numbers are added separately. This operation is named addition command and is designated by the symbol SK(CK).

In this way,

$$(2^pA) \cdot SK\,(2^qB) = 2^{p+q}\,(A + B).$$

1-8. Control operations.

In automatic digital computers those control systems have found the most wide-spread application where, after having performed the command located in the cell with the number k of the memory device, the computer starts to perform automatically the command recorded in cell k+1 and so forth. The change of normal sequential order of commands to be performed by the computer is achieved by the control operation.

If the program for solving some problem consists of n commands, then they may be placed into n cells of the memory having consecutive numbers, for example, cells numbered c+1, c+2, ..., c+n. Thereby the first command of the program is placed into the cell having the number c+1, the second one into cell c+2 and so on.

The control device(1) has a special address register (counter). The address of a series command is formed in it which must be read from the memory device and which must be performed by the computer(1).

The operator at the control panel selects in the address register the number c+1 (the number of that cell in which the first command of the program is stored) and then starts the computer. The control device will select the command from cell c+1, according to the cell number established at the address register, and after performing this command, it will add a one to the address register and in this

(1) Concerning the control device see Paragraph 2-9.

way the number c+2 is produced. Then the computer will start to perform the command stored in cell c+2 and so forth. In this way all commands of the program are performed in a consecutive order.

However, sometimes it will be necessary that the computer performs the command stored in the cell with the number p, after having performed the first k commands of the program, i.e. after having performed the commands stored in cells c+1 - c+k. In other words, the sequential order of performing commands is interrupted, for example, is it necessary to return to a previously performed command. For this purpose the so-called operation of the "unconditional transfer" or the "unconditional transfer of control" is used. This operation (TC) is written in the following manner:

c + k	TC	p	-	-
c + k + 1		...	...	...

and is read: "transfer control to the command located in cell p".

Performing the operation of unconditional transfer consists of transmitting the number p into the address register, after which the ordinary addition of a one will not occur.

Consequently, after the command of unconditional transfer of control (the command stored in cell c+k) we do not change over to the command stored in the subsequent cell c + k + 1 as usual, but to the command stored in cell p, the number of which is indicated by the first address of the command of unconditional transfer of control. In this way, an unconditional switching of the computing process is achieved from one section of the program to another one each time when we arrive at the command stored in cell c + k.

Besides the afore-mentioned unconditional transfer of control, there is the so-called conditional transfer of control, whereby the operation of comparing two numbers is used for performing the transfer which has the following appearance (we assume that this command is stored in a cell with the number m):

m	<	α	β	γ
m + 1	....	....	....	....

This operation has the following purpose: if a number located in cell α (according to the first address) is smaller than the number stored in cell β (according to the second address), then the next one performs the command stored in cell γ, and not the one stored in cell m + 1, which is indicated by the third address of the operation of comparison. If this number stored in cell α is greater

or equal to the number stored in cell β , then the next one performs the command from cell m + 1. It is said that the operation of comparison "sends off" to the third address in case the first number is smaller than the second one, and "passes" - in the opposite case. Here, the switching of the count from one section of the program to another one takes place depending upon the performance of a given condition (relation of the magnitude of two numbers). Therefore, this operation is called the conditional transfer of control. As we will see in Chapter 3, the operation of the conditional transfer of control plays a principal role in the automation of the calculating process on digital computers.

Practically, the operation of conditional transfer of control is achieved on the basis of determining the sign of the difference of numbers which are stored in cells α (according to the first address) and β (according to the second address). If the sign of this difference is negative, then the number γ is transmitted to the address register of the control device, and if in the opposite case a one, as usual, is added to the same register.

The same conditional transfer of control may be achieved by comparing the moduli of two numbers, which 2sends off" in case the modulus of the first number is smaller than the modulus of the second one, and it "passes" in the opposite case. We will also introduce the operation of conditional transfer of control which will "pass" in case two numbers are equal and "send off" in the opposite case.

Finally, the operation stopping the computer after all calculations have been finished and after the program has been exhausted, belongs also to the control operations.

According to the explanations of this paragraph, it becomes evident in which way the control device of a computer distinguishes the cells in which the numbers (constants) are stored from those cells containing commands, although both types appear as ordinary numbers and do not show any differences as such.

Actually, we ourselves select manually at the control panel the number of the cell in which the first command is stored. Furthermore, the entire program is either located in consecutive memory cells, or it switches by itself the control to those cells in which the commands are stored until the entire program has been processed, while the last command will stop the computer.

1-9. Command code of a conventional computer.

We considered in the preceding paragraphs the basic operations which are to be performed by computers. Different computers may have different ranges of operations depending on the particularities of the respective designs.

Below, there is a table of operations for some conventional, three-address computer.

The table shows that for some operations, for example the operation of number transfer, not all three addresses are used. The code of such a command contains zeros in the digits of the missing address. If, for example, the operation of number transfer received the code designation 010001, then the code of the command for transferring a number from cell 0.....01 to cell 10.....01 will have the following form

$$\underbrace{010001}_{6}\ \underbrace{0.....01}_{12}\ \underbrace{00.....0}_{12}\ \underbrace{10.....01}_{12}$$

Command Code of a Conventional Computer

Nr	Symbol of Operation	IA	IIA	IIIA	Contents of Operation
1	+	α	β	γ	The number from cell α is added to the number of cell β and the sum is recorded in cell γ.
2	-	α	β	γ	The number stored in cell β is subtracted from the number of cell α and the difference is recorded in cell γ.
3	x	α	β	γ	The number from cell α is multiplied by the number from cell β and the product is recorded in cell γ.
4	:	α	β	γ	The number from cell α is divided by the number from cell β and the quotient is recorded in cell γ.
5	SK	α	β	γ	Adding command. In cell γ a number is formed, the order of which is equal to the sum of the orders of the numbers stored in cells α and β, while the mantissa is the sum of their mantissas.

STAT

Nr	Symbol of Operation	IA	IIA	IIIA	Contents of Operation
6	[ ]	α	β	γ	Take the integral or the fractional part. The fractional part of the number from cell α is recorded in a non-normalized form in cell β, while the integral part goes into the last digits of cell γ.
7	$\wedge$	α	β	γ	The number from cell α is logically multiplied by the number from cell β and the result is recorded in cell γ.
8	$\leftarrow$	α	k	γ	Shift to left. The number from cell α is shifted to the left by k digits and the result is recorded in cell γ. The right k digits of cell γ are filled with zeros.
9	$\rightarrow$	α	k	γ	Shift to right. The number from cell α is shifted by k digits to the right and the result is recorded in cell γ. The left k digits of cell γ are filled with zeros.
10	PCh	α	-	γ	Transfer number. The number from cell α is transferred to cell γ.
11	PCh	α	-	γ	Transfer by modulus. The modulus of a number from cell α is transferred to cell γ.
12	-PCh	α	-	γ	Transfer with reverse sign. The number from cell α is transferred to cell γ with reversed sign.
13	Print	α	-	-	The number stored in cell α is printed.
14	PU (TC)	α	-	-	Transfer of control. The command stored in cell α is performed next.
15	$<$	α		γ	Comparing numbers. If a number stored in cell α is smaller than the number stored in cell β, the command stored in cell γ is performed next. In the opposite case that command is performed next whose number is by one larger than the number of the given command.

STAT

STAT

Hr	Symbol of Operation	IA	IIA	IIIA	Contents of Operation
16	$\|<\|$	α	β	γ	Comparing moduli. If the modulus of a number stored in cell α is smaller than the modulus of the number stored in cell β, then that command is performed next which is stored in cell γ. In the opposite case that command is performed next the number of which is by one larger than the number of the given command.
17	$\neq$	α	β	γ	Comparing for equality. If the number stored in cell α is not equal to the number stored in cell β, then that command is performed next which is stored in cell γ. If these numbers are equal, then the command is performed, whose number is by one larger than the number of the given command.
18	Stop	-	-	-	The calculation is interrupted and the machine is stopped.

STAT

STAT

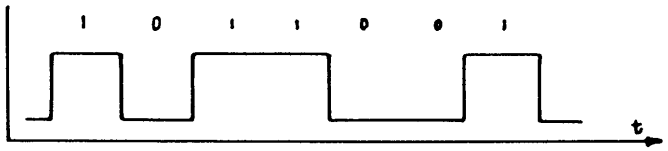
CHAPTER TWO.

Principles of Functioning of Automatic Digital Computers.

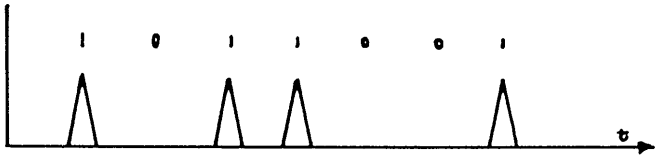
2-1. Conceptions of sequential and parallel codes.

Presently, the binary number system is used in the majority of digital computers. As it was said before, only two numbers are used in the binary system, the 0 and the 1, and consequently various elements (mechanical, electrical or magnetic) which have two stable states may be used for representing binary numbers. For example, an electromagnetic relay may serve as such an element. The switched-on state of such a relay may receive the designation 1, while the switched-off state means 0, or vice versa. In exactly the same manner, the blocked state of an electronic tube, when there is no current flowing thru it, may be given, for example, the number 1, while the open state of the tube receives the value 0.

In an electrical system, 0 and 1 are represented (coded) in the computer by high or low voltage levels at a corresponding point of the circuit arrangement, or by positive or negative electrical pulses, or by the absence or presence of pulses at corresponding terminals (pulse code). As an example, the number 1011001 is represented a) by the potential code and b) by the pulse code as shown by Figure 2-1.



a)



b)

Figure 2-1.

Let us consider some conceptions characterizing signals in the form of pulses and in the form of voltage levels.

The pulse may be characterized by the amplitude of the pulse U_A, the width (duration) of the pulse at the base t_0, the amplitude of

STAT

the overshoot U_v (Figure 2-2). The times of pulse build-up and droop, t_p and t_z, are called the front and rear pulse fronts respectively.

Analogous conceptions may be used for the potential signal (Figure 2-2b). The potential signal is characterized furthermore by the difference U_s of the upper and lower voltage levels. The conceptions of the forward and rear fronts of the potential signal are always linked with the transition processes either from the lower to the higher, or from the higher to the lower voltage levels respectively (compare Figure 2-2, b and c).

Two coding methods are used for numbers (and commands) in computers. For example, let us consider the code for the negative, ten-digit binary number 1.1011010011.

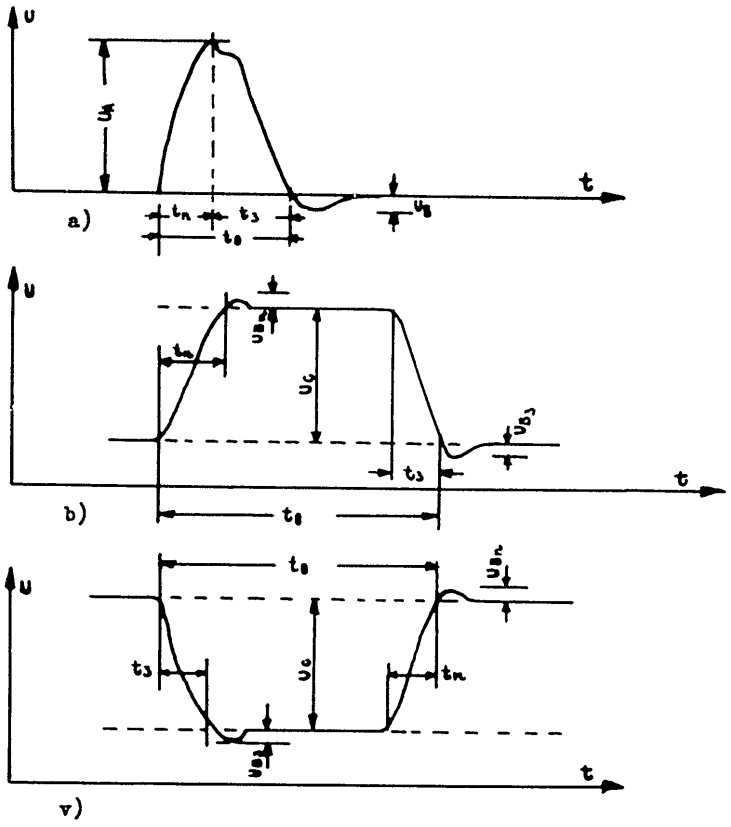


Figure 2-2.

Figure 2-3.

Here, the binary digit left of the decimal point indicates the sign of the number (the sign "plus" is represented by a zero, while a one stands for "minus"). The binary code of the number under consideration may be represented in the form of some time-sequential routine of pulses which are passing thru a circuit. The one may correspond to the presence of pulses, while the zero stands for the absence of the latter. Then, within determined intervals, which are created by auxiliary pulses, the signals will pass thru the circuits in a consecutive order one after the other, corresponding to the figures in the digits of the number (Figure 2-3). Such a method of coding numbers, where the code of the number is time-based, is called "consecutive coding" or "timing code".

With the other method of coding - with the "parallel coding" (or space code), the code of the number is not scanned within a time unit, but simultaneously in several electrical circuits (i.e. in space). The number of circuits is equal to the amount of digits of a number. At one and the same moment, signals originate in all circuits, indicating the value of the figures in the corresponding digits of a binary number (Figure 2-4).

Depending upon the code used, the computers are called computers with consecutive or parallel action. Computers with mixed action do also exist, in some blocks of which the parallel code is used (for example, in the arithmetic block), while the consecutive code is used in the others (for example, in the memory).

With the consecutive code system, the operations with numbers, among them the transfer of numbers from one device into another one, are performed alternately for each digit of a number and therefore the computers with consecutive action work slower than the computers with parallel action. However, the computers with parallel action require a larger amount of equipment, since with the parallel code it is necessary to have as many circuits for the transmission of signals, receiving and solving elements, as there are digits (under consideration of the digit for the sign) of the numbers with which the computer operates.

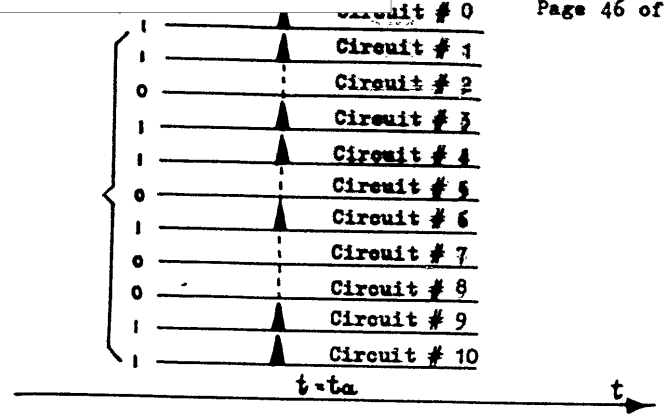


Figure 2-4.

2-2. Basic electronic elements of the automatic digital computer.

The trigger or the electronic relay is one of the basic elements of electronic digital computers. The circuit of a trigger (Figure 2-5) consists of two vacuum tubes, of which the anodes and grids are interconnected by feedback circuits.

With a reduction of the anode current passing thru the vacuum tube, also the voltage drop is reduced at the resistance in its anode circuit, and consequently the anode voltage is increased. Vice versa, the increase of the anode current is accompanied by a reduction of the voltage at the anode of the tube.

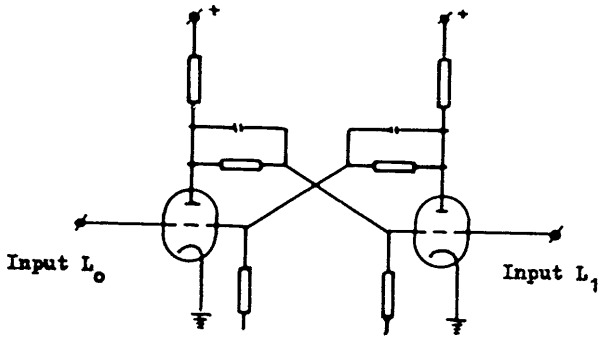


Figure 2-5.

In a trigger circuit, an anode current reduction of the first tube causes a potential increase at the grid of the second tube because of the presence of the feedback circuits. In turn, the increased current of the second tube leads to a potential reduction at the grid of the first tube which causes a further current reduction at the first tube, and so on. This process proceeds with avalanche speed until the first tube is not blocked completely and the second tube does not open. Then, the grid of the first tube receives negative potential and the grid of the second tube positive potential.

If a negative impulse enters the grid of the second tube which had a positive potential up to that time, then the current of the second tube begins to decrease and the system is "reversed": the first tube is opened while the second tube is blocked.

In this way, a trigger circuit has two stable states, whereby the anode current of tube L_0 is high in one of them while that of tube L_1 is low, and in the other state it is vice versa. If we designate one of these stable states with a zero and the other one with a one, then each digit of a binary number may be represented by a corresponding trigger. Let us assume that the state of a trigger corresponds to zero when tube L_0 has a high anode voltage and to a one when the anode voltage is high at tube L_1.

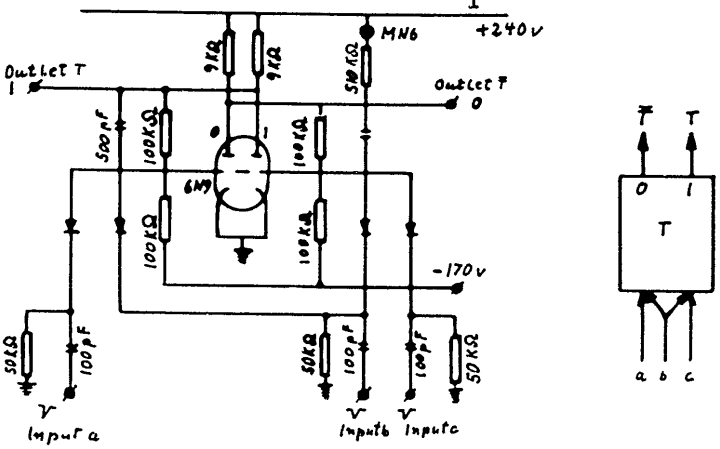


Figure 2-6.

The control of trigger circuits is achieved by feeding negative pulses to the grid. For bringing a trigger into state 0, it is sufficient to feed a negative pulse to the grid of tube L_0. If the trigger was in state 1 prior to feeding the pulse, then the negative pulse will change it to state zero. Exactly in the same way it is sufficient to feed a negative pulse to the grid of tube L_1 for bringing a trigger into state 1.

For representing a binary number, as many triggers are required as there are digits in a number and each trigger must be brought to state 0 or 1 depending upon the value of the corresponding digit. Such a system of several triggers which is to be used for storing a binary number is called a trigger register.

If the grids of both trigger tubes are connected by a valve and if negative pulses are fed to a common point, then each pulse will change the system from one stable state to the other, while the state of the system will indicate whether the total amount of pulses

STAT

was an even or an odd number. It is easily noticed that the trigger works as a binary counter in the case under consideration.

The properties of the triggers, which permit to use them for storing binary numbers or for building counters are the reasons for the wide-spread application of trigger circuits in digital computers. Figure 2-6 shows one of the trigger circuits used in practice(1). At the right side the conventional representation of a trigger is shown as it is used in functional diagrams of computer units. The voltage at the anode of the closed tube of the trigger shown by Figure 2-6 amounts to approximately 200 volts, while it is around 100 volts at the anode of the open tube. The trigger has two outlets and three inputs. A high voltage level at the outlet T is observed when the trigger is in state 1. In state 0, a high voltage level will be observed at the circuit outlet T. A neon lamp is connected parallel to the outlet T. The lamp will go on, when the left trigger tube is open, i.e. when the trigger is in state 1. When feeding a negative pulse to the input, then the trigger will change to state 0. Exactly in the same manner, the trigger will be in state 1 after feeding a negative pulse to input b. If it is necessary to reverse the state of the trigger, then input b is used which serves both trigger tubes. Input b is called "counting input", since it is used when the trigger works in a counter circuit where each subsequent pulse fed into the trigger must reverse the trigger counter of the given digit to the opposite state. For controlling the trigger shown by Figure 2-6, triangular, pointed pulses are used, having an amplitude of 40 - 60 volts and a width of 3 - 4 microseconds at the base. The valves at the trigger input circuits divide the grid circuits of the trigger from the pulse source circuits. The time-voltage diagram for a trigger is shown by Figure 2-7.

In the aforementioned trigger, the state 0 or 1 is determined by voltage levels at the anodes of the tubes. Recently, in electronic computers the socalled "dynamic" trigger found application in which the state of the circuit (0 or 1) is determined by the absence of the presence of oscillations (pulses) at the circuit outlet. Figure 2-8 shows the diagram of a dynamic trigger (L. 17).

---

(1) Figure 2-6 shows the trigger circuit of computer M-3 (L.3)

STAT

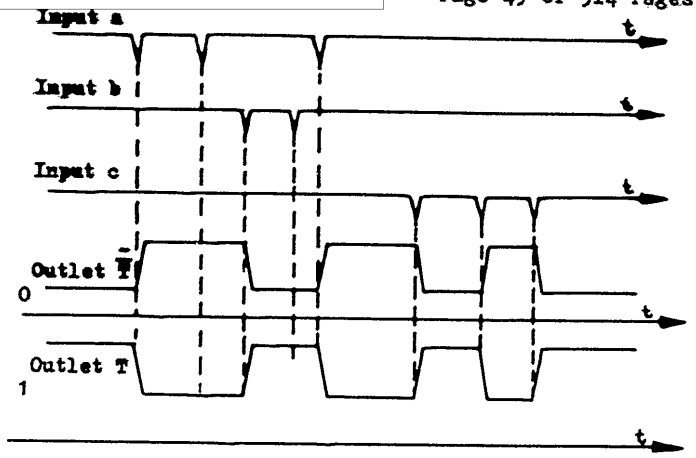


Figure 2-7.

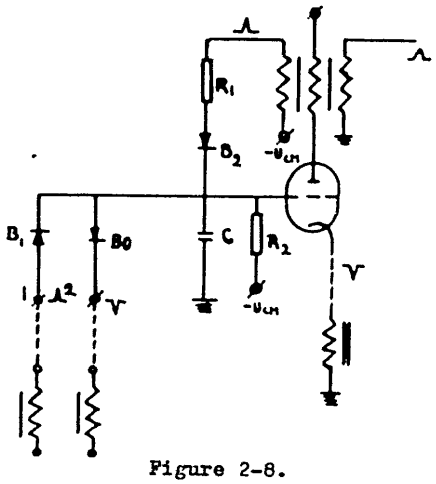


Figure 2-8.

The cathode of the tube receives uninterruptedly synchronized negative pulses which reduce the cathode voltage. The trigger is set to state 1 by feeding a positive pulse to terminal 1. The pulse charges the capacitor C thru the diode B_1, whereby the grid potential of the tube is sharply increased. In this case, the magnitude of the negative synchronizing pulses at the cathode is adequate for opening the tube. Thereby, positive pulses are generated in the secondary windings of the output transformer. From one secondary winding, the positive pulses enter the circuit of the computing device, and from the other one they achieve a charge of the capacitor C passing thru resistor R_1 and diode B_2 and thereby the operating conditions of the circuit are maintained.

The trigger is changed to state 0 by feeding negative pulses to terminal 2. Thereby capacitor C is discharged thru diode B_0. The blocking voltage at the grid of the tube increases sharply and the synchronizing pulses cannot pass thru the tube.

STAT

The capacitor C acts as some memory cell in the considered circuit. The voltage level of the capacitor changes with a change in the state of the circuit and this voltage may also be used as anode voltage of conventional (potential) triggers for control of the other elements of the computer.

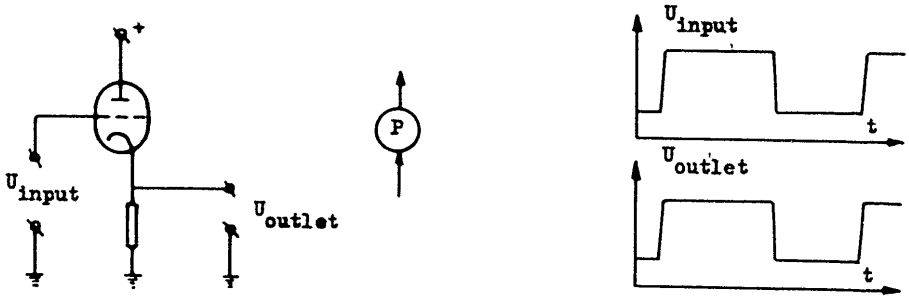


Figure 2-9.

When using high-speed triggers for controlling the external circuit of the voltage level of capacitor C, as shown by Figure 2-8, then the frequency of the synchronizing pulses is not determined, but the duration of entering pulses of zeros and ones.

The circuit of the dynamic trigger contains only one tube, Figure 2-8. The advantages of such a circuit are high input and low output resistances, comparatively low requirements for the stability of tube and circuit element parameters, the possibility to obtain signals with great front steepness in circuits with low-power and economical tubes.

The control of the different operations in computers is based on combinations which act on the electronic circuits and which consist either of electric pulses or voltage levels, or finally, of electric pulses and voltage levels simultaneously. Therefore, different elements for forming and converting electric impulses and voltage levels (voltage elements) are widely used in electronic computers together with devices for performing certain control laws, or, more precisely, logic laws[1]. For example, the aforementioned trigger (Figure 2-6) converts short, negative control pulses entering its input to certain voltage levels at the output terminals of the circuit.

In many cases, the voltage levels of the trigger anodes are fed into an external circuit thru intermediate elements - cathode followers. The cathode repeater (Figure 2-9) is a dc amplifier with a load connected to the cathode circuit. When feeding positive high-level

---

[1] These devices will be considered in the following paragraph.

STAT

voltages to the grid of the cathode follower, the tube of the follower is opened, the current passing thru the tube is rising and increases the voltage drop at the resistance in the cathode circuit of the tube. If a low voltage level is fed to the grid of the tube, the latter is blocked by the dc bias voltage, the current passing thru the load resistance decreases, and, consequently, the voltage drop is reduced. In this way, the high level of input voltage corresponds to a high level of output voltage and vice versa.

The cathode follower has a high input and a low output resistance and therefore it is used as a power amplifier for the control signals. The connection of the cathode follower to the trigger outlet makes the work of the latter independent of the load circuit parameters. Figure 2-10 (L.4) shows the diagram of the trigger - cathode follower block used in the M-2 computer.

The power amplification of control signals with a simultaneous change of the signal power level is also achieved by using a dc amplifier with an anode load, also called inverter (Figure 2-11). In this circuit, the low level of the output voltage corresponds to a high level of input voltage and vice versa.

In a number of cases it becomes necessary to obtain rectangular pulses in control circuits. The half-wave multivibrator circuit is used for this purpose. Sometimes this circuit is called "kipp relay" (Figure 2-12).

A positive voltage is fed thru the resistance R to the grid of the right triode. Therefore, during the absence of signals at the input, the right tube is open all the time, while the left tube is blocked. A low voltage level is observed at outlet 1, while there is a high voltage level at the terminal of outlet 2.

When feeding a negative pulse to the input, the right tube is blocked, while the left one is opened. Then, within a certain time, determined by the time constant $\tau = RC$, the potential at the grid of the right tube increases again to such a value that the right tube is opened and the left one blocked. At the terminal of Outlet 1 and Outlet 2, rectangular pulses are obtained, the duration of which may be controlled by changing the magnitudes of R and C.

The kipp relay is used in pulse shaping circuits and for building time delay circuits.

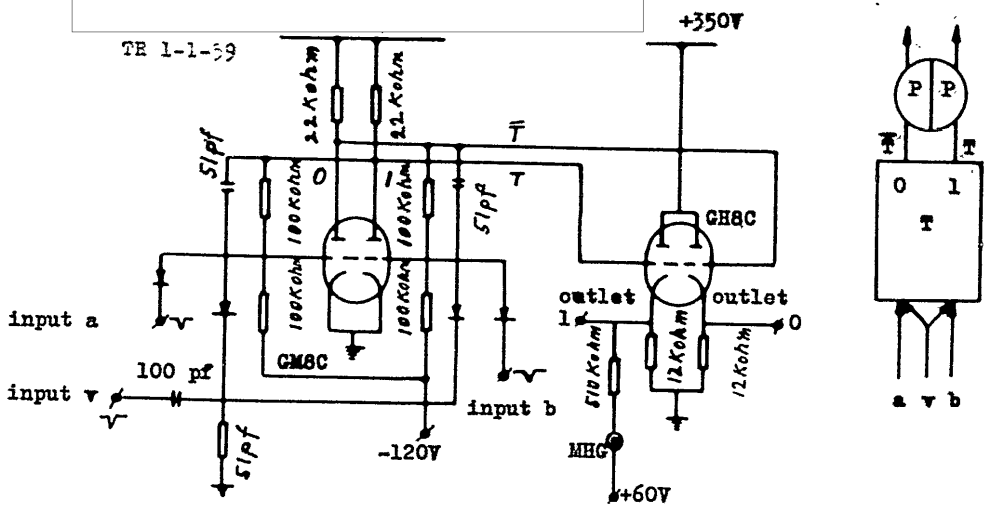


Figure 2-10.

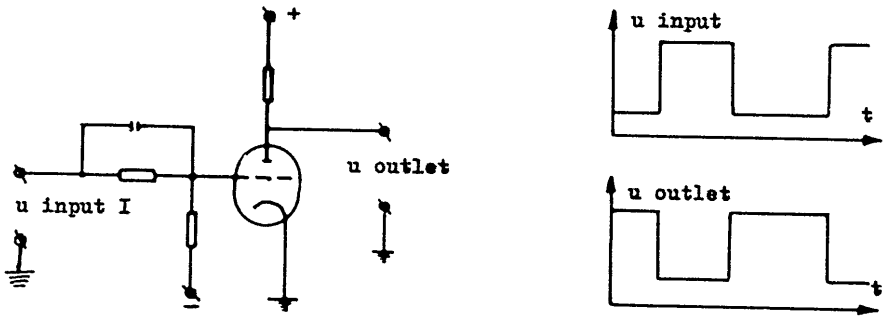


Figure 2-11.

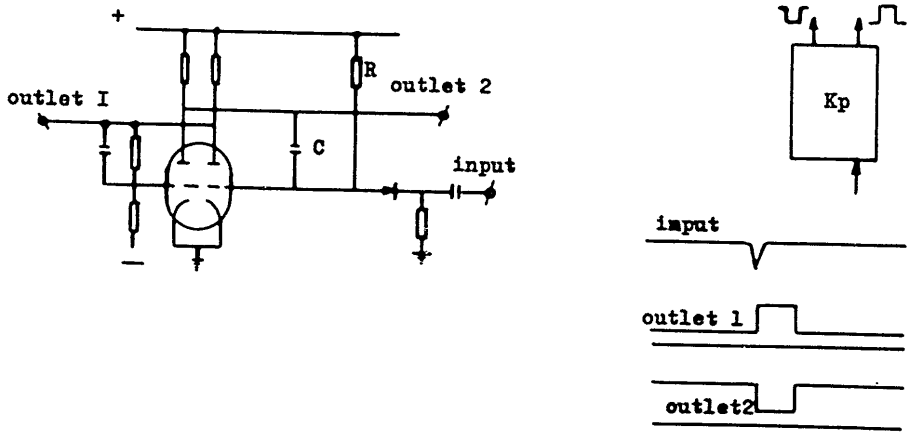


Figure 2-12.

Pulses in control circuits may be obtained by differentiating signals (of outlet voltages) of triggers or kipp relays by means of differentiating circuits RC. The positive pulse will correspond to the forward front of the signal and the negative pulse to the rear front (Figure 2-13). Frequently, a differentiating circuit is used at the input circuits of triggers, kipp relays and other devices for improving the impulse front at the input (Figure 2-6). Special electronic

pulse shaping networks are used in all corresponding cases for shaping pulses in control circuits. The shape and the amplitude at the outlet of the shaping circuit depends only to a small extent on the parameters of the inconing pulse. This is very important for the reliability of the control circuits. The shaping network (Figure 2-14) consists of a differentiating circuit and an electronic oscillator with impact excitation(blocking oscillator).

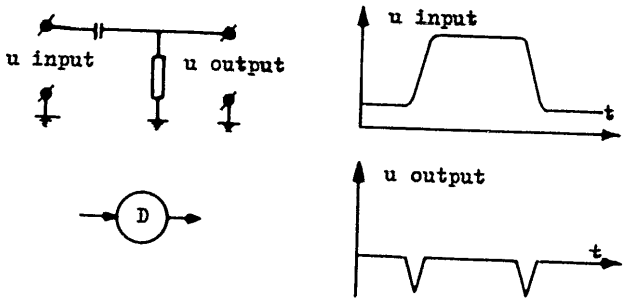


Figure 2-13.

A pulse transformer, whose secondary winding is shunted by a diode, is connected to the anode circuit of the tube. When feeding a positive pulse to the grid, the tube will open suddenly and powerful oscillations will arise in the oscillatory circuit formed by the inductance and the distributed capacitance of the pulse transformer. These oscillations are determined by the parameters of the oscillatory circuit in the second half-cycle and the oscillations in the circuit are interrupted.

The negative and positive pulses are taken from the terminals corresponding to the primary and secondary winding of the transformer. The right side of Figure 2-14 shows the conventional representation of a shaping network.

Transistors instead of vacuum tubes are used for increasing the reliability of the operation of computers and for reducing their weight and power consumption.

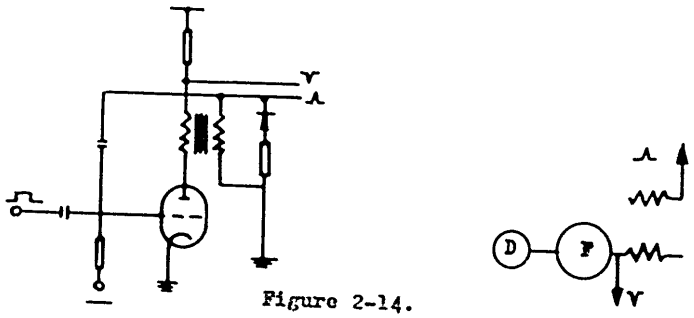


Figure 2-14.

STAT

2-3. Circuits for performing elementary logical operations.

The circuits for controlling the different operations of a computer are based on the use of various combinations of a small number of circuit types which perform elementary logical operations. Basically, three circuit types are used: a) separation; b) coincidence and c) negation. These circuits are built with semiconductor valves and vacuum tubes. The elementary logical circuits with semiconductor valves (coincidence and separation circuits) are also called decoders. The logical circuits create certain control signals (potential- or pulse-type) when certain conditions concerning signals are filled at the inputs of these circuits.

S e p a r a t i o n c i r c u i t s or mixer circuits have one outlet and several inputs. Figure 2-15 shows a separation circuit composed of semiconductor valves with two inputs. Let us assume that a high voltage level at the input and at the outlet of the circuit corresponds to a one, while the low voltage level stands for a zero. If there is a low voltage level at all inputs, then there will be a low voltage level at the output of the circuit. If an input signal consisting of a high voltage level is fed to any of the inputs or to some of them, then a current will arise and a voltage drop in the resistor and a high voltage level will appear at the circuit outlet.

The separation circuits work according to the logical law OR. A signal, corresponding to a one will appear at the outlet, if at input a or at input b (or the others, if there are more than two) the signal of a one is applied. The valves in the circuit under consideration separate incoming signal circuits and prevent that signals from one circuit enter the signal source of another circuit.

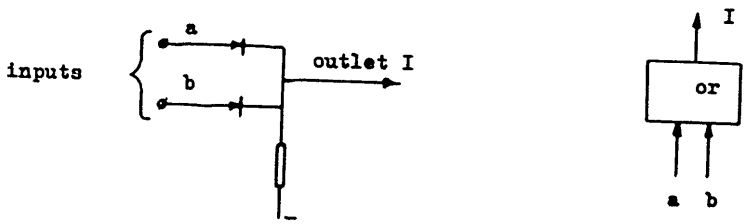


Figure 2-15.

The separation circuits work with potential and pulse input signals.

N e g a t i o n c i r c u i t s have one input and one output

STAT

and have the signal for the one at the outlet, if the signal of a zero enters the input and vice versa. The inverter circuit (Figure 2-11) may be used as a negation circuit. When feeding a low-level voltage to the input, the tube is blocked and its anode voltage (circuit outlet) has a high level. In turn, at a high level of input grid voltage the tube is opened and the anode voltage drops to its lower level.

The negation circuits work according to the logical law NO. If the signal of a 1 is fed to the input, then we will have at the terminator the signal "no 1", i.e. zero. If the signal of a zero is fed to the input, then at the terminator we will have the signal "no zero", i.e. a one.

Negation circuits are represented on functional diagrams by a rectangle with the inscript NO.

Coincidence circuits have one output and several inputs. Figure 2-16 shows one of the possible variations of a coincidence circuit composed of semiconductor diodes. If a low voltage level U_H is fed to all inputs then a current will pass thru the resistor R and the diodes, while the voltage at the outlet will be equal to the voltage U_c after deduction of the voltage drop at the resistor R which will correspond to the low voltage level. If to some, but not to all inputs, a high-level voltage U_B is fed, then the corresponding valves will be blocked and no current will pass thru them. However, current will continue to flow thru those valves to which low-level voltages are being fed from the input side and if the resistance in the valve circuits are much lower than R, then there will be a low voltage level at the outlet as previously because of the voltage drop at the resistance R.

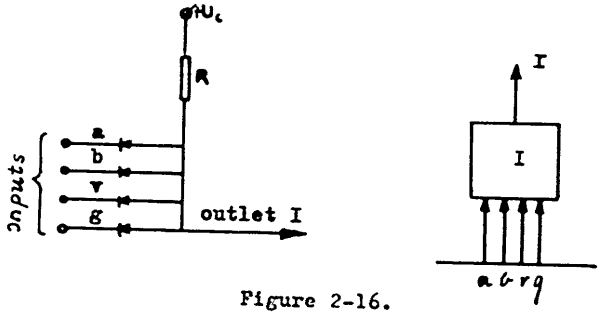


Figure 2-16.

If a high-level voltage U_B is fed to all inputs, then all valves will be blocked and the voltage at the outlet will rise to the value corresponding to a high level. The coincidence circuit works according to the logical law AND. We will find at its output the signal "one"

only in case the signal "one" is fed to all inputs a, b, ...

Coincidence circuits composed of semiconductor valves are usually applied in circuits with potential input signals.

When a coincidence circuit is required to which one input signal is given in the form of a voltage level while the other one is a pulse signal, or when both signals consist of pulses, then frequently an electronic circuit called "pulse valve" is used. Figure 2-17 shows one type of pulse valve circuits with a pentode for practical application. The first and the third grid are used for controlling the tube. The voltage level is fed to the first grid ("control voltage"), usually coming from the outlet of a logical network composed of semiconductor valves or from a trigger circuit, while the pulse is fed to the third grid ("selector pulse").

The circuit and input signal parameters are selected in such a way that the tube is opened by a positive pulse at the third grid only when there is a high-level voltage at the first grid.

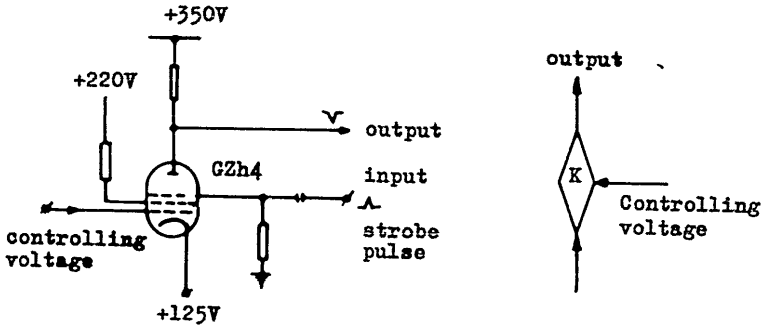


Figure 2-17.

In this way a powerful negative pulse is obtained at the anode of the tube. Feeding a high-level voltage to the first grid during the absence of a pulse on the third grid, or the presence of a pulse at the third grid at a low voltage level at the first grid, do not open the tube.

The valve may function as a coincidence circuit for positive pulses at the first and third grids. It is used in the same way as the pulse shaping network.

If a positive pulse is required at the output, a valve with a pulse transformer is used in the anode circuit with shunting of the secondary winding by a semiconductor valve analogously to the circuit shown by Figure 2-14.

The non-conformity circuit ("skhema nesootvetstiya" - "decoder of non-conformity") is widely used in various computers (for example M-2, M-3) as an elementary, logical type-circuit.

The non-conformity circuit consists of two coincidence circuits with three inputs each, which are connected with each other in a peculiar way. Voltage levels are fed to two inputs of each coincidence circuit, taken from opposite trigger outlets, while the signal of the control voltage is fed to the third input which is common for both circuits (Figure 2-18).

If a high-level voltage is fed to the terminals connected with the valves V_1, V_2 and V_7*, these valves and also V_5 will be blocked. A high-level voltage is obtained at the output. If a low-level voltage was fed to one of the indicated valves, then a current would flow thru the resistor R_1 and because of the voltage drop at R_1, the level of the outlet voltage would be reduced.

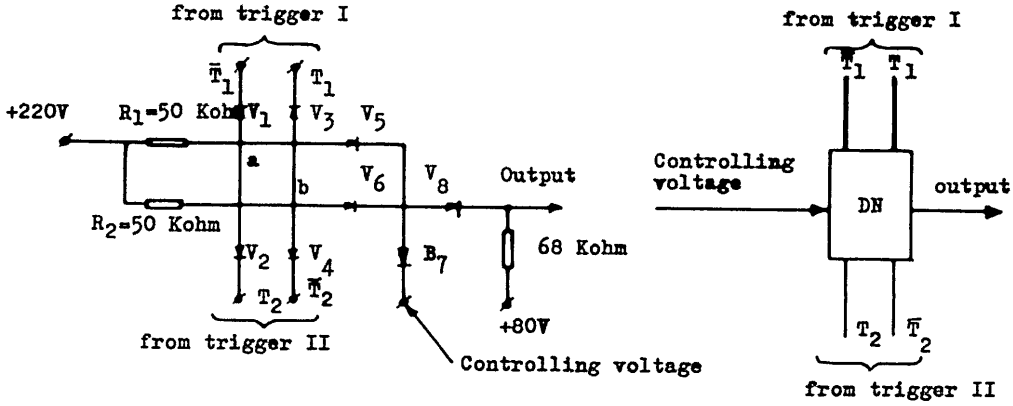


Figure 2-18.

The circuit formed by valves V_3, V_4, V_6, V_7 and the resistor R_2 works in exactly the same way. Consequently, a high voltage level will be obtained at the outlet of the non-conformity circuit only in case the following requirements are met simultaneously: a) both triggers, whose outlets are connected to the inputs of the non-conformity circuit, must be in different positions; b) a high-level control voltage must be fed to the additional input of the circuit.

2-4. The performance of some operations by Means of logical circuits.

In the present paragraph we will consider the performance of the following operations by means of simple logical circuits: the transfer

* If the first trigger was in position 0, and the second in position 1, then a high-level voltage is applied at valves V_1 and V_2.

of a number from one system into another one, the transfer of a code from one trigger register to another one and the shift of code.

Figure 2-19 shows a diode matrix for converting numbers from the binary system into the octal system. The circuit consists of eight coincidence circuits having three inputs each. The inputs of these circuits are connected in the present example to the outlet terminals of three triggers, corresponding to the three digits of a binary number. By means of three triggers any number from zero to seven may be transformed to the binary system. We remember that the high-level voltage of the trigger will be at terminal 0(T) or 1(T), if the trigger is correspondingly in states 0 or 1. It is easy to check that in the circuit shown by Figure 2-19, the high voltage

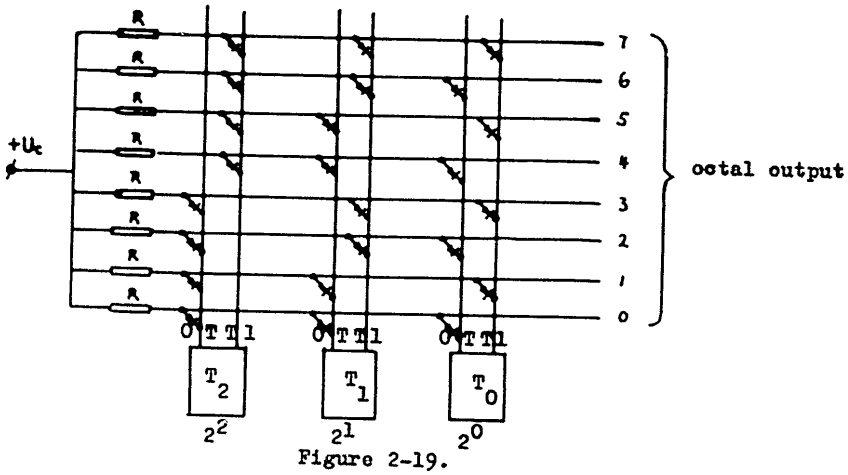


Figure 2-19.

level will be at the bus bar whose number corresponds to the binary numbers established at the trigger register T_0-T_1-T2.

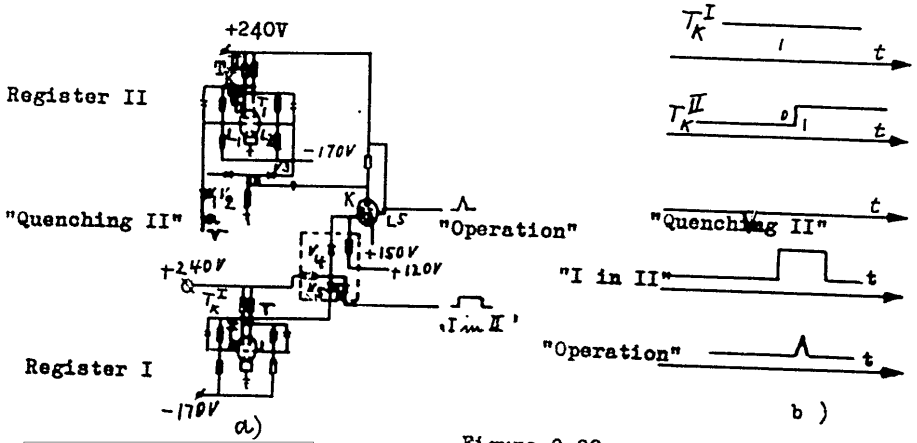


Figure 2-20.

Networks, analogous to Figure 2-19, may be used for selecting circuits corresponding to the code at the network input.

Let us consider the network for transferring the codes of numbers from one trigger register to another. Figure 2-20a represents such a circuit which is to be used on the M-3 computer (the circuit for one digit is shown). This network consists of the triggers T_k^I and T^{II} of the k-th digit of registers I and II, the coincidence circuit "AND" composed of semiconductor valves and the pulse valve K.

When transferring the code of a number from register I to register II it is necessary to bring in each digit the triggers of register II to the same state as those of register I. Preliminarily the control circuit will transmit a negative pulse "CLEAR" to the triggers of register II, whereby all triggers of this register are placed in zero position.

The auxiliary signal "I and II" is then transmitted to the input of the "AND" circuit as a wide rectangular pulse corresponding to a high voltage level. The voltage level from outlet "one" of trigger T_k^1 of register I is fed to the second input of the "AND" circuit. A low voltage level will enter the "AND" circuit, if this trigger is in position 0 and when a low-level voltage will be at the outlet of the "AND" circuit (at the first grid of valve K). When feeding the pulse "Operation" to the first grid of the tube, the tube does not pass the pulse and the trigger T_k^{II} remains in state 0.

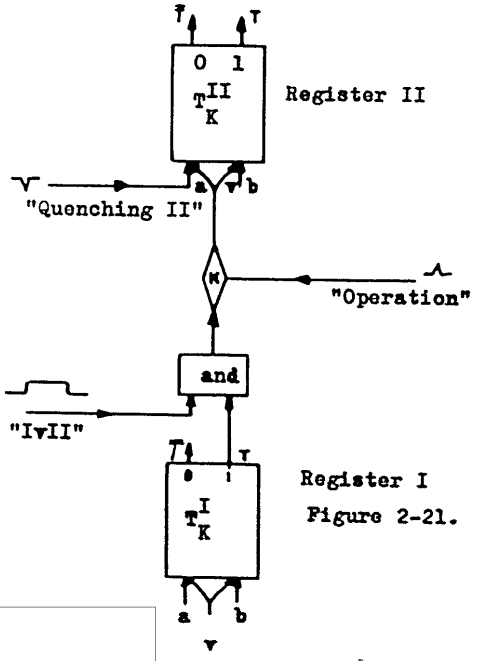


Figure 2-21.

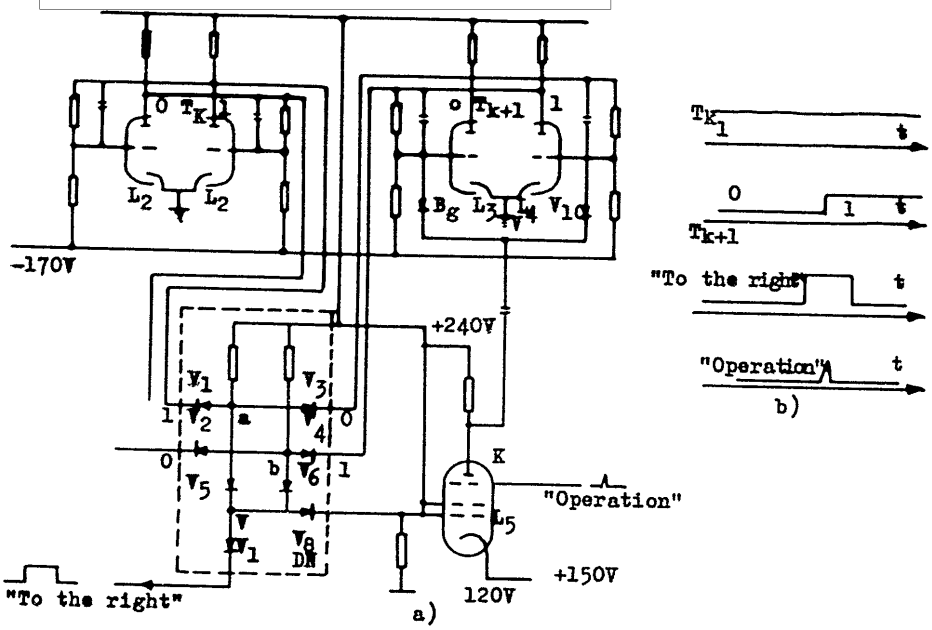


Figure 2-22.

If trigger T_k^I of register I is in state 1, then a high voltage level will enter from its outlet the input of the "AND" circuit and further to the first grid of the valve K. In this case, when feeding the positive pulse "Operation" to the third grid of the valve, a negative pulse will originate at the anode of the valve, changing the trigger of register II to the state 1, which is that of the corresponding trigger in register I in the equivalent digit. Figure 2-20b shows the time graph for the circuit under consideration and Figure 2-21 represents the corresponding block diagram.

Let us consider the circuit for shifting the number code to the right in a register, which is being used in the M-3 computer. For simplification only two triggers of the k-th and (k + 1)-th digit are shown by Figure 2-22a, and further, those input circuits of the triggers were left out which do not participate in the operation under consideration.

When shifting to the right by one digit, the trigger T_{k+1} must be brought into that state in which trigger T_k was prior to the shifting. The state of triggers T_k and T_{k+1} is compared by the nonconformity circuit DN. Let us assume that the triggers are in different states. In this case, a high-level voltage is fed to the first grid of valve K, when feeding to the control input of the nonconformity circuit the positive, auxiliary, wide, rectangular signal "To the right", corresponding to a high voltage level. When

feeding to the third grid of the tube L_5 the positive pulse "Operation", a negative pulse arises at the anode of the tube, reversing the state of trigger T_{k+1}, i.e. changing it to the state of trigger T_k.

If triggers T_k and T_{k+1} are in an identical state, then there will be a low-level voltage at the first grid of valve K and the transmission of pulse "Operation" will not be accompanied by the appearance of a pulse at the input of trigger T_{k+1}. The state of trigger T_{k+1} remains unchanged. Figure 2-22b shows the time graph of the circuit under consideration and Figure 2-23 shows the block diagram.

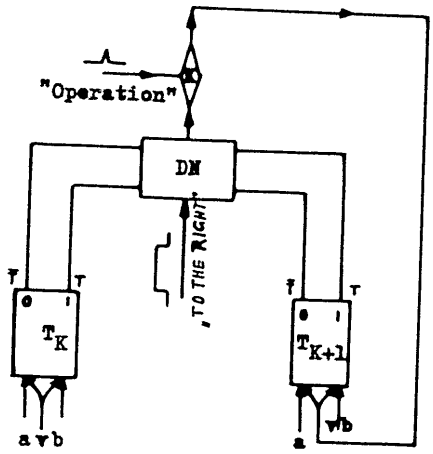


Figure 2-23.

2-5. Particularities of performing arithmetic operations on computers. Conceptions of the complementary and reverse codes.

In the present paragraph we will consider in which way the performance of all arithmetic operations on computers may be reduced to addition operations.

As we already indicated before, the sign plus is represented in a computer by a "zero" in the sign digit, while a "one" stands for the sign minus. A positive binary number with a decimal point fixed in front of the top digit:

$$G^+ = +0.\ \gamma_1\gamma_2 \ldots\ldots \gamma_n \quad (\gamma_i = 1 \text{ or } 0) \qquad (2\text{-}1)$$

is represented in a computer in the following way:

$$(G^+)_{dir} = 0.\gamma_1\gamma_2 \ldots\ldots\gamma_n\ . \qquad (2\text{-}2)$$

Analogously the negative binary number

$$G^- = 0.\gamma_1\gamma_2 \ldots\ldots \gamma_n \qquad (2\text{-}3)$$

is represented in a computer in the following way:

$$(G^-)_{dir} = 1.\gamma_1\gamma_2 \ldots\ldots \gamma_n \cdot \qquad (2\text{-}4)$$

The method of representing numbers (2-2) and (2-4) is called direct code representation according to positive and negative numbers.

The subtraction operation is reduced to a simple arithmetic addition operation by means of special codes (reverse or complementary codes), which are to be used for representation of negative numbers in a computer.

In order to represent a negative binary number (2-3) by a reverse code, it is necessary to place a "one" in the sign digit, while in all other digits the ones are replaced by zeros, and the zeros by ones:

$$(G^-)_{rev} = 1.\sigma_1\sigma_2 \ldots\ldots \sigma_n , \qquad (2\text{-}5)$$

whereby $\sigma_i = 1$ when $\gamma_i = 0$ and $\sigma_i = 0$ when $\gamma_i = 1$.

When recording a negative number in the complementary code, a "one" is placed in the sign digit, while the numerical part of the number is replaced by supplementing the modulus of a number to a whole unity. The negative number $G^- = 0.\gamma_1\gamma_2 \ldots\ldots \gamma_n$ has the following appearance in the complementary code:

$$(G^-)_{comp} = 1.\varepsilon_1\varepsilon_2 \ldots\ldots \varepsilon_n , \qquad (2\text{-}6)$$

whereby $$0.\varepsilon_1\varepsilon_2 \ldots\ldots \varepsilon_n = 1 - 0.\gamma_1\gamma_2 \ldots\ldots \gamma_n . \qquad (2\text{-}7)$$

It is easily noticed that

$$(G^-)_{comp} = (G^-)_{rev} + 2^{-n} , \qquad (2\text{-}8)$$

whereby n indicated the amount of digits of a number. In this way, the complementary code may be obtained from the reverse code by adding to it the unity of the power digit .

Let us establish the connection between the same negative numbers G^- with the numbers $(G^-)_{rev}$ and (G^-) comp, which are represented by the reverse and complementary codes. Subtracting (2-3) from (2-5) we have

$$(G^-)_{rev} - G^- = 1.\sigma_1\sigma_2 \ldots\ldots \sigma_n - (-0.\gamma_1\gamma_2 \ldots\ldots \gamma_n) =$$

$$= 1.11 \ldots\ldots 1 = 2 - 2^{-n}$$

because $\sigma_i + \gamma_i = 1$.

STAT

Consequently,

$$(G^-)_{rev} = G^- + 2 - 2^{-n} . \qquad (2\text{-}9)$$

Subtracting (2-3) from (2-6) we have

$$(G^-)_{comp} - G^- = 1.\varepsilon_1\varepsilon_2 \cdots\cdots \varepsilon_n - (-0.\gamma_1\gamma_2 \cdots\cdots \gamma_n) .$$

Considering (2-7) we obtain:

$$(G^-)_{comp} = G^- + 2 . \qquad (2\text{-}10)$$

By using the reverse or the complementary code, the subtraction and addition operations of numbers with different signs may be reduced to a plain addition of the codes of numbers.

Let us consider the use of the reverse code for the algebraic addition of two numbers G and Q, when one, or both numbers are negative. In this case the following rule may be formulated (we assume that the modulus of the algebraic sum is smaller than one).

For the algebraic addition of two binary numbers with application of the reverse code, positive addends are represented by the direct code, while the negative addends are shown by reversed code and the arithmetic summation of these codes is performed including the digits of the signs, which are considered in this case as the digits of whole unities. When a carry-over is required, from the sign digit, the unity of the carry-over is added to the lower digit of the sum of the codes[1]. In the result, an algebraic sum is obtained, in direct code if the sum is positive, in reverse code if it is negative.

This rule is easily checked. Let us take the n-digit, binary numbers $G^- < 0$ and $Q^- < 0$ (with a fixed decimal point). Their algebraic sum is $-1 < G^- + Q^- < 0$. In accordance with (2-9) we may write:

$$(G^-)_{rev} + (Q^-)_{rev} = G^- + 2 - 2^{-n} + Q^- + 2 - 2^{-n} =$$

$$= [2] + (2 - 2^{-n}) + (G^- + Q^-) - 2^{-n} .$$

The 2, in brackets, corresponds to the carry-over from the sign digit. In this case, according to the formulated rule, the carry-over unity is added to the lower digit of the sum.

As a result we obtain:

$$(G^-)_{rev} + (Q^-)_{rev} = (G^- + Q^-) + (2 - 2^{-n}) - 2^{-n} + 2^{-n}$$

or

$$(G^-)_{rev} + (Q^-)_{rev} = (G^- + Q^-) + 2 - 2^{-n} ,$$

---

[1] This operation is called end around carry.

STAT

from where according to (2-9)

$$(G^-)_{rev} + (Q^-)_{rev} = (Q^- + Q^-)_{rev} . \qquad (2\text{-}11)$$

In this way, the addition with the end around carry of the reversed codes of two negative numbers produces the reverse code of their algebraic sum.

Now, we will consider the algebraic sum of two numbers - the positive G^+ and the negative Q^-. Then

$$(G^+)_{dir} + (Q^-)_{rev} = G^+ + Q^- + 2 - 2^{-n} . \qquad (2\text{-}12)$$

If $(G^+ + Q^-) > 0$, then $2 + (G^+ + Q^-) > 2$ and the carry over begins from the sign digit. Since this is an end around carry, a 2^{-n} is to be added instead of a 2 to the expression under consideration.

In the result we have:

$$(G^+)_{dir} + (Q^-)_{rev} = (G^+ + Q^-)_{dir}$$

$$(\text{when } G^+ + Q^- > 0). \qquad (2\text{-}13)$$

If $(G^+ + Q^-) < 0$, then $(2 + G^+ + Q^-) < 2$ and the carry-over does not start. In this case we obtain from (2-12):

$$(G^+)_{dir} + (Q^-)_{rev} = (G^+ + Q^-)_{rev}$$

$$(\text{when } G^+ - Q^- < 0). \qquad (2\text{-}14)$$

In this way the addition of the codes according to the presented rule leads to an algebraic sum in direct code if this sum is positive, and in the reverse code if it is negative.

Now we will consider the use of the complementary code for the algebraic addition and we will present the respective rule (we assume that the modulus of the algebraic sum is smaller than one).

For the algebraic addition of two binary numbers with the application of the complementary code, the positive numbers are represented in direct code, while the negative numbers are shown in the complementary code and the arithmetic summation of these codes is performed with inclusion of the sign digits, which are considered as digits of integer unities. When the carry-over from the sign digit is started, the unity of the carry-over is dropped. In the result an algebraic sum is obtained in direct code, if this sum is positive, and in complementary code if this sum is negative.

In the same way, if $G^- < 0$ and $Q^- < 0$, then according to (2-10)

$$(G^-)_{comp} + (Q^-)_{comp} = G^- + 2 + Q^- + 2 = 2 + (G^- + Q^- + 2) .$$

STAT

Since $-1 < G^- + Q^- \quad 0$, the magnitude within parenthesis is smaller than two, but larger than one. The two standing outside the parenthesis forms the carry-over from the sign digit which will be dropped. We obtain as a result:

$$(G^-)_{comp} + (Q^-)_{comp} = (G^- + Q^-) + 2 = (G^- + Q^-)_{comp} \qquad (2\text{-}15)$$

If $G^+ < 0$, and $Q^- < 0$, then $(G^+)_{dir} + (Q^-)_{comp} = (G^+ + Q^-) + 2$.

If thereby $0 < G^+ + Q^- < 1$, then the two in parenthesis will give the unity of carry-over from the digit sign, which is dropped and then we obtain

$$(G^+)_{dir} + (Q^-)_{comp} = (G^+ + Q^-)_{comp} \qquad (2\text{-}16)$$

(when $G^+ + Q^- > 0$).

If $-1 < G^+ + Q^- < 0$, then $(Q^+ + Q^-) + 2 \quad 2$ and the carry-over from the sign digit does not start. In this case

$$(G^+)_{dir} + (Q^-)_{comp} = (G^+ + Q^-)_{comp} \qquad (2\text{-}17)$$

(when $G^+ + Q^- < 0$).

In this way we showed that the subtraction operation, which is to be considered as an algebraic addition, may be replaced by the operation of transforming the direct codes of negative numbers into reverse or complementary codes and by simple arithmetic addition of the corresponding codes of the numbers. If the result is obtained in the reverse or complementary code (the algebraic sum is negative), then, if so required, it may be transformed to the direct code.

For the multiplication (or division) of binary numbers, the sign of the product (or the quotient) is obtained as a sum (without consideration of the carry-over) of the sign codes of the multiplicand and the multiplier (or of the dividend and the divisor) (Table 2-1).

When multiplying binary numbers with a fixed decimal point, multiplicand and multiplier are represented in direct code.

The multiplication itself on computers is usually reduced to a succession of operations of adding the multiplicand to the sum of the, already previously calculated, partial product with a shift to the right by one digit of the sum obtained. The code of the multiplier is fed digit by digit (usually beginning with the lower one) to the circuit elements which checks the presence of a 1 or a 0 in the given digit. If there is a 1 in the given digit, the addition of the code of the multiplicand to the sum already located in the adder is performed. Then the shift to the right by one digit of the new sum is performed. If there is a 0 in the given digit of the

STAT

multiplier, the addition of the multiplicand is not performed, while only the code is shifted in the adder to the right by one digit.

The afore-mentioned statement is illustrated by the following example:

.1.101001 x 0.010011	Multiplicand (-0.101001) Multiplier (+0.010011)
1. + 0. 1.	Determination of the sign code of the product
000000 + 101001 101001	Clearing of the adder. Addition of the multiplicand to the code of the adder (a 1 is in the sixth place)
010100 + 101001 111101	Shifting the adder code to the right by one digit. Adding the multiplicand to the code of the adder (a 1 is in the fifth place of the multiplier)
011110	Shifting the adder code to the right by one digit. An addition of the multiplicand is not performed (a 0 stands in the fourth place of the multiplier)
001111	Shifting the adder code to the right by one digit. An addition of the multiplicand is not performed (a 0 stands in the fourth place of the multiplier)
000111 + 101001 110000	Shifting the adder code to the right by one digit. Adding of the multiplicand to the adder code (a 1 is in the second place of the multiplier).
011000	Shifting the adder code to the right by one digit. An addition of the multiplicand is not performed (a 0 stands in the first place of the multiplier).
001100	Shifting the adder code to the right by one digit.
1.001100	Product (-0.001100)

Table 2-1

Operations with signs	(+) x (+) = (+) (+) : (+) = (+)	(+) x (-) = (-) (+) : (-) = (-)	(-) x (+) = (-) (-) : (+) = (-)	(-) x (-) = (+) (-) : (-) = (+)
Operations with code signs	0 + 0 = 1	0 + 1 = 1	1 + 0 = 1	1 + 1 = 0*

* 1 + 1 = 0 + the carry-over unity to the top digit which is dropped when adding sign codes.

STAT

In computers with a fixed decimal point, a division is possible only in that case when the divisor is larger than the dividend, since in the opposite case, the quotient will be larger than 1 and an overfilling of the digits will occur. The division of binary numbers on computers is reduced to a sequence of operation of subtracting the divisor from the dividend or the remainder of the preceding subtraction and shifting the remainder to the left by one digit. Thereby the quotient is obtained consecutively digit by digit beginning with the top digit.

There are several methods for performing a division on computers. One of these methods consists of the following. If the remainder is positive, then a 1 is placed into the series digit of the quotient. Then the remainder is shifted to the left by one digit and the subtraction of the divisor is performed once more. If the remainder is negative, then in the respective digit of the quotient a 0 will be placed. The divisor is added to the negative remainder, i.e. the divisor restores the positive remainder. The restored remainder is then shifted to the left by one digit and the divisor is subtracted once more.

When subtracting the divisor, the complementary or reverse code of the divisor is added to the code of the remainder.

We will consider an example. We will use the complementary code for the subtracting operation (refer to the table of the right hand column).

As a result the direct code of the quotient was obtained 1.011111, which corresponds to number -0.011111.

In computers with a floating decimal point the performance of arithmetic operations becomes more complicated, since, besides the actual arithmetic operations with mantissas, which are performed in the same way as in computers with a fixed decimal point, corresponding operations with orders must be performed and several auxiliary operations - equalizing of orders by shifting the mantissa to the right and the normalization of numbers (refer to Paragraph 1-5).

When adding or subtracting numbers on a computer with a floating decimal point, the operation of equalizing the number order must be performed, shifting to the right the mantissa of the number with the smallest order by the number of digits, equal to the difference of the orders of these numbers. Then the mantissas are added or subtracted and the result is normalized. For the multiplication (division) the orders of the numbers are added (subtracted) while the mantissas are multiplied (divided). Then the result is normalized.

STAT

1.010110 0.101101	Dividend (-0.010110) Devisor (+0,101101)	Digits of the Quotient
1. + 0. 1.	Determination of the code sign of the quotient	
0.101100 + 1.010011	Shifting the dividend to the left by one digit. Divisor (complementary code)	
1.111111 + 0.101101	Remainder (complementary code) Remainder <0 Divisor (direct code)	0
0.101100 1.011000 + 1.010011	Restored remainder Shifting the remainder to the left by one digit (direct code). A whole unity is stored in the sign digit. Divisor (complementary code)	
0.101011 1.010110 + 1.010011	Remainder (direct code). Remainder >0 Shifting the remainder to the left by one digit (direct code) Divisor (complementary code)	1
0.101001 1.010010 + 1.010011	Remainder (direct code). Remainder >0 Shifting the remainder to the left by one digit (direct code) Divisor (complementary code)	1
0.100101 1.001010 + 1.010011	Remainder (direct code). Remainder >0 Shifting the remainder to the left by one digit (direct code) Divisor (complementary code)	1
0.011101 0.111010 + 1.010011	Remainder (direct code). Remainder >0 Shifting the remainder to the left by one digit (direct code) Divisor (complementary code)	1
0.001101	Remainder (direct code). Remainder >0	1

For the algebraic addition of the orders it is necessary to represent negative orders by the complementary or reverse code.

Negative numbers (mantissas) are frequently stored in direct code in the memory units of computers and they are used in such a form for the multiplication operation and partially also for dividing. For the algebraic addition or subtraction, negative numbers are pre-

STAT

liminarily transferred to the complementary or reverse code. The negative result of these operations is also obtained in the complementary or the reverse code and prior to placing them into the memory device they are converted to the direct code.

The negative orders of numbers are easier stored in a memory device in the complementary or the reverse codes, since only addition and subtraction operations are performed with the orders.

2-6. Arithmetic units.

The arithmetic unit is that part of the computer in which arithmetic and some other operations are performed on numbers.

In a computer with a floating decimal point there are according to its design two arithmetic units - one for the operations on the mantissas and the other for the operations on orders.

It was shown in the preceding paragraph, how, by means of special codes (complementary and reverse). all operations may be reduced to an addition. Therefore, the adder of the number codes is the basic part of an arithmetic unit, and in case it is a computer with a floating decimal point, it is also the adder of the order codes.

The arithmetic unit contains trigger registers in the form of blocks for a temporary storage of the number codes on which the operations are performed. In a number of cases, these trigger registers are frequently adders themselves.

We list the operations which may be usually performed by an arithmetic unit: 1) Setting the registers of the arithmetic block to 0 position; 2) receiving the codes of numbers and commands from the memory and placing them into the register of the arithmetic unit; 3) selecting the complementary or reverse code; 4) shifting the code of a number to the right or the left; 5) adding the codes of numbers; 6) transferring the codes of numbers to the memory unit.

The arithmetic units will perform also several other operations. For example, in the computer M-3, the register of the arithmetic unit is used for transferring the codes of numbers and commands to the memory when the program is put into the computer, for carrying codes from one memory cell to another one, for transmitting numbers from the memory unit to the output and printing machine.

In small computers with consecutive action, the arithmetic device performs only a part of the afore-mentioned operations.

We turn now to the description of the adder, the basic element of an arithmetic unit.

STAT

When adding two numbers independently of the basis, in each digit an addition of three numbers is performed: the figures of the given digit of the first summand, the figures of the given digit of the second summand and the numbers (1 or 0) of the carry-over from the neighboring lower digit. As a result of the adding operation of one digit, a figure of this digit in the sum and a figure (0 or 1) of the carry to the next top digit are obtained.

For examining the addition of two binary numbers it is adequate to retrace the operation for anyone of the digits of these numbers. Depending upon the value of the figures to be added and the presence or absence of carry unities from the preceding lower digit, the addition result will be different. Eight versions are possible which are listed in Table 2-2.

The first four versions correspond to an addition of the digits of two binary numbers during the absence of a carry unity from the lower digit.

It is easily noticed that the block diagram, shown by Figure 2-24, provides the performance of the addition operation for the first four versions of Table 2-2.

Table 2-2

Carry-over from the preceding lower digit	First addend	Second addend	Sum	Carry-over to the following higher digit
0	0	0	0	0
0	1	0	1	0
0	0	1	1	0
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1
1	0	1	0	1
1	1	1	1	1

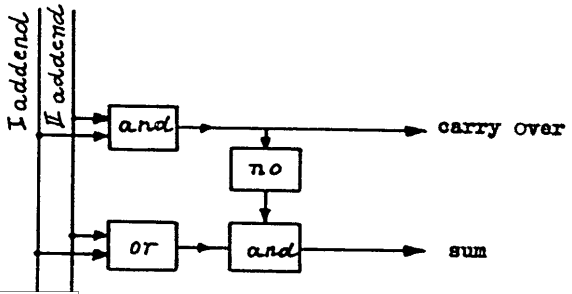


Figure 2-24.

STAT

For example, in the version corresponding to the fourth line of the table, the first and the second addend are equal to 1 and consequently positive pulses (pulses of the "one") will enter the respective buses simultaneously. A positive pulse will appear in this case at the outlet of the AND circuit which is also connected to these buses, which indicates the unity of the transfer to the top digit. The outlet of the AND circuit feeds the input of the NO circuit, which, if receiving a positive pulse creates a negative pulse at the outlet.

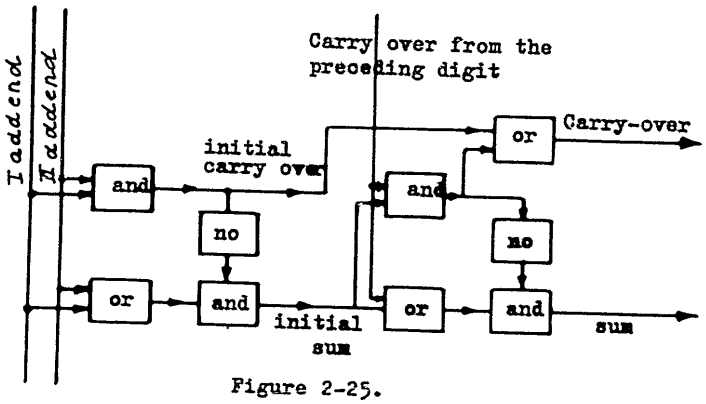


Figure 2-25.

A positive pulse of the OR circuit enters the inputs of the second AND circuit while a negative pulse comes from the NO circuit. Therefore no pulse will be fed to the bus of the sum. The network shown by Figure 2-25 is called single-digit "half adder" since it provides only the addition of two numbers out of three (the third is the figure of the carry-over), which participate in the digital summation.

The full digital summation with consideration of the carry-over figure may be performed in two cycles of the circuit formed by two half-adders and a separation circuit of the OR type (Figure 2-25). It is necessary to point out that this circuit works in two cycles. Pulses, corresponding to the binary figures in the digits of the numbers to be added, are fed to the input of the circuit during the first cycle and the first half-adder develops signals of sum and carry-over according to the rules of the first four lines of Table 2-2*. Then in the second cycle the second half-adder performs the summation of the figures of the initial sum and the carry-over from the preceding lower digit.

---

* This sum and carry-over is not final (it is called initial), since it is obtained without considering the carry-over figures from the preceding lower digit. We notice that the initial sums and the carry-over may not be equal to one.

STAT

Since only two figures are added simultaneously in the circuit shown by Figure 2-25, it is called a summator with two inputs. It is possible to build by logical circuits a single-digit summator with which all three figures are added simultaneously (first and second addends and carry-over). Such a device is called a summator with three inputs. Its functional circuit is shown by Figure 2-26. The circuit deciphers and transmits to the outlet the required signals of the sum and the carry-over of any of eight combinations of input signals as indicated by Table 2-2.

If, for example, to all three input buses signals of the one are fed (corresponds to the eighth line of Table 2-2), then at the outlet of the coincidence circuits 1, 2, 3, 4, and at the separation of circuit 6, there will be also signals of a 1.

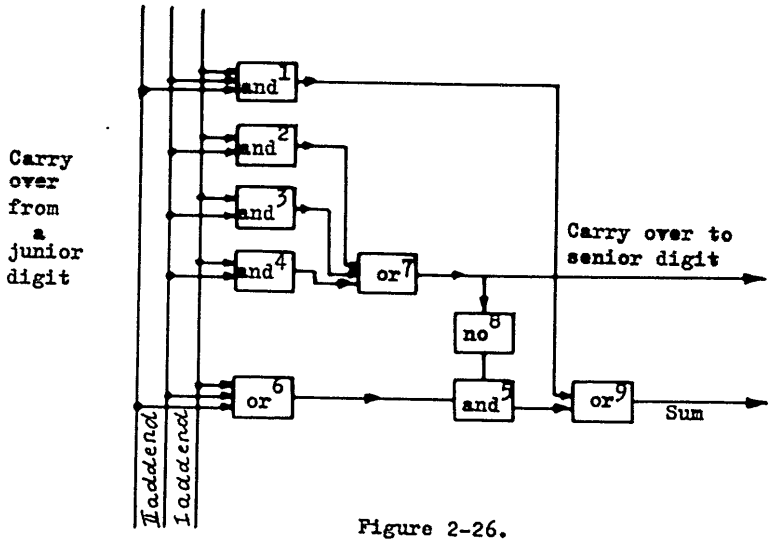


Figure 2-26.

The signal of the 1 will also appear at the outlet of the separation circuit 7, i.e. the signal of the carry-over of the 1 will appear in the higher digit. The signal 1 will be at the input of the NO circuit while the signal 0 will be at its output. Therefore, at the outlet of the coincidence of circuit 5 there will be also the signal 0. However, the signal of a 1 will enter the input of the separation circuit 9 coming from the outlet of the coincidence circuit 1, and it will go to the outlet and at the terminal "Sum" there will be the signal of a "one. In this way it is possible to track the operation network also with other combinations of input signals.

Up to this time, one-digit summators and the adding operation were considered in one digit. Now we will discuss the summation of number codes containing a certain amount of digits.

STAT

We distinguish between parallel and serial action adders.

In serial action adders, the adding operation is performed by turns, separately for each digit beginning with the lowest one. Such a summing device can utilize only one single-digit adder, to whose inputs pulses are fed alternatingly within certain time intervals according to binary figures of the digits to be added. We may say that adding process is time-based in a serial adder.

In a parallel action adder, the adding process is performed simultaneously for all digits of a number. Naturally, in this case it is necessary to have as many solving and control circuits as there are digits in the numbers to be added. In a parallel adder the adding process is performed by a group of solving and control circuits, or in other words, in space. It is necessary to take into consideration that even in parallel adders, the time-based serial carry-over of the one from the lower to the higher digits cannot be completely avoided.

The time required for performing the adding operation is considerably smaller with a parallel adder than in a serial adder. Consequently, parallel adders are always used when it is necessary to obtain a high speed of calculations although this is connected with an increase of the equipment volume.

The functioning of a serial adder is explained by the block diagram shown by Figure 2-27 (L 13). The work cycles are created by a synchronizing pulse generator.

The digits (pulses corresponding to them) of numbers to be added are fed consecutively one after the other within certain time intervals thru the valves K_1 and K_2, controlled by synchronizing (timing) pulses q to a single-digit adder with three inputs. Furthermore, the signal of the carry-over (a zero or a one) from the preceding lower digit is fed to the third input. This signal was developed in the preceding operation cycle of the circuit when the summing of the preceding lower digit was performed and which was stored (delayed) corresponding to the carry-over pulse by the delay circuit (Z on Figure 2-27) and sent out to the input of the circuit at the required moment.

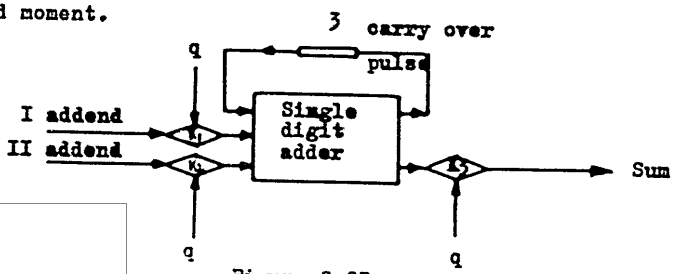


Figure 2-27.

STAT

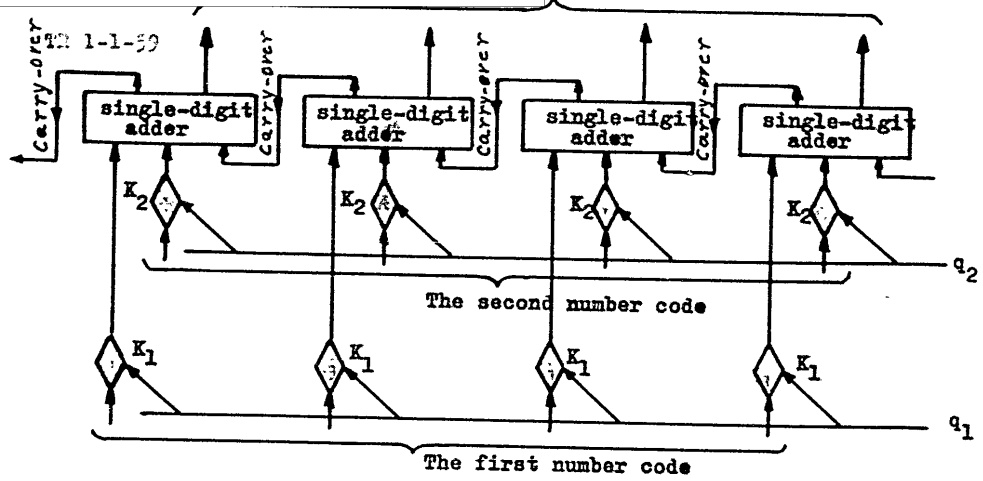


Figure 2-28.

The signal of the sum is transmitted to the external circuit of valve K_3, which is also controlled by synchronizing pulses.

Figure 2-28 explains the principle of functioning of the parallel adder consisting of single-digit adders with three inputs. The codes of the figures corresponding to the digits of the first and the second number and the carry-over signal from the preceding lower digit enter the inputs of each single-digit adder.

The codes of the first and second numbers are fed to the adder thru the valves K_1 and K_2, opened by pulses q_1 and q_2. The summing element has two outlets in each digit, from which the signal determining the figure (1 or 0) of the corresponding sum digit, and the signal of the carry-over to the following higher digit, are taken.

In the circuit under consideration, the time required for performing the adding operation depends on the time of completing the process in the single-digit adder itself and on the number of digits. Actually, in case the carry-over spreads from the lowest to the highest digit, the time of the adding operation will be determined by the sum of the time for completing the processes in all digits of the adder.

Until now we considered serial and parallel adders, whose output signals were determined by the codes of the two numbers to be added, which were fed to the input simultaneously. Such devices are called coincidence type adders. They are based on the utilization of logical circuits.

Together with the coincidence type adders, counter type adders are widely used in computers. In these devices, the codes of the addends enter the adder separately and at different moments of time.

STAT

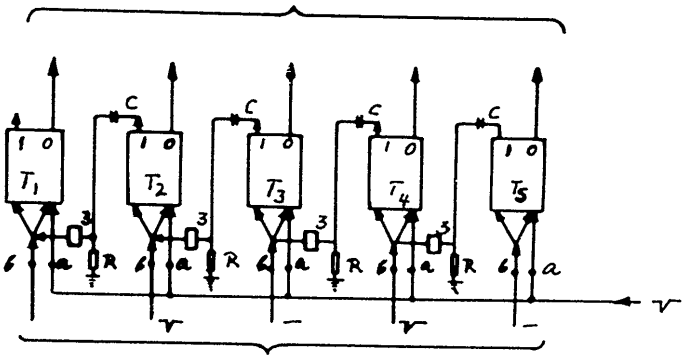


Figure 2-29.

When feeding a new number to the input, the sum of this number is formed with that number which was previously in the adder. A counter type adder may add successively any amount of numbers entering its input. The sum obtained in this way is stored in the adder even after the code signals of the addend are discontinued.

Trigger counting cells are used in counter type adders. Figure 2-29 explains the principle of functioning of a parallel counter type adder. The network consists of five triggers, T_1, T_2, T_3, T_4, and T_5. The parallel pulse code of a number (negative pulse means a 1, absence of pulses means a 0) is fed to the counting inputs (inputs v). Each subsequent negative pulse being fed to the counting input of the trigger, will bring the latter into the opposite state. In the adder, the feeding of each subsequent negative pulse to the input of some trigger means an increase of the sum by a 1 in the respective digit.

Let us assume that the trigger of some digit is in state 1. If a negative pulse enters its input, i.e. a "one" is added in this digit, then the trigger must change to state "zero", but, in addition, the carry-over pulse for the "one" to the neighboring higher digit must appear. In this way, each time when the state of some trigger changes from "one" to "zero", an additional negative pulse (carry-over pulse) must appear at the input of the neighboring higher digit. The transition of the trigger from state 1 to state 0 is controlled by the differential circuit RC connected to the trigger outlets for the one.

When a trigger changes from state 1 to state 0 the voltage at its outlet for the "one" will drop and the differential circuit develops a negative pulse (carry-over pulse) which enters the counting input of the trigger for the next higher digit, which corresponds to an addition of a "one" to this digit. The negative carry-over pulse

STAT

changes the trigger from state 0 to state 1 and the voltage at its output for the "one" increases and a positive pulse origirates at the outlet of the differential circuit which does not change the state of the neighboring trigger. In other words, in this case the carry-over pulse does not arise.

It is necessary that the carry pulses enter the inputs of the triggers after the transient process in them is completed which is caused by the number code pulses fed to the input of the adder. In the opposite case, when the carry pulse arrives at the trigger input at the moment when the transient process caused by the code pulse is not yet finished, it will not have any influence on the state of the trigger.

For eliminating losses of carry pulses, the delay circuit Z (Figure 2-29) is included, delaying the carry pulses for the time required for finishing the transient process in the trigger.

The adder is set at zero by feeding a negative pulse to the zero inputs of all triggers. If now the pulse code of a number is fed to the counting inputs of the triggers, then the adder will add this number to the zero.

For example, when feeding to the input the code of the number 01010, triggers T_1, T_3, T_5 remain in state 0, while triggers T_2 and T_4 change to state 1, since negative pulses are fed to their counting inputs. Now, the state of the adder triggers will represent the number 01010.

If, thereafter, the code of number 01011 is fed to the adder input, then this number is added to the number 01010, i.e. a sum is obtained

$$\begin{array}{r} 1\ 1 \\ 01010 \\ + \\ 01011 \\ \hline 10101 \end{array}$$

Actually, the negative pulses of the code of number 01011 change triggers T_2, T_4, and T_5 to the opposite state. Thereby, trigger T_5 appears in state 1. Triggers T_2 and T_4 change to state 0, but negative carry pulses arise which change triggers T_1 and T_3 to state 1. As a result the adder will show the sum 10101.

If the carry pulse enters the input of a trigger being in state 1, then the latter changes to state 0 and thereby a new carry pulse originates in the subsequent higher digit. Under certain conditions the unity of the carry will pass thru the entire trigger register from the lowest to the highest digit. It is easily noticed that the time required for performing the adding operation will be determined by the total delay time of the carry pulses, if the time required for

STAT

the transient processes in the triggers is neglected. If in each delay circuit the carry pulse is delayed by the time t_z, then in the worse case (when the 1 of the carry passes thru the entire adder) the duration of the operation will be

$$T_{o.n} = t_z (n - 1)$$

whereby n is the number of adder digits.

For reducing the time required for the adding operation, the consecutive transfer of the carry within the time unit from one digit to the other is eliminated and a circuit is used in which the carry is performed to all digits simultaneously (L 17). This is achieved by dividing the adding process itself into two cycles: 1) the adder performs the addition by digits (without considering the carry) and develops the carry signals; 2) to the digit sum in the adder the carries are added simultaneously and a final sum is formed.

We will explain this by a practical example. We will add the two binary numbers:

10011101100011101	I addend
00100100011110001	II addend
10111001111101100	digit sum
00001000000100010	carry
11000010000001110	final sum

Considering the formation of the final sum, we notice that the addition of the carry unity to the digit of the digit sum causes a widespread carry left of the digit, from which it originated, to the first left digit with the figure "zero". Thereby in all these digits, including the first left of the "zero" digit, the figures are changed to the reversed "one"s.

As soon as a digit sum and carry pulses were obtained in the first adding cycle, it is possible to separate by means of logical circuits those digit groups in which the figure code must be reversed and the carry pulse must be transmitted simultaneously to the counting inputs of the triggers.

Figure 2-30 represents a block diagram, explaining the principle of operation of a parallel counting adder with simultaneous carry in all digits. When the synchronized pulse q_1 is fed to valves K_1, the pulse code of the number enters the counting inputs of the adder triggers. Thereby a digit-by-digit addition occurs in each trigger cell of the code already in the adder with the newly introduced code. Simultaneously, by means of the differentiating circuits D, the carry pulses are formed in the corresponding digits. These pulses are fed

STAT

to valves K_2. Since positive signals are required for opening the valves, the differentiating cells are connected to the "zero" outlets of the triggers and the positive pulse at the input of valves K_2 corresponds to the carry pulse. Within a certain time interval, adequate for finishing the transient processes, caused by the pulse code of the numbers, the synchronizing pulse q_2 is transmitted, opening those of the valves K_2, which received on the other grid comparatively wide pulses from the differentiating circuits. The carry pulses pass thru the opened valves K_2 to the trigger input of the neighboring higher digit and, in addition pass from right to left on the valve circuit AA' and simultaneously enter the inputs of the corresponding trigger cells which are connected to this circuit, changing them to the reverse state.

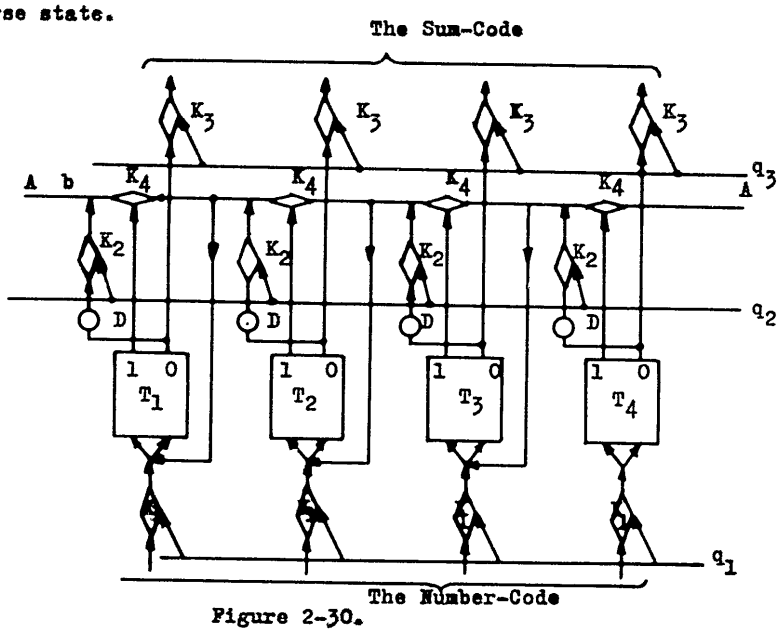


Figure 2-30.

The passage of carry pulses along circuit AA' is controlled by valves K_4. Each of these valves passes the carry pulse to the left, if the respective trigger is in state 1; it is not passed if the trigger is in state 0. For this purpose, the voltage levels from the trigger outlets for the 1 are fed to the control inputs of valves K_4. In this way, valves K_2 and K_4 form logical networks, separating those digit groups in which the codes must be changed to reverse carry pulses. The code of the sum is transmitted from the adder thru valves K_3, opened by the synchronizing pulse q_3. When reversing the trigger cells, new carry pulses arise which already must not have an influence on the state of the circuits. For this purpose, valves K_2 block the passage of new carry pulses.

STAT

In the circuit under consideration, the time required for the

"passage of the carry unity" thru the adder is basically by the time of the transient processes in the series circuit of the valves K_4.

The coincidence adder, in combination with the trigger register which "remembers" the number codes, has the property of a counter, i.e. it can add new addends to a previously established sum and will store the result obtained. As an example, we will consider the parallel coincidence adder with a trigger register which is used in the M-3 computer (L. 3 and 4).

In this network, as well as in the aforementioned adder, the adding process is performed in two cycles. The contents of the cycles, however, are different. During the first cycle the carry unities are determined which are stored in a special trigger register - the memory for carry unities. During the second cycle the digit-by-digit addition is performed (without formation of new carry unities) of the figures of the first and second addend and the carry figures obtained during the first cycle.

Let us examine the addition of two binary numbers:

+ 01011011	first addend
00101011	second addend
10000110	sum

This operation may be divided into two parts according to the aforementioned work cycles of the adder.

First cycle: The carry unities are determined:

01011011	first addend
00101011	second addend
11110110	carry

whereby here, in difference to the numerical example considered above in connection with Figure 2-29, not only the primary carry unities are shown in the line "carry" which appear when there are "one"s in the analogous digits of the first and second addends, but also all carry unities originating during the addition of the two numbers under consideration.

Second cycle: Digit-by-digit addition (without forming new carries)of the two addends and carry unities

01011011	First addend
11110110	carry
00101011	Second addend
10000110	sum

STAT

Taking into consideration the last two tables, it is possible to establish the following logical rules for producing a carry and a sum.

1. A carry unity appears in a given digit, if two "one"s were located in the preceding digits of the first and the second addend and of the carry.

2. For forming the sum, the binary figures of each digit of the first addend must be reversed if there are different figures in this cycle of the second addend and the carry, and it must remain unchanged if the numbers are identical.

The first rule is performed in the first work cycle of the adder and the other one in the second cycle. The operation in both cycles is achieved by means of logical circuits.

In the adder under consideration, the totalizer is combined with the trigger memories of the first and second addends(1), whereby the trigger register of the first addend functions simultaneously as a counter(2). Furthermore there is a special trigger register for remembering the carry unities.

Figure 2-31 shows the block diagram of the circuit which controls the memory register of the carry unities according to rule 1. The diagram shows to a smaller extent to the adding operation the triggers and control tubes of the n-th and the lower n+1 digit. The carry unities are formed by means of simple circuits, consisting of coincidence and separation circuits as well as special tube circuits for shaping pulses. A double triode is used for this purpose in each digit. A pulse transformer is connected to the anode of the right tube (tube L_1). The negative pulse from the anode circuit of tube L_1 is fed to the "one" input of the trigger of the respective digit in the memory register of the carry unities. The positive pulse from the secondary transformer winding enters the grid of the left tube (tube L_2). The left tube L_2 of the lower n+1 digit is connected parallel with tube L_1 of the n-th digit to one and the same anode load. The coincidence circuits (with three inputs) are connected to the grids of tube L_1, while the separation circuits are connected to the grids of tube L_2. Furthermore, as we will see in the future, the tube L_2 itself works in the same way as a coincidence circuit.

(1) Trigger memory registers for numbers are required also for the counter adders of the type shown by Figure 2-30, but there these registers are functionally connected to a smaller extent to the adding operation.

(2) In the command code of computer M-3 (see Appendix 1) which is further explained by Figure 2-31 and 2-32, the register of the first addend and the sum and the register of the second addend are called register of the second number and the sum and register of the first number respectively.

STAT

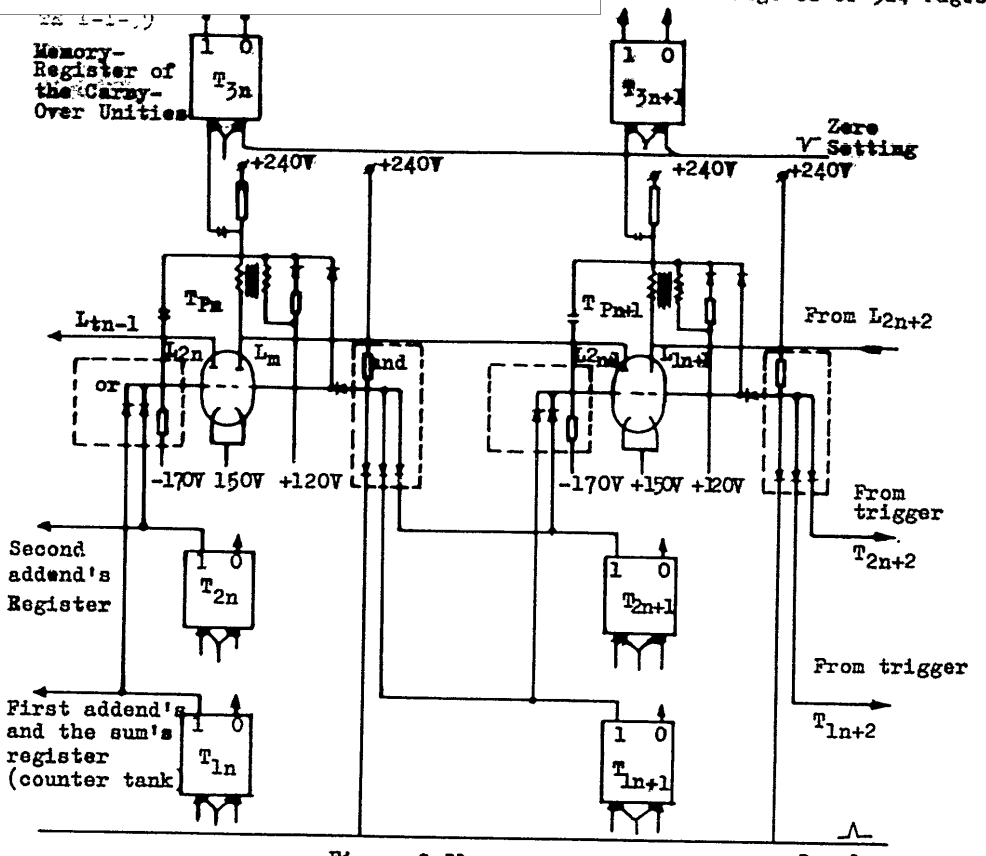


Figure 2-31.

Preliminarily the memory register of the carry unities is set at zero by a negative pulse. The formation of the carry unities in the carry register begins after feeding the positive pulse "Run 1". This signal is fed simultaneously to the respective inputs of the coincidence circuits in all digits of the adder.

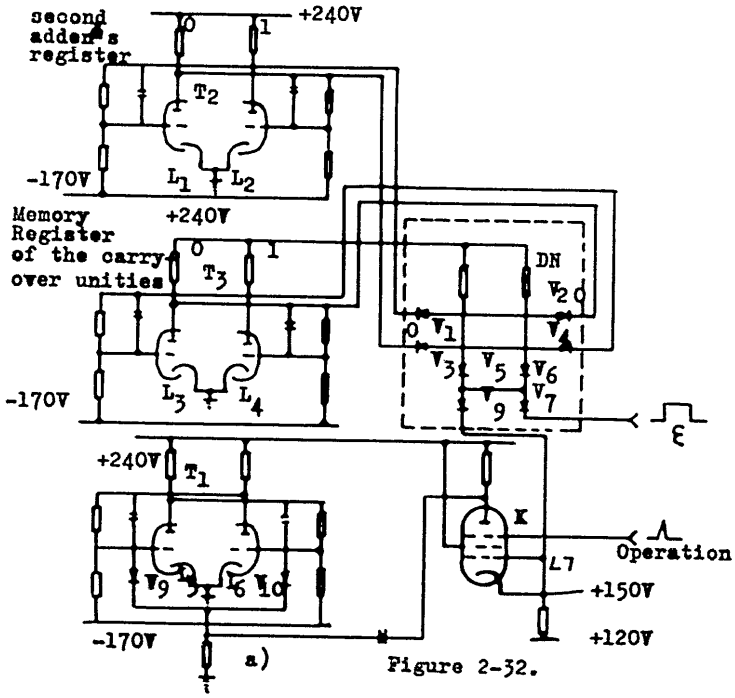
Let us assume, for example, that triggers T_1 and T_2 of the n+1 digit are in the state of a "one" and that there is no carry from the n+2 digit. Then at the outlet of the AND circuit, which is connected to the grid circuit of tube L_{1n}, a high voltage level will appear. Tube L_{1n} is opened and thereby a negative pulse is generated which puts to state "one" the trigger T_{3n} of the memory register of the carry unity. Simultaneously a positive pulse enters the grid of tube L_{2n} from the secondary coil of transformer Tr_n and the potential of the grid of tube L_{2n} will be adequate to open the tube L_{2n} under the influence of this positive pulse, if even one of the triggers T_{1n} and T_{2n} of the registers of the first and second numbers are in state one. Tube L_{2n} is connected parallel to tube L_{1n+1} to a common anode load (transformer circuit Tr_{n-1}) and the negative pulse generated in this circuit puts into state one the trigger of the next higher n-1 digit of the carry unity memory register (trigger T_{3n-1}).

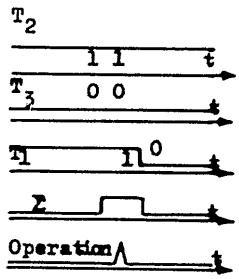
The carry process, originating in the n-1 digit in the explained manner will spread to the side of the higher digit until it reaches such a digit in which both triggers T_{1i} and T_{2i} are in "zero" position.

Actually, if both triggers T_{1i} and T_{2i} of some digit are in "zero" state and the carry pulse is generated in the anode circuit of tube L_{1i} setting trigger T_{3i} to state "one", the positive pulse generated thereby cannot pass thru tube L_{2i} since from both triggers T_{1i} and T_{2i} low voltage levels are fed. Thus, the spreading of the carry unity in the group of digits under consideration is interrupted.

It is important to remark that the formation of the carry unity may begin at one time at several digits and may spread simultaneously from right to left. The time required for setting the carry unities in the trigger register T_z is determined by the duration of transient processes in the circuits of tubes L_1 and L_2. Under the most unfavorable condition, when the carry unity "runs" from the lowest to the highest digit, the time required for forming the unities will be equal to the sum of delay times in the circuits of tubes L_1 and L_2 of all digits.

During the second cycle the logical circuits set the triggers of the sum register (counter) in all digits to the state which is determined by rule 2.





b)

Figure 2-32.

We will now consider the operation of the logical circuits during the second work cycle of the adder. Figure 2-32a represents such a circuit arrangement for one digit of an adder. The logical non-conformity circuit DN checks the non-coincidence of the state of triggers of the second addend register and the carry unity memory register. The outlet of the non-conformity circuit feeds the first grid of valve K. An auxiliary, wide, positive signal Σ is fed to the control input of the non-conformity circuit. If at this time the triggers T_2 and T_3 are in the opposite states, then a high voltage level is fed from the non-conformity circuit to the first grid of the valve while a low voltage level appears in the other case. Then, still before signal Σ fades out, the positive signal "operation" is fed to the third grid of the valve and digit-by-digit summation is performed (second work cycle of the adder).

If triggers T_2 and T_3 are in different positions, then a negative pulse from the valve will enter the counting input of trigger T_1 and the latter is changed to the opposite state. If triggers T_2 and T_3 are in the same state, the valve will be blocked by a low voltage level at the first grid and the state of the trigger does not change.

Apparently, coincidence parallel adders have a greater future, since the logical circuits on which they are based provide more possibilities for building high-speed arithmetic units.

2-7. Memory units.

Memory units of computers may be built with trigger circuits, but in this case the number of tubes would be increased to a considerable extent. The magnetic drum memory is a simpler and more reliable device. This device functions analogously to a sound recorder. The binary numbers are recorded on the surface of a rotating drum which is covered by a layer of ferro-magnetic material. The recording and reading is performed by "heads" - miniature electromagnets placed close to the surface of the drum with a small gap (Figure 2-33a).

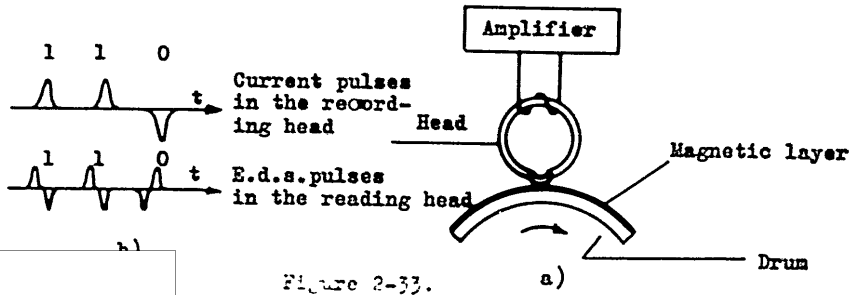


Figure 2-33.

STAT

When a current pulse passes thru the coil of the recording head, a magnetic flux is generated in the gap. A part of this flux is short-circuited by the section of the ferro-magnetic layer of the drum passing at this time underneath the head, whereby the layer is magnetized thus a small magnet is formed on the ferromagnetic layer. The polarity of such a magnet is determined by the polarity of the current pulse in the winding of the recorder head. For recording "zero"s and "one"s on such a drum, current pulses of reversed polarity are used. In this case, the polarity of the magnetized sections of the drum surface determines the value of the recorded numbers (1 or 0). The reading of the information recorded on the surface of the drum is performed by so-called "reading heads". The recording and reading heads are of identical design and in a number of devices one and the same head is used simultaneously for recording and reading. When a magnetized section of the drum surface passes, a voltage is induced in the reading head, the polarity (phase) of which indicates whether in the given section a one or a zero were recorded (Figure 2-33b).

In magnetic memories also another method of recording is used, whereby the ferromagnetic layer is preliminarily magnetized up to the saturation point in one direction. The magnetizing effect of current pulses during recording of, for example "zero"s, is directed in the same direction. Pulses of "one"s, having reversed polarity, magnetize corresponding sections to the saturation point in the opposite direction. With this recording method a voltage is induced in the reading heads only when reading a 1. Figure 2-34 shows corresponding induction curves B which were produced by the magnetized layer in the reading head and the e.m.f. in its coil.

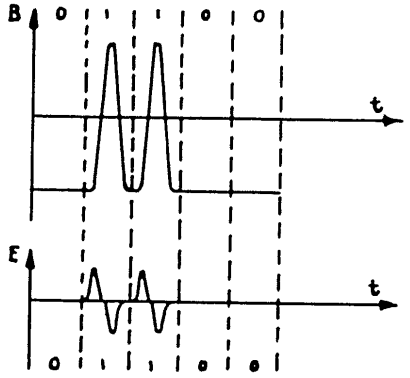


Figure 2-34.

There are several systems for recording the numbers on the surface of a drum. In one of these systems, the cells into which the binary numbers are recorded are located along the generatrix of the drum surface. The values of all binary numbers are recorded

STAT

simultaneously. With such a system it is necessary to have as many recording and reading heads as there are binary digits of a number to be recorded. The heads are located parallel to the generatrix of the drum. The numbers 1 or 0 are recorded on each track passing underneath the head, according to the digits of all numbers located in the memory unit.

Usually, the volume of a memory with a magnetic drum comprises 1024 ÷ 4096 numbers or commands. However, magnetic drums are built also for a considerably larger memory volume.

For finding the required number of a cell when recording or reading markers are used, which are permanently recorded on the drum by magnetizing or by mechanical means and which are located on the so-called marker track. The "zero" marker signs are applied at a separate track; also one of the markers of the marker track may serve as "zero" mark, but in this case it is performed differently from the others, for example by means of magnetizing in the opposite direction.

Figure 2-35 (L. 5) shows one of the versions for the control networks built with a magnetic drum. The markers create pulses in the marker head, which, after passing the amplifier and shaping block UF, enter the input of the trigger counter of marker pulses. The number of digits of the counter depends on the memory volume. For example, at a memory volume of 2048 numbers, the counter will have 11 digits. With each rotation of the drum, the impulse of the zero marker sets the counter to "zero".

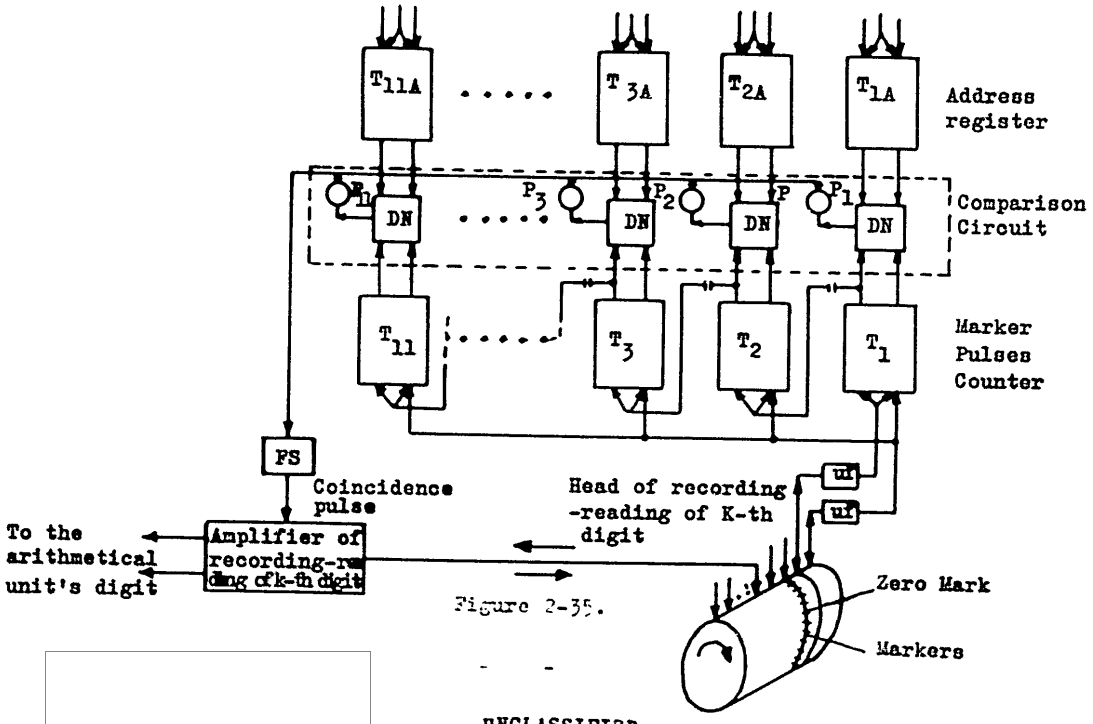


Figure 2-35.

STAT

The number (address) of the memory cell from which the number is to be read, or into which the number is to be recorded enters the trigger "address register"; by means of the "comparison circuit", the state of the trigger address register is compared with the state of the triggers of the marker pulse counter. The coincidence of the state of the triggers of these registers shows that the required memory cell reached the head. In this case, the comparison circuit shapes the "coihcidence pulse" and in accordance with the operation to be performed a reading or recording of the number is done.

The comparison circuit consists of the non-conformity decoders DN, to whose inputs voltage levels are fed from the anodes of the corresponding triggers of the address register. The outlets of the decoders are connected with the cathode followers which have one common load. At the moment when all triggers of the address register and the counter are in identical states, a low voltage level is established at the outlets of all decoders and consequently at the load resistances of the cathode followers and in the FS block a coincidence pulse is shaped.

The counter must be in 0 state at the moment when the signal of the 0 mark ("zero" marker) is read, provided the counter works properly. The latter is controlled by a special circuit (which is not shown on Figure 2-35) blocking the work of the comparison circuit in case of malfunction of the counter and switches on an acoustic signal.

The magnetic drum memory unit has one essential disadvantage, the relatively long time required for selecting one number from the memory. This time is determined from the moment the reading order is fed to the reading heads of the corresponding memory cell. On the average, this time is identical to one half-rotation of the drum. It is necessary to remember that the performance of one operation requires as a rule several references to the memory. The time required for the selection of one number from a magnetic drum memory amounts to 5-10 milli-seconds. As a result, computers with a magnetic drum memory will work only with a speed which does not exceed tens or hundreds of operations per second. Thereby the high-speed possibilities created by electronic arithmetic and control units are not used which otherwise might perform 1000 operations per second.

Increasing the work speed of a computer does not only increase its productivity but also reduces the possibility of computer failures during the process of solving a certain task.

Ultrasound delay lines, electronic ray tubes and other devices are used as high speed memory units. If one may record on the screen

[redacted] STAT

of an electron ray tube, for example, 32 x 32 = 1024 the figures 0 or 1, then for recording of 1014 binary 32-digit numbers 32-electron ray tubes must be available. The time required for reading or recording memory in an electron ray tube amounts to only 10 microseconds.

Until recently high-speed memory units were built chiefly with electron-ray tubes. Presently high-speed memory units find a more and more widespread application in which ferrite cores are used. For this purpose miniature toroidal cores are used having an outside diameter of 2-3 mm or less. The ferrite cores have almost a rectangular hysteresis loop, and with some idealization one may say that the magnetizing curve corresponds to Figure 2-36.

If a series of pulses with changing polarity is fed to the coil of the core creating thereby a magnetic field exceeding H_c (Figure 2-36), then each incoming pulse changes the core from one magnetic state into the opposite one. Owing to the rectangular hysteresis loop the core will keep reliably the state of magnetization transferred to it. One magnetic state may be designated as 1 and the opposite as 0, and the core may be used as an element of a memory unit.

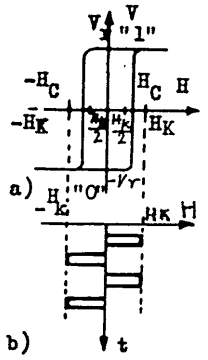


Figure 2-36.

Figure 2-37 shows the principal circuit of a memory unit in the form of a plane matrix, formed by a lattice of vertical and horizontal wire buses. The toroidal cores are located at the intersections, whereby one vertical and one horizontal bus passes thru each core. One common coil which contacts all cores serves for reading.

The control of recording and reading operations in the device under consideration is performed by means of coincidence of signals. Let us assume, it is required to record signal "one" in core a. In this case signals $+I_k/2$ are fed coincidentally to the horizontal and vertical buses.

STAT

These signals are added in the core forming the total signal I_k. The magnitude of signal I_k is selected in such a way that the requirement (Figure 2-36) is met:

$$H_k > H_c \quad \text{and} \quad \frac{H_k}{2} < H_c,$$

whereby H_k and $\frac{H_k}{2}$ - magnetic field intensity of the core field corresponding to signals I_k and $I_k/2$.

If the core a was in state 0, then, after feeding the signals $I_k/2$, the added signals change core a to state 1. If core a was in state 1 previously, then after the feeding of the signals its state will not change. The state of all other cores located on the selected buses remains unchanged since the signal $I_k/2$ is not adequate for changing the magnetization direction of these cores. The signal 0 is recorded by the negative pulse $-I_k/2$.

For reading the signal recorded in the core a, the signals $-I_k/2$ are transmitted to the same buses, but having an opposite polarity of the signals for recording a 1. If a 1 was registered in core a, then the reading signals would be added and core a would be re-magnetized whereby a pulse would be transmitted to the reading coil. If a zero was recorded in core a, the reading signals would not change its state and there would be no pulse in the reading coil.

When reading a 1, the state of core a is changed to 0. In order to restore the information in the memory automatically after reading, special circuits are used. For example, the reading may be performed by two-polar pulses ($-I_k/2$, $+I_k/2$), whereby pulses $+I_k/2$ (restoring to state 1) are fed to the buses only when pulse 1 originates in the reading coil.

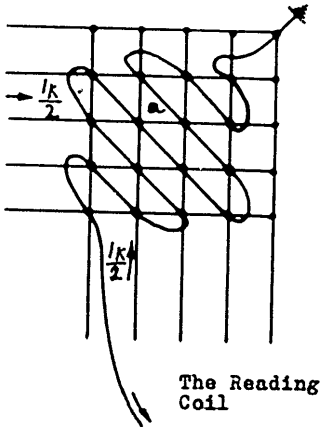


Figure 2-37.

STAT

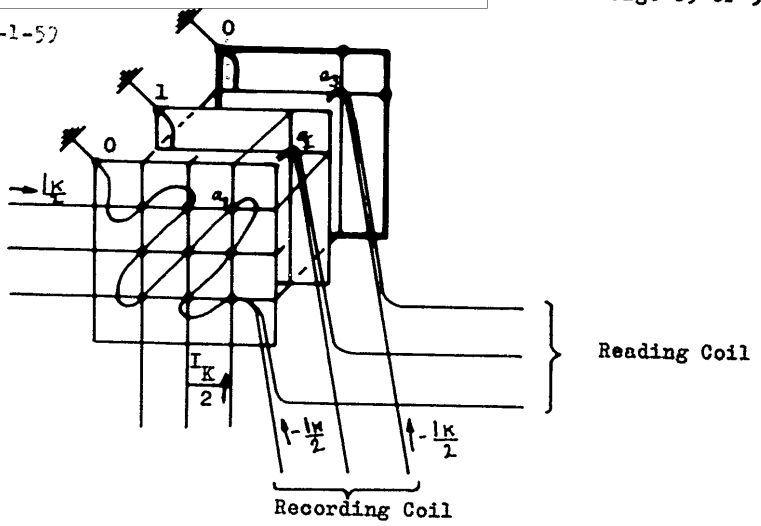


Figure 2-38.

Figure 2-38 shows one of the versions of a "three-dimensional matrix memory" circuit with coincidence (parallel) recording and reading of figures. The three-dimensional matrix consists of identical plane matrices the number of which is equal to the amount of binary digits of the numbers to be stored in the memory. The number of cores in each plane matrix is equal to the number of addresses of the memory, i.e. the amount of numbers to be stored in the memory. Correspondingly located horizontal and vertical buses of the plane matrices are connected in-series with each other. Each plane matrix has a recording coil and a reading coil, which are connected to all its cores (Figure 2-38).

The section of the required address in the three-dimensional matrix is performed by a coincidence circuit with simultaneous feeding to the corresponding horizontal and vertical buses of signals $+ I_k/2$ or $-I_k/2$. Thereby a simultaneous selection is achieved for recording or reading of identically located cores in all plane matrices, for example, cores a_1, a_2, a_3,....., in which the figures of the respective digits of the binary numbers were stored.

For reading, pulses $-I_k/2$ are fed to the respective buses, whereby pulses are generated in the reading coils of the plane matrices, if unities were stored in the "read" cores and whereby no pulses are generated if "zero"s were stored.

After the reading follows the cycle for restoring the information.

The recording is preceded by a reading cycle without subsequent restoring, and as a result, all cores of the selected address are in state 0. Then pulses $I_k/2$ are fed to the respective buses, whereby, if a "zero" is to be recorded into the cores of the given digits pulse $-I_k/2$ is fed, and no pulse at all if a 1 is to be recorded.

STAT

STAT

The control of the matrix memory is achieved by special matrix switches, performing the task of converting the address of the number (given as certain combinations of voltage levels of the address register triggers) to pulses on the proper vertical and horizontal buses of the matrix memory.

The ferrite core memories have a number of advantages compared with electron-ray tube memory units, since they permit to improve the reliability of the work of the memory, to reduce the quantity and volume of electronic equipment. In distinction to memory units with magnetic drums, the ferrite core memory units do not use any moving parts. This permits to shorten the time required for the selection of a number from memory to several (6-8) microseconds.

2-8. Input and output devices.

Devices consisting of photoelements are used for putting the program into a computer, whereby the program is recorded on perforated tape. When the holes of the perforated tape pass the photoelements, then corresponding current pulse combinations are obtained at the outlet terminals of the input device, which enter the computer.

Figure 2-39a shows an example of a perforation system for figures, the signs of plus and minus and of service operations on a five-position tape used in computer M-3. The figures of the decimal and octal numbers are perforated in binary-coded form. The holes of the extreme left column indicate whether a number or the sign of a number was recorded in a particular line.

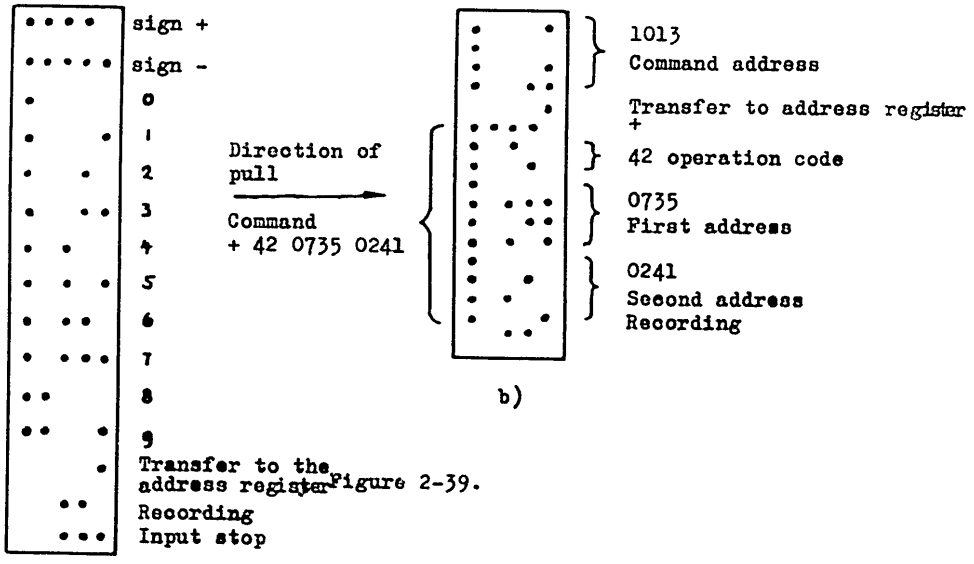


Figure 2-39.

STAT

The codes of the auxiliary operations (transfer to the address register, recording, end of input, etc.) do not have holes in the extreme left column. In this way each figure of a decimal or octal number, sign or service operation is shown by one line of holes in the tape.

Figure 2-39b shows an example for the perforation of command +42 0735 0241*, which is to be placed into memory cell 1013. First, figure after figure of the cell number is perforated (1013), where the command will go. Then the auxiliary operation "Transfer to address register" is perforated. After this, the sign + is perforated, and followed by all the figures and commands and finally by the service operation "Recording". Thereafter such a cycle is repeated for the next command, and so on.

When the tape is processed, the input apparatus reads and transmits to the computer the codes line by line. Initially, the computer receives the address of the cell into which the command is to be placed. The command is recorded in the memory, after the service operation code "Record" has been transmitted, in the cell whose number was indicated in the address register (1013).

Numbers are fed to the computer analogously(1). The introduction is possible without punching addresses on the tape according to which the numbers are to be placed in the memory. In this case the placing of the numbers is performed by the computer by means of an auxiliary program.

For automatic printing of the data entering from the computer, electromagnetic printing devices may be used. For speeding up the output of data, they are recorded on magnetic tape or on photographic film instead of a letter-printing electromagnetic device.

---

* In accordance with the command code of computer M-3 (see Appendix) this command means: The contents of cell 0241 is to divide by the contents of cell 0735. The quotient is to be recorded in cell 0241 of the memory unit and is to be printed. At the start of the command the sign + is perforated thus the command will occupy the same number of digits as the number.

(1) The control panel of computer M-3 has a switch for introducing octal and decimal numbers.

2-9. Control units.

The individual units of a computer - memory, arithmetic unit, input and output devices - have their own control units. The common control unit of the computer provides the interaction of the individual units thus the computing process is performed automatically according to a previously established program which is located in the memory unit.

The control unit generates in a pre-determined sequence control pulses which actuate the corresponding units of the computer for performing the following elementary operations:

1. Reading the code of a series command from the memory unit and transmitting it to the proper units of the computer.

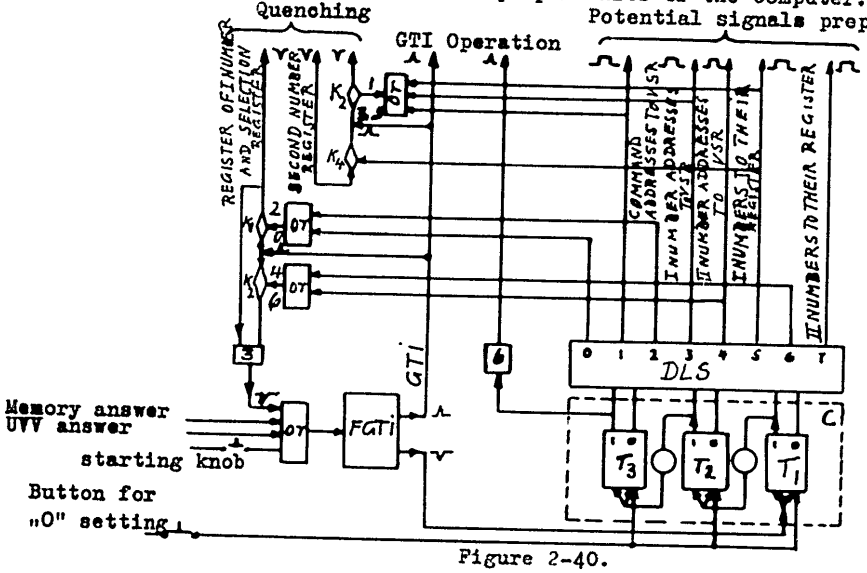


Figure 2-40.

2. Reading of number codes from the memory unit and transmitting them to the register of the arithmetic unit.

3. Performing the ope ation indicated by the command code (arithmetic operation, transferring one number from one cell of the memory to another one, and so on).

4. Transferring the result obtained to the memory unit and to the output printing device (if this is provided by the command).

5. Forming the address of the next command which is to be performed by the computer.

Two control systems are used in computers: the asynchronous and the synchronous systems. With the asynchronous system, the control is achieved after the termination of the elementary operations into

STAT

which the process of performing commands is divided (reading of commands, reading of numbers, arithmetic operations, recording of numbers and others). The computer changes to the performance of the next command only after having obtained the signal of the performance of the preceding operation. With the synchronous system there is no control after the termination of individual operations. For the performance of each elementary operation a certain time is allocated, corresponding to one or several cycles (synchronizing pulses), and then the computer continues with the next operation.

To consider the different principles of control unit design of computers is beyond the scope of this book. We will limit ourselves to a short description of the principle of control unit functioning of the M-3 computer, which is built according to the asynchronous system (L. 3 and 5).

The central control unit of the M-3 computer contains also two address registers besides a pulse generating block for controlling the function of the individual units of the computer. These registers are: a) a selection register (SR) to which the number of the memory unit cell is fed, from which a reading is to be taken or into which the result is to be written; b) the start register (PR), in which the address of the subsequent command is formed.

The operation code is transmitted to the operation block (BO) of the arithmetic unit control device (device of local control) which upon reception of the code from the central control unit decodes this code and performs the control of the operations to be performed by the arithmetic unit (addition, multiplication, etc.).

Figure 2-40 shows a simplified block diagram of the central control unit (the address registers are not shown) and Figure 2-41 shows its time graph. The work cycle of the central control unit for performing a full two-address command consists of eight steps which are determined by the states of the three-digit binary counter C, consisting of triggers T_1, T_2, T_3. Prior to the start of the work cycle the trigger is set at 0, and then with each step its state changes by one unity. Upon completion of the work cycle the counter is again set at "zero".

The pulses of the answers concerning the completion of elementary operations: reading and recording - from the memory device, printing - from the input and output unit UVV, and also from the circuit of the central control unit, enter the input of the PGTI network, which forms from them the basic positive step control pulses GTI (altogether seven pulses in one work cycle) and with some delay, the negative pulses enter the input of the counter.

STAT

STAT

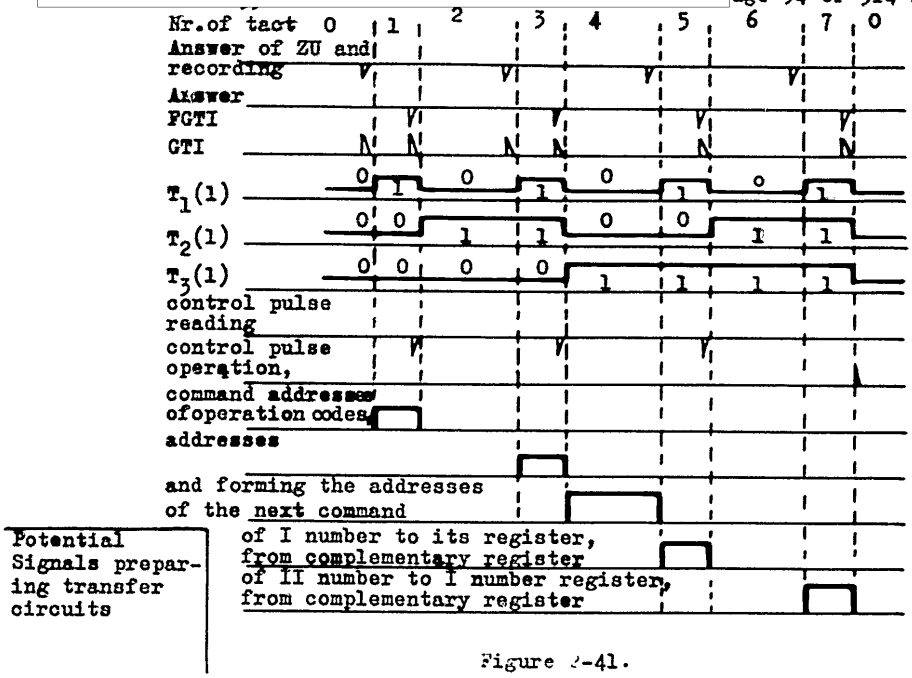


Figure 3-41.

The voltage from the counter trigger anodes is fed to the inputs of the logical diode circuit DLS and in dependence from the state of the counter at the corresponding output of this circuit there will be a potential signal of a high level which passes thru the decoders and opens the required valves. In the same manner the preparation for performing certain elementary is made (transfer of codes from one register to another one, clearing of a register, and others). These operations themselves are produced by the main step pulses which pass thru the valves prepared in this manner.

For starting the computer it is necessary to set the address of the first command. The counter is placed in "zero" state by pushing the "Button for 'zero' setting". From the "Start button", a negative pulse is fed to the input of the control circuit, which acts as an answer pulse of the memory unit, after which the computer begins to work automatically.

During the "zero" step of the work cycle (counter S is in "zero" position) the negative pulse "Answer from the memory" is fed to the control unit, indicating the termination of recording the result, which was obtained during the performance of the preceding command. From this pulse the main step (positive) pulse GTI is activated in the FGTI block and enters valve K_1, to whose second input in "zero" state of the counter a high voltage level is fed from the outlet 0 of the DLS circuit. The negative pulse "Clearing", formed by valve K_1 sets to "zero" the register of the first number of the arithmetic unit and the selective address register. This same pulse through

STAT

a chain of entrapment is fed into the control network (PGTI Answer pulse).

The PGTI network, whose time shift affecting the GTI shaping of pulse generates a negative pulse, changes thereby the counter to state 1 and begins with the first work step. At this state of the counter, a signal appears at outlet 1 of the DLS network, opening the respective valves and prepares the transmission of the serial command from the start register to the selection register.

Then, the signal "Answer PGTI" arrives from the delay line at the input of the control, corresponding to the control pulse "Clearing" generated by the circuit during the preceding step. From the pulse "Answer PGTI", the circuit shapes the second positive pulse GTI, which performs the transfer of the address of the serial command to the selection register(1).

Pulse GTI passes thru the valve K_2, open in state 1 of the counter, and forms the control pulse "Reading", which causes the reading of the commands in the auxiliary register of the arithmetic unit. Then, with the time shift a negative pulse is generated, changing the counter to state 2.

The work of the control network is interrupted until the arrival of the pulse "Answer ZU", which enters from the memory at the end of the second step at the moment of the command reading. The PGTI circuit generates from the pulse "Answer ZU" a third positive pulse GTI, and then after some time shift, the pulse which changes the counter to state 3 is generated.

The third pulse GTI enters valve K_1, open during state 2 of the counter, and at the outlet of the valve once more a negative pulse is shaped for clearing the first number register and the selection register. The same pulse, after having passed thru the delay line, enters the input of the control network (pulse "Answer PGTI") when the counter is already in position 3. With it, the circuit shapes the fourth GTI pulse, which, after having passed thru corresponding valves, opened by the high-level potential signal at outlet 3 of the DLS circuit (in counter position 3), performs the transfer of the operation code from the auxiliary register to the operation unit, and the transfer of the address of the first number to the selection register and the addition of the unity of the lower digit in the start register (forming the address of the next command). Further-

(1) The work of the circuit for transferring codes from one register to another was described in Paragraph 2-4.

STAT

more, pulse GTI shapes the control pulse "Reading" at valve K_2, causing thereby the reading of the first number of the auxiliary register. Then with some time shift, the pulse is shaped which changes the counter to state 4, after which the control network awaits the signal from the memory concerning the reading of the first number.

At the moment of reading the first number, the signal "Answer ZU" enters the input of the control network and the fifth GTI pulse is generated which passes thru the valves opened by the high-level potential signal at outlet 4 of the DLS circuit and performs the transfer of the address of the second number from the auxiliary register to the selection register. This same pulse shapes the pulse "Answer FGTI" at valve K_3 which enters the delay line.

Then after some time shift the counter moves to state 5. Thereafter the signal "Answer FGTI" is fed from the delay line to the input of the network and generates the sixth GTI pulse, which passes thru the valves opened by a high-level signal from outlet 5 of the DLS circuit, and performs the transfer of the first number from the auxiliary register to the first number register. Simultaneously, pulse GTI generates at valve K_2 the pulse "Reading" (of the second number), and at valve K_4 the pulse "Clearing the register of the second number" thereby preparing this register for the reception of the second number.

Then with some time shift the counter moves to state 6 and the network awaits the answer signal from the memory unit, arriving at the moment of reading the second number.

The signal "Answer ZU" generates the seventh GTI pulse in the FGTI circuit, which forms the pulse "Answer FGTI" in valve K_3, then entering the delay line. The counter changes to position 7 whereby the valves are opened for the transfer of the number from the auxiliary register to the second number register whereas the transfer itself is performed under the influence of the eighth impulse GTI, after the impulse "Answer FGTI" entered the circuit input from the delay line. Thereupon an impulse arises which throws the counter over in the position "0".

The blocking generator B, which generates the positive control impulse "Operation" and activates the mechanism of local control operating in the arithmetical knot in accordance with the code preset in the operational block, comes into action at the moment of throw-over of the counter's trigger T_2 in the "zero" position.

STAT

2-10. The principal characteristics of digital computing machines.

With respect to their purpose, the computing machines can be universal or special if they intend to solve special type problems. The structural features of such special machines are relatively simple. The principal characteristics of the computing machines include the number of operational columns (usually 30-40 binary columns), the capacities of the inner (operational) and of the external memory devices, the work speed, the way of introduction of comma (or decimal point), the employed system of calculation, the peculiarities of the code (single-address, double address, or three-address) and the character of operations the machine is capable of performing. Besides, the computing machine is characterized by the number of tubes employed, by the amount of energy that is consumes, by an area required for its accommodation and by the number of attending personnel. The universal digital computers can be subdivided into three classes, - the large, the medium and the small. The large computers are primarily intended for solution of present day problems in physics and engineering. Such machines work at a speed capable of handling several thousands or tens of thousands of operations per second and they contain several thousand electronic tubes. The BESM computer constructed under the direction of Academician S.A. Lebedev (L. 16), can serve as an example of such machines. It works at a speed of 8-10,000 operations per second and contains 4,000 tubes.

The BESM computer has a rapidly operating memory device on ferrite cores, with a capacity for 1,024 numbers. In addition, it has a memory device of a magnetic drum type, with a capacity for 5,120 numbers and another memory device on magnetic bands, of a practically limitless capacity. It operates with binary numbers, has a floating comma (decimal point), uses 32 columns for representation of the mantissa of a number, five for the sequence of a number and two columns for symbols of the mantissa and the sequence.

The small computing machines are employed primarily for solving economic and engineering calculations. In order to reduce their size, they are usually provided with a fixed comma. They have about 1,000 electronic tubes, or less. The magnetic drum is the principal memory device in such machines. They are devoid of any special (operational) rapidly-acting memory devices, or, in the best case, have them for a limited (10-100) number of addresses. Depending on the rate of velocity of the drum's rotation, the speed of these machines lacking the memory cells, averages from 30 to 60 complete arithmetical operations per second. In order to shorten the time of getting to the magnetic drum, the recording and the reading used to

STAT

be switched over onto the heads located on each path of the drum, so as to use that head which will be first approached by the cell carrying the wanted address. Besides, the same purpose is served by the use of such programming systems, where the command code contains the address of the next command. This way, the numbers and commands are spread over the drum in such a way, as to secure the largest possible number of operations during one revolution of the drum. The class of small computing machines includes the Soviet-made machines M-3 and "Ural", purported for engineering calculations. Their basic characteristics will be indicated in the following paragraphs. The computing machines of the medium class possess characteristics of interim character, being half-way between those of the large and the small ones. For example, the computing machine M-2, designed under the direction of Corresponding Member of the AS USSR I.S. Bruk, contains 1,676 tubes and performs 2,000 operations per second. Its inner electronic-tube memorizing device has a capacity for 512 numbers. It can function with the floating or with the fixed comma. The column network has 34 binary columns (L. 4).

2-11. The universal digital computing machine M-3.

This machine is of a small size, designed and constructed by members of LUMS, AS USSR, and NIIEP Gosplana USSR (L. 3 and 5). It is intended for use by computing laboratories, designing offices and scientific-research institutes.

The principal data of the machine is as follows:

Precision of computation	Up to 9 decimal columns
Spacing of numbers at input into the machine and output onto print	7 decimal or 10 octonary
Computing system for numbers at input into machine and output onto print	Decimal and octonary
Computing system for numbers and commands in the machine	Binary
Spacing of numbers in the machine	30 binary columns and one column of the number's symbol
The way of reproduction of numbers in the machine	With comma affixed in front of the senior column
System of programming	Single and twin-address
Number of commands	48

STAT

Speed of performing an operation within the arithmetical knot:	
-addition	60 mk/sec
-subtraction	75-120 mk/sec
-multiplication	1,900 mk/sec
-division	2,000 mk/sec
Total operational speed (at 3,000 rev/sec of the drum)	On an average 1,800 complete arithmetical operations per minute, with recording of the results
Capacity of memory device on magnetic drum	2,048* numbers or twin-address commands
Speed of entry from perforated band:	
-electromechanical transmitter	30 numbers per minute
-photo transmitter	1,200 numbers per minute
Output device:	
Speed of printing and perforation of results	30 numbers per minute
Number of tubes	770
Intake power	10 kw
Area of machine's cabinets	3.3 sq m

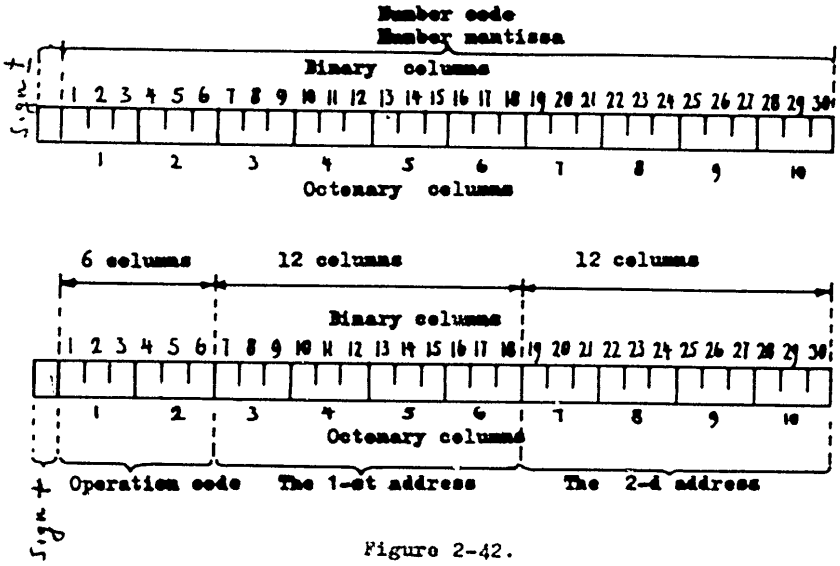


Figure 2-42.

* In the first models 1,125.

STAT

The memory device's capacity can be increased to 4,096 by the addition of a second memory device on the magnetic drum or on the ferrite cores.

The M-3 machine has parallel action, i.e. it has the arithmetic and memory devices of parallel types. The machine is operated asynchronously. The possibility of adding of a unit of quick-acting ferrite memory to the machine's structure, allows for an increase of work speed to 1,500 - 2,000 operations per second, which corresponds to the work speed of the arithmetic and control mechanisms.

Figure 2-42 shows the codes of numbers and commands of the machine M-3. The numbers appear in the form of 30 column binary numbers with the comma affixed before the senior column. The 31st column serves for encoding the symbol of a number (in this column "0" corresponds to plus, "1" to minus).

The basic system of programming is the twin-address. For recording the commands - 30 binary columns are used, be it with a plus or with a minus symbol. Six senior binary (or two octonary) columns are used for the operation's code, the rest of the columns are subdivided in two, per 12 binary columns (or per four octonary), and are used for recording the addresses of the first and the second number. The machine performs all arithmetic operations and a series of logical and auxiliary operations - such as transfer of a number from one memory cell to another, logical multiplication, conditional and unconditional transfer of control and intake of numbers from the perforated tape. The conditional transfer operation, depending on the type of symbol of preceding operation, transfers the control by the first or by the second command address of the conditional transfer.

The commands, i.e. the codes of the operations, are registered in octonary notation. The type of operations is encoded by the second octonary figure of the operation's code. In the cases of arithmetical operations and of logical multiplication, the first octonary code figure determines the additional specificity of the executed operation, whether the recording into the memory device, the printing, etc. are actually being performed (see Table 2-3). A table of commands of the machine M-3 is given in Supplement I.

The M-3 can operate as twin-address and as single-address with counter-tank. In the second case it makes use of the result of the preceding action retained on the recorder of the arithmetic unit. The single-address operations are recorded, as a matter of fact, as twin-address operations: the second address is then filled with

STAT

zeros.

The operational principles of the arithmetical, the memorizing and the controlling devices of M-3 have been briefly described in respective paragraphs of this chapter.

Table 2-3.

Octonary figure of operation's code	Type of operation (coded by the second octonary figure of the code of operation	Particularity of operation (coded by the first figure of the code of operation)(1)
0	Addition	Operation with the numbers from the first and the second addresses, with recording of result by the second address.
1	Subtraction	Same as for 0, but without recording the result in the memory device.
2	Division	Operation with the numbers from the first address and the results of the preceding operation retained in arithmetical device, with recording the result by the second address.
3	Multiplication	Same as for 2, but without recording the result in the memory device.
4	Conditional and unconditional transfer of the control	Same as for 0. Besides, the result is printed.
5	Transfer of numbers from one memory cell to another	Same as for 1, but operation affects the modules of numbers.
6	Logical multiplication	Same as for 2. Besides, the result is printed.
7	Input	Same as for 3, but operation affects the modules of numbers.

(1) Concerns only to arithmetical operations and logical multiplication.

2-12. The universal digital computing machine "Ural".

The computing machine "Ural"(1) belongs in the class of small automatic digital computers. It is intended for use in the engineering research and has the following basic characteristics(2):

Computation precision	To 10 decimal columns
Grouping of decimal numbers at their input in the machine and at their output onto print	9 decimal columns (the last one is even)
Computation system for numbers at input into the machine and output onto print	Decimal and octonary
Computation system for numbers and commands	Double
Grouping of numbers in the machine	35 binary columns and one column for the number's symbol
The way of representing the numbers	With the comma affixed before the senior column
Programming system	Single-address
Total operational speed (of drum at 6,000 rev/min)	100 single-address, single-cycle operations
Number of commands	30
Number of cycles at execution of separate operations:	
-all operations except of division, normalization and group-operations	1 cycle
-normalization	2 cycles
-division	4 cycles
-group operations	Depending on the command
Duration of one cycle	10 msec
Capacity of memory device on the magnetic drum	1,024 figures or 2,048 single-address commands
Counter tank on the magnetic tape	40,000 of 36 column numbers, or 80,000 commands
Counter tank on the perforated tape	To 10,000 numbers or commands

(1) In this paragraph use was made of the article by Yu.Ya. Bazilevskiy (L. 1)

(2) The given data are based on the pamphlet "The "Ural" Universal Computing Machine", All Union Trade Exhibition, 19-6

STAT

Speed of input from perforated tape	4,500 numbers per minute
Speed of printing the results	100 numbers per minute
Number of tubes in the machine	700
Source of power	Three-phase current 220 v $\pm$ 10 %, frequency 50 gc
Intake power	7.5 kw

Figure 2-43 represents the command and the number codes of the computer "Ural. The numbers are represented by means of 35 binary columns (corresponding to 10.5 decimal columns). The comma is fixed before the senior column of the number. The 36th column is earmarked for representing the sign (plus is depicted as 0, minus as 1). Thus, the range of numbers of operational moduli is from 2^{-35} to 1. Dispositions of a binary number in a full and in a not full cell are shown in Figure 2-43, a and b.

The command code (Figure 2-43b) occupies a half of the network's columns, i.e. 18 binary columns, wherefrom 5 columns are used for the operational code, 11 for the number's address, one for encoding the sign of a full cell, one for encoding the sign of the blocking of the overfill. One full cell of the memory device accommodates two commands.

Figure 2-44 represents a block diagram of the "Ural"'s arithmetical . .. It consists of an accumulating summator, a basic register, a quotient register, an output register, a local program pick-up (block of local control) and an input shifter.

The machine "Ural" has a consecutive-parallel arithmetical From the memory device on the magnetic drum (also from the magnetic tape), the number codes are fed through the input shifter into the basic register, per nine binary columns in parallel manner, or per four columns consecutively. In a similar way is performed the transfer of the number codes from the summator onto the magnetic drum and onto the magnetic tape. The machine has a parallel-action accumulating summator.

The positive numbers appear in the summator directly encoded, the negative numbers appear reversely encoded.

The synchronizing (cycling) impulses are fed from the magnetic drum (one cycle per one revolution). The operation of addition is performed in one cycle, of division, in four cycles.

A complete table of the "Ural is given in Supplement II. At the addition or subtraction, the number in cell with the address indicated

in the command is added to the number in the summator, or is subtracted therefrom. At multiplication, the number in the summator is multiplied by the number from the cell with the address indicated by the command: the product is formed in the summator. At the division, the dividend is in the summator, the divider is in the cell with the address indicated by the command.

Besides, the multiplication can be performed the following way: the number in the arithmetical knot's register is multiplied by the number from the cell with the address indicated by the command. The product is added to the number in the summator. This operation is convenient for calculations of sums of conjugated products $a_1b_1 + a_2b_2$, since it reduces the number of required resortings to the memory device.

The advantageous point of the machine "Ural" consists in its capability of performing such operations which can, sometimes, facilitate a reduction of the program's volume and increase the work speed (normalization, shift of codes by a prescribed number of columns, summation of conjugated products, repetition of the calculation cycles a desired number of times).

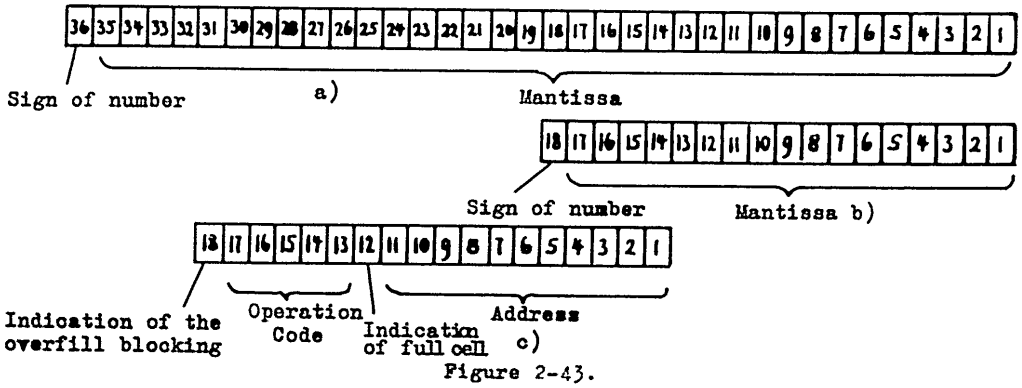


Figure 2-43.

So, for example, the ability of the machine to shift the code by a desired number of columns and normalization accelerate the calculation process by using the floating comma(1).

(1) Calculation with a floating comma can be performed by a machine provided with fixed comma, through the utilization of special subprograms.

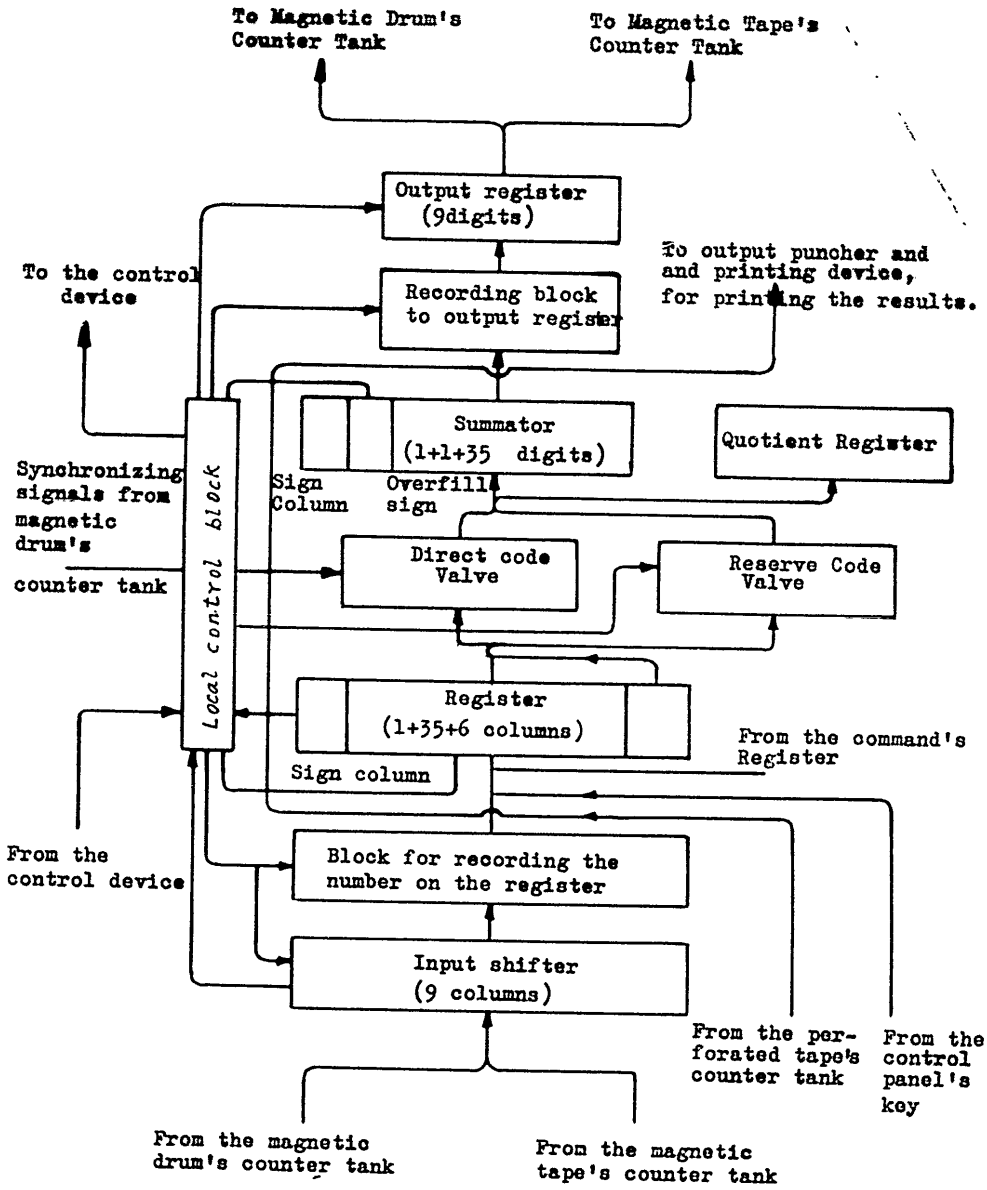


Figure 2-44

THIRD CHAPTER

The Technique of Programming.

3-1. The simplest example of a program.

The first chapter contained a general definition of the program of solution by the machine, of mathematical tasks, description of various codes of automatic digital computing machines and description of methods of holding the commands in the machine. In this chapter we shall examine the basic ways of composition of programs and, if not indicated otherwise, shall do this on the basis of the conditional machine with the floating comma, considered in the first chapter. Let us start off with the simplest example brought up in Paragraph 1-3. As we noted, for calculation of determinant

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

the following actions must be executed:

1	x	a	d	ad
2	x	b	c	bc
3	-	ad	bc	ad - bc

In order to perform these calculations by means of the machine, the figures a,b,c and d should be placed in the memory device. Let them be stored in the memory cells α, β, γ and δ. Besides, extra cells should be allotted for safekeeping of obtained products ad and bc, and of the final answer ad - bc. Let them be cells ζ_1, ζ_2 and ξ. In order to get the obtained result out of the machine, we supplement the three above named actions with a command, which would print the result, i.e. the content of the cell ξ. For stopping the machine, the command "Stop" should be put at the end of the program. Now we can compose the program of calculation of the determinant of the second order. This program is shown in a table below. It contains five commands. On the left of every command there appears the number of the cell in which it is stored, whereas on the right of it appears the operation's result. Under the commands, the table shows in which cells the figures are stored.

Cell Nr	Operation	IA	IIA	IIIA	Results
k + 1	x	a	δ	ζ_1	ad
k + 2	x	β	γ	ζ_2	bc
k + 3	-	ζ_1	ζ_2	ξ	ad - bc
k + 4	Print	ξ	-	-	ad - bc
k + 5	Stop				

α	a
β	b
γ	c
δ	d

The feeding of such a program into the machine is performed in the following manner: At first, the concrete numbers should be assigned to the cells. After that, every command should be encoded by a numerical expression. This engenders a program having the form of a sequence of five binary numbers. For their storing somewhich five sequential memory cells should be allotted. Their numbers are shown as k + 1 - k+5. At last, the program and the figures a, b, c and d should be fed into the cells allocated for them, and then we can get about the solving of the problem. For this purpose we compose, on the address register, number k + 1, i.e. the number of the cell storing the first command, and engage the machine. Calling upon the consecutive memory cells beginning from the cell k + 1, the machine will now execute one command after another. It will compute the number ad - bc, will have it printed (command k + 4) and finally will be stopped by the command k + 5. Let us note, that for getting this problem solved we needed 12 memory cells: five for storing the commands, four for storing the initial numbers and three for storing the final computation results.

For another example, we shall now examine a program of multiplication of two complex figures.

$$(a_1 + ib_1)(a_2 + ib_2) = a_3 + ib_3 ,$$

where, as it is known,

$$a_3 = a_1a_2 - b_1b_2 , \quad b_3 = a_1b_2 + a_2b_1 .$$

It is impossible to hold in the machine such complex figures. Only a substantial part and a coefficient of an imaginary part can be held. For this, for example, the cells α_1, β_1, α_2, β_2 and can be used. Likewise the product of multiplication must be segregated in two separate cells α_3 and β_3. Therefore, the program

STAT

UNCLASSIFIED

of multiplication of two complex figures will look like this:

k + 1	x	α_1	α_2	ζ	a_1a_2
k + 2	x	β_1	β_2	α_3	b_1b_2
k + 3	-	ζ_1	α_3	α_3	$a_1a_2 - b_1b_2 = a_3$
k + 4	x	α_1	β_2	ζ_1	a_1b_2
k + 5	x	α_2	β_1	β_2	a_2b_1
k + 6	+	ζ_1	β_3	β_3	$a_1b_1 + a_2b_1 = b_3$
k + 7	Print	α_3	-	-	a_3
k + 8	Print	β_3	-	-	b_3
k + 9	Stop				

α_1	a_1
β_1	b_1
α_2	a_2
β_2	b_2

In this program, the results of intermediary actions a_1a_2 and a_1b_2 are placed in the cell . Such cells containing the results of intermediary actions usually are called "work cells". The product a_1b_2 can also be placed in the same cell, since we have no more need in the figure a_1a_2 stored therein.

In the command k + 2, the product b_1b_2 is placed in cell a_3, which is thereby used temporarily as a work cell.

It is evident, that the computation of the above-considered determinant or the products of complex figures with the use of the machine is more complicated than by hand. In this connection, the following should be noted: (text is missing) ..... if a separate command is allotted to every action, at plotting the program, then the thus created program will have no practical value. Its plotting will require a time equal to that needed for computation by hand, and the machine will have to have a huge number of cells in the memory device. However, almost every computation process consists of recurrently repeated parts or cycles of a small number of operations with identical successions of actions, where only the figures being operated upon are subject to changes at various stages of actions. For example, let the numerical solution of a common differential equation $y' = f(x, y)$ be treated by Euler's method, by the formula

$$y_{i+1} = y_i + f(x_i, y_i)(x_{i+1} - x_i) .$$

For determination at every point X_0, X_1, ... X_n of the values of unknown function y(x), only one succession of actions should be made, merely replacing, on the way from point X_{i+1} to point X_{i+2}, of values X_i, Y_i by values X_{i+1} and the newly calculated value Y_{i+1}. Making use of this peculiarity of calculation processes, it is necessary to be able to plot such programs (herein rests the skill of the program-plotting) in which a large number of actions is covered by a small number of commands. This can be accomplished with the use of two principal methods explained in the following paragraphs.

3-2. Transformation of contents of cells.

The first of these methods can be well understood from the following simple example. Let the machine be applied for computation of power a^{20}. First of all the figure a must be fed into some cell of the memory device. In some cell β we shall successively form the powers a, a^2, a^3, ... The program of computation of a^{20} will have the following appearance (on the right of the table are shown the results of every action):

k + 1	PCh (Print)	"1"	–	β	1
k + 2	x	α	β	β	a
k + 3	x	α	β	β	a^2
k + 4	x	α	β	β	a^3
...	.....	...	...	...	...
k + 21	x	α	β	β	a^{20}
k + 22	Stop				

The first command to the cell β transfers one. Instead of making an indication in this command of the number of that cell which stores the "one", we simply show the "one" as number "1" (in quotation marks). Henceforward we shall indicate the cell containing an "a" as α. The unity 1 is fed into the cell β in order to be able in the command k+2 to multiply it by the content of the cell α, i.e. create in the cell β a figure a. Such program includes 22 commands, yet, it is evident, 20 of them are entirely identical. Instead of plotting 20 identical commands, one can try to reproduce 20 times the very same command. For this purpose one employs the operation of transfer of the control, whereby the program is changed to the following simplified appearance(on the right of the table are shown the results of actions at consecutive stages of computation):

STAT

k+1	Print (PCh)	"1"	-	β	1			
k+2	x	a	β	β	a	a^2	a^3	....
k+3	Transfer of control (PU	k+2	-	-				

In this program, the command k+3 transfers the control to the cell k+2, that is, the command k+2 will be executed again and again, but the content of the cell β will all the time be changed. If at the first execution of command k+2 it contained number 1, then, before the second execution of the same command the cell β will contain the figure a, before the third execution a^2, and so on. Thus, a sequence of powers a, a^2, a^3, etc, will be formed therein. The thus formed program will have a disadvantage, in that the machine will not stop at a time when the desired number a^{20} is formed in the cell β, but will go on working, all the time executing the same operations k+2 and k+3. It will "get cycled", so to say. In order to preclude this event, one should employ the operation of comparison, mentioned in Paragraph 1-9. For this, some cell γ should be singled out and employed to serve as a counter. Prior to its use it should be "cleared" from any other content, i.e. it is to contain zero. After every execution of command k+2, the cell γ would get a unity (a one) fed into it, by means whereof it would count up how many times the command k+2 was executed in the cell β. Upon the introduction of the command of comparison, that command would compare the number 20 with the number formed in the cell γ. As long as that number would be less than 20, the operation of comparison would refer to the command k+2 (the cell k+2 should be placed in the third address of the command). As soon as the cell γ would contain the number 20, the operation of comparison would let pass, whereupon the program would be as follows:

k	PCh (Print)	"0"	-	γ	0				
k+1	PCh	"1"	-	β	1				
k+2	x	α	β	β	a	a^2	a^3	....	a^{20}
k+3	+	"1"	γ	γ	1	2	3		20
k+4		γ	"20"	k+2					
k+5	Stop								

It is evident, that in this program, the commands k+2 - k+4 are executed 20 times. In Paragraph 3-1 such groups of commands, executed several times, have been named a cycle. The introduction of such cycles makes it possible to construct programs containing but a small number of commands, which are, however, sufficient for the machine to perform a huge number of operations. Should we place the

STAT

numbers 0, 1 and 20 into respective cells ζ_1, ζ_2, and ζ_3, the program would assume its final appearance:

k	PCh (Print)	ζ_1	-	γ	0					
k+1	" "	ζ_2	-	β	1					
k+2	x	α	β	β	a	a^2	a^3	.....	a^{20}	
k+3	+	ζ_2	γ	γ	1	2	3		20	
k+4	$\neq$	γ	ζ_3	k+2						
k+5	Print	β								
k+6	Stop									
ζ_1	0									
ζ_2	1									
ζ_3	20									
α	a									

Here again, certain numbers of cells should be taken to serve as k, k+1 ...., k+6, ζ_1, ζ_2, ζ_3, α, β and γ, and the program should get encoded, that is, be represented by a succession of binary numbers. Let us note, that the operation of comparison k+4 in its third address contains the number of the cell k+2, i.e. one of the very cells which store the program itself. That being so, the numbers 0, 1, 20 and the program itself may be now brought into the memory cells earmarked for them, whereupon we may get down to solving the problem. The only action to be made is to compose on the control panel the number k and to engage the machine. Executing the program, the machine will have the quantity a^{20} computed and printed and then will come to a stop. It should be noted, that the number of commands contained in this program does in no way depend on the quantity of exponent 20.

All we should do, in order to find out any desired power a^n with the use of the same program, is to replace in the cell ζ_3 (not in the program itself!) the 20 by "n".

Now, let us examine a somewhat more complex example. Let it be required to find out the squares of consecutive numbers

$$1^2, 2^2, 3^2, \ldots\ldots, N^2.$$

For this, the numbers 1, 2,...., N are placed, respectively into the cells ZU, numbered as a_1, a_2,....., a_n. The program may look as follows:

STAT

k+1	x	a_1	a_1	α	1^2
k+2	Print	α	-	-	1^2
k+3	x	a_2	a_2	α	2^2
k+4	Print	α	-	-	2^2
.....	.....	.....	.....	.....	.....
k+2N+1	x	a_N	a_N	α	N^2
k+2N	Print		-	-	N^2
k+2N+1	Stop				
a_1	1				
a_2	2				
.....	.....				
a_N	N				

Apparently, the program for computation of squares of N numbers requires 3N+2 cells of the memory device (N of cells a_1, a_2,.... a_n for storing the numbers 1, 2,....., N, cells a and 2N+1 for storing the commands). The program consists of a unit containing two commands, repeated N times. Its second commands are the same, whereas the first commands differ from them in that the numbers 1, 2,...., N in them are being multiplied by themselves. However, the numbers 1, 2, ..., N can be formed in the cell β consecutively, by way of adding to its content the number one. Then, the cells a_1, a_2,..., a_n in the commands k+1, k+3,...., k+2N-1 can be replaced by the cell β and the program would look as follows:

k+1	PCh .	"0"	-	β	0
k+2	+	"1"	β	β	1
k+3	x	β	β	α	1^2
k+4	Print	α	-	-	1^2
k+5	+	"1"	β	β	2
k+6	x	β	β	α	2^2
k+7	Print	α	-	-	2^2
.....	.....	.....	.....	.....	.....
k+3N-1	+	"1"	β	β	N
k+3N	x	β	β	α	N^2
k+3N+1	Print	α	-	-	N^2
k+3N+2	Stop				

Now the whole program consists of a cycle repeated N times, containing entirely the same commands (the first command "clears" the cell β , i.e. sets it at zero). The kind of the commands does not change with the repetition of the cycle, so that the whole program can be recorded by only five commands:

STAT

k+1	PCh	"0"	-	β	0					
k+2	+	"1"	β	β	1	2	3		N	
k+3	x	β	β	α	1^2	2^2	3^2	....	N^2	.....
k+4	Print	α	-	-	1^2	2^2	3^2		N^2	
k+5	PU	k+2	-	-						

Upon the completion of the cycle, the command k+5 brings the procedure back to the outset of the cycle (it transfers the control to the cell k+2), but the cell β contains presently not zero, as it had at the first execution of the command k+2, but the number one. Consequently, the addition of the number one to the content of the cell β, forms therein the number two and by repetition of the program it will have 2^2, and so forth. Like the program referred to before, this program can "get cycled" and we must again employ the operation of comparison, to preclude this.

k+1	PCh	"0"	-	β
k+2	+	"1"	β	β
k+3	x	β	β	α
k+4	Print	α	-	-
k+5	$\prec$	β	"N"	k+2
k+6	Stop			

At the close of every stage of computation, the number N is compared with the content of the cell β, i.e. with the number the square whereof has already been found. As long as that number is less than N, the operation of comparison refers us to the command k+2 and we go through the whole cycle all over again. But, as soon as, after a number of operations, a number N is formed in the cell β, the operation of comparison lets pass and the machine comes to a stop. Now we have a program consisting of six commands, capable of computing the squares of any desired quantity of numbers, prior to its coming to a stop, by itself. (The number of commands in the program does not depend on the number N: only "the constant of comparison" of N changes along with N).

The same group of commands (k+2 - k+4) was made use of again, by way of constructing of a program containing a recurrently-repeated cycle, engendered by the changing of the number stored in the cell β at every new passage of these commands. The operation of comparison was used for securing the repetition of the cycle a desired number of times. That operation compared the content of the cell β with the number N. Number one was additionally fed into the cell β at the beginning of every cycle, thereby imparting to it the function of

a counter. The same principle is used for computation of factorials N of consecutive numbers

$$1!;\ 2!,\ \ldots\ldots,\ N!$$

Since $(n+1)! = n!(n+1)$, it is sufficient to take some cell α and have numbers 1, 2, ..., N formed in it, then multiply the already known factorials by them.

k+1	PCh	"1"	-	α	1				
k+2	PCh	"0"	-	β	0				
k+3	+	"1"	β	β	1	2	3		N
k+4	x	α	β	α	1!	2!	3!		N!
k+5	Print	α	-	-	1!	2!	3!	....	N!
k+6	<	β	N	k+3					
k+7	Stop								

3-3. The programs with automatic choice of number of cycles.

In this paragraph, we shall examine a more complex program, in which the number of repetitions of the cycle is not known in advance. Let us suppose, that the value of the function e^x for some value of the argument x must be found out with a set absolute error ε, by way of expansion in the series:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots\ldots\ldots$$

Depending upon the quantity of x, we shall have to find a different number of members, for securing the set degree of precision, when this number of members is not known beforehand. In order to determine the precision of computation at the given x, we shall have to, after the addition of the next member, evaluate the whole remainder. Should that remainder be more than ε, we shall have to compute one more member and to go on doing so until the remainder became less than .

If $|x| < 1$, then the whole remainder

$$R_n = \frac{x^{n+1}}{(n+1)!} + \frac{x^{n+2}}{(n+2)!} + \ldots\ldots$$

by the modulus is less than the last undiscarded member, i.e.

$$|R_n| < \frac{x^n}{n!}.$$

Consequently, the process of summation must be stopped as soon as the last computed member becomes less than ε. Therefore, all members in the series must be consecutively compared with ε. In the program adduced below, the number x is placed in the cell a,

whereas the sum is being built up in the cell β . The consecutive whole numbers 1, 2, 3, ... are formed in the cell γ_1 and the n-member of the series $\frac{x^n}{n!}$ - in the cell γ_2 . Inasmuch as the cells β , γ_1 , γ_2 may still retain some numbers left over from the solution of preceding problems, they either must be cleared out by putting in the zeros (this must be done to the cell γ_1), or they should be filled with the numbers that would be formed at the first stage of computation (the value of the first member of the series,- the number one-, is put in the cell β), or they can be filled with such number, the value whereof will be used later (the number one is put in the cell γ_2). Anyway, every program must begin with commands in which the cells used for this program are being "prepared" but whose contents at the outset of the program were still unknown. Hereinafter we shall not repeat the explanation of the commands that are "preparing" the cells. Assuming that $|x| < 1$, the program of computation will be as follows:

Program of computation of e^x

k+1	PCh	"1"	-	β	1			
k+2	PCh	"0"	-	γ_1	0			
k+3	PCh	"1"	-	γ_2	1			
k+4	+	"1"	γ_1	γ_1	1	2		
k+5	x	γ_2	α	γ_2	x	$\frac{x^2}{1!}$		
k+6	:	γ_2	γ_1	γ_2	$\frac{x}{1!}$	$\frac{x^2}{2!}$		
k+7	+	γ_2	β	β	$1+\frac{x}{1!}$	$1+\frac{x}{1!}+\frac{x^2}{2!}$	....	e^x
k+8	$\lvert<\rvert$	"ε"	γ_2	k+4				
k+9	Print	β	-	-				
k+10	Stop							
α	x							

In here, the command k+8 compares the number ε with the modulus of the next computed member $\frac{x^n}{n!}$ (cell γ_2) and, as long as this member is more than ε , returns us right to the outset of the cycle (the cell k+4 stands in the third address of this command). Thus, by comparing the set number with $\frac{x^n}{n!}$, formed in the computation process, the machine itself determines the necessary number of repetitions of the cycle. It should be clear, from this, what a great role for the digital computors constitute the conditional transfer operations, which make it possible to switch over from one part of program to another, at a proper, yet not known in advance, moment. The block-diagram (logical scheme) (Figure 3-1), facilitates the comprehension

STAT

of this program.

By analogy with the above, applying the expansion

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots\ldots,$$

we can construct a program for computation of the value of function sin x at a point x. The difference is, that here the next member is found by way of multiplication of the preceding one by $-x^2$ and through division by (n+1)(n+2).

Program of computation of sin x

k+1	PCh	α	-	β	x
k+2	x	α	α	γ_1	x^2
k+3	- PCh	γ_1	-	γ_1	$-x^2$
k+4	PCh	"0"	-	γ_2	0
k+5	PCh	"1"	-	γ_3	1
k+6	x	α	γ_1	α	$-x^3, \frac{x^5}{3!} \ldots$
k+7	+	"2"	γ_2	γ_2	2, 4, ...
k+8	+	"2"	γ_3	γ_3	3, 5, ...
k+9	:	α	γ_2	α	$-\frac{x^3}{2}, \frac{x^5}{3!4}$
k+10	:	α	γ_2	α	$-\frac{x^3}{3!}, \frac{x^5}{5!}$
k+11	+	α	β	β	sin x
k+12	|<|	"ε"	α	k+6	
k+13	Print	β			
k+14	Stop				
α	x				

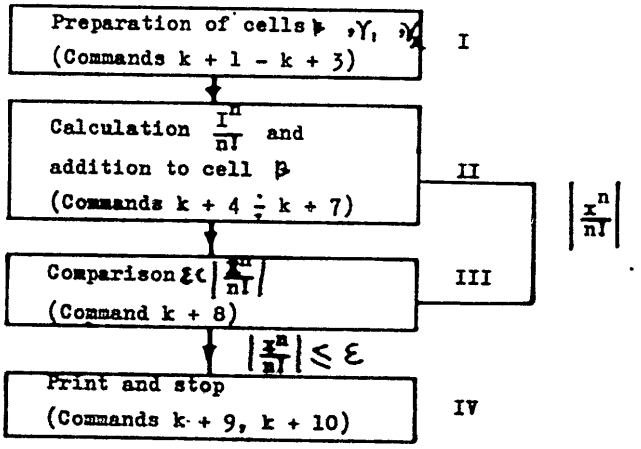


Figure 3-1.

The reader can draw up the block diagram of this program, himself. With the use of transformation of formulas employed for computation, the number of the commands in the program can be reduced. Differences of ratios of denominators of two consecutive members of the series for sinus, form an arithmetic progression. Indeed, if we state that:

$$\sin x = \frac{x}{c_1} - \frac{x^3}{c_2} + \frac{x^5}{c_3} - \frac{x^7}{c_4} + \ldots, \qquad c = (2i-1)!$$

and designate

$$a_i = \frac{c_{i+1}}{c_i},$$

then

$$\Delta_i = a_{i+1} - a_i = \frac{c_{i+2}}{c_{i+1}} - \frac{c_{i+1}}{c_i} = \frac{(2i+3)!}{(2i+1)!} - \frac{(2i+1)!}{(2i+1)!} =$$

$$= (2i+3)(2i+2) - (2i+1)2i = 8i + 6.$$

By the strength of this, the law of formation of coefficients $\frac{1}{c_1}, \frac{1}{c_2}, \frac{1}{c_3} \ldots\ldots$ of the series will be as follows:

$$\frac{1}{c_{i+1}} = \frac{1}{c_i a_i}; \quad a_i = a_{i-1} + \Delta_{i-1}; \quad \Delta_i = 8i + 6; \quad i = 1, 2, 3, \ldots$$

Let us form Δ_i in the cell γ_2 and a_1 in the cell γ_3. Since $\Delta_0 = 6$, then the cell γ_2 has to get -2 and since $\Delta_1 = a_1 = 6$, then $a_0 = 0$ and, consequently, the cell γ_3 at the outset should get zero. Then, the program will be as follows:

Program of computation of sin x

k+1	PCh	α	-	β	x			
k+2	x	α	α	γ_1	x^2			
k+3	- PCh	γ_2	-	γ_1	$-x^2$			
k+4	PCh	"-2"	-	γ_2	-2			
k+5	PCh	"0"	-	γ_3	0			
k+6	x	α	γ_1	α	$-x^3$	$\frac{x^5}{c_2}$		
k+7	+	"8"	γ_2	γ_2	$\Delta_0 = 6$	$\Delta 1 = 14$		
k+8	+	γ_2	γ_3	γ_3	$a_1 = 6$	$a_2 = 20$		
k+9	:	α	γ_3	α	$\frac{-x^3}{c_1 a_1}$ $\frac{-x^3}{c_2}$	$\frac{x^5}{c_2 a_2}$ $\frac{x^5}{c_3}$		
k+10	+	α	β	β	$\frac{x}{c_1} - \frac{x^3}{c_2}$	$\frac{x}{c_1} - \frac{x^3}{c_2} + \frac{x^5}{c_3}$	...	sin x
k+11	I<I	"ε"	α	k+6				
k+12	Print	β	-	-				
k+13	Stop							
α	x							

The program has been shortened only by one command, however, since that command is contained in the repeatedly performed cycle, this reduction can be of a considerable significance, especially so for the machines possessing a relatively low work speed.

3-4. The operation of addition of the commands.

While considering the various operations the digital computer is capable of performing, we, in Paragraph 1-7, have examined the addition of commands. That operation provides for separate addition of sequences of numbers (respective groups in the command express the code of the operation) and of mantissas of numbers (respective groups express the addresses of the command), and no normalization is performed after such an addition. Using this operation, we can transform the kind of the commands. Henceforth, taking into consideration the operation of addition of commands, we shall put the words "addition" and "sum" in quotation marks.

Let us put into cell α the command

α	x	γ_1	γ_2	γ_3

Now we assign the operation the code 10 000 1 and assume that

$$\gamma_1 = 0 \ldots. 011; \gamma_2 = 0 \ldots. 0111; \gamma_3 = 0 \ldots. 01,$$

then the code of this command will be expressed by the number

100001	0.....011	0.....0111	0....01
p	A		

In other words, the cell α contains number $2^p A$. To this number we "add", the number

$$2^q B = \underbrace{000000}_{q} \quad \underbrace{00.....01}_{B},$$

stored in the cell β and put the "sum" back into the cell α. By doing so we execute the command:

CK	α	β	α

The sequence of the new number is:

$$p + q = 100001,$$

and the mantissa is:

$$\begin{array}{l} 0....0110....01110....01 \\ \underline{0....0000....00000....01} \\ 0 \quad 0110,...01110....10 \end{array}$$

This number cannot be normalized. Thus, the following number is formed in the cell α :

100001	0....011	0....0111	0....010

The code of operation and the first two addresses in the cell α have not change, yet the third address is increased by one. That is, at present the cell α contains the following command code:

x	γ_1	γ_2	$\gamma_3 + 1$

Also, by the use of the operation of addition of commands to the command held in the cell α (of the unnormalized number $2^0 \cdot 2^{-36}$), we change the kind of the command. The number $2^0 \cdot 2^{-36}$ may be called a unit of the third address of our conditional (hypothetical) machine and we designate it 1IIIA. If, as heretofore, the cell storing the number a is designated "a", then the performed operation of addition of commands appears this way:

CK	α	"1IIIA"	α

It becomes clear now, why no normalization can be made after the addition of commands. Indeed, otherwise we should, in the given case, have had to shift the mantissa by ten columns to the left and subtract ten ones from the sequence, which would have entirely changed not only the operation code, but the command addresses as well.

If we "add" to α the number

$$2^0 \cdot 2^{-24} = \underbrace{000000}_{6} \quad \underbrace{0...0}_{12} \quad \underbrace{0...01}_{12} \quad \underbrace{0...0}_{12},$$

which is called the unit of the second address - 1IIA - (it stands in the lower group of the second address) and put the sum not in the cell α, but in the cell β, i.e. execute the command:

CK	α	"1IIA"	β

then the following command would be formed in the cell δ :

δ

x	γ_1	$\gamma_2 + 1$	γ_3

One of the addresses of the command can be transformed not by one but by few of them. So, for example, we can transform the first address by 3 unities, wherefore we should "add" to the command code the number (it is designated 3IA)

$$\underbrace{0...0}_{6} \quad \underbrace{0...011}_{12} \quad \underbrace{0...0}_{12} \quad \underbrace{0...0}_{12}$$

Moreover, two, even three addresses can be transformed at one time. For example, to the command α we can "add" one unit of the first (1IA) and two units of the second address (2IIIA), that is the number

$$\underbrace{0\ldots0}_{6}\quad\underbrace{0\ldots01}_{12}\quad\underbrace{0\ldots0}_{12}\quad\underbrace{0\ldots010}_{12}$$

Then the command α will be as follows:

α	x	$\gamma_1 + 1$	γ_2	$\gamma_3 + 2$

Thus, by selection of an appropriate number we can mutate, at our pleasure, not only the addresses, but also the code of operation of every command.

3-5. Transformation of commands in the programs.

In all the programs considered above, we observed, that numbers stored in some addresses of commands of a cycle underwent changes at every passage of the cycle, yet the kind of the commands as such remained unchanged. Another way of cyclic passage of the same group of commands consists in that the vary kind of the commands is subject to changing at separate stages of calculation. This way, using the operation of addition of commands will be considered now, at first on elementary example. Suppose that 50 numbers $x_1, x_2 \ldots\ldots x_{50}$ must be multiplied by the machine. They can be respectively placed in the cells c + 1, c + 2,....., c + 50. A program containing 52 commands can be easily constructed for the solution of this problem, viz.

K + 1	PCH	"1"	—	α	1
K + 2	x	C + 1	α	α	x_1
K + 3	x	C + 2	α	α	$x_1\ x_2$
K + 4	x	C + 3	α	α	$x_1\ x_2\ x_3$
...	.....	...	...	...	.....
K + 51	x	C + 50	α	α	$x_1\ x_2 \ldots\ldots\ldots x_{50}$
K + 52	Stop				

It can be shortened by way of addition of commands. Let the multiplication operation get the code 000001; a cell with the number $\underbrace{100\ldots0}_{12}$ is taken to be α and a zero cell $\underbrace{00\ldots0}_{12}$ is taken to be c . Then, the command codes k +2 and k + 3 look like this:

k+2	000001	00...001	10...0	10...0
k+3	000001	00...010	10...0	10...0
	6	12	12	12

These two commands, regarded as numbers, differ from each other only in that in the first address of one command stands 1, whereas in the same address of the second command stands 2 (written in binary code). In other words, the command code k+3 surpasses the k+2 by one unit of the first address. Consequently, the command code k+3 can be created by "addition" of 1IA to the number stored in the cell k+2 (i.e. to the command code k+2):

CK	k + 2	"1IA"	k + 2

Presently, the cell k+2 contains the command code k+3. Should we "add" to the content of the cell k+2 a unit of the first address, i.e. execute once more the just formulated command, then we should have the command code k+4, and so on. Thus, by successive "adding" 1IA to the content of the cell k+2, it is possible to have in it all the successively formed dommands k+3, k+4,...., k+51. If, after every such transformation, we turn the control over to the cell k+2, then there will be successively formed with the help of the command located in there at the given moment, all the products x_1, x_1x_2, $x_1x_2x_3$... Presently, the program will contain only four commands:

k + 1	P,Ch.	"1"	-	a	1			
k + 2	X	c + 1	a	a_1	x_1	x_1x_2	$x_1x_2x_3$	...
k + 3	CK	k + 2	"1IA"	k + 2				
k + 4	PU	k + 2	-	-				

As the cell k+4 here has unconditional control, so there is bound to occur the already familiar to us "getting cycled", i.e. upon having produced the products x_1x_2,....., x_{50}, the machine will not come to a stop, but will go on multiplying these products by the numbers stored in the cells c + 51, c + 52,.... and so on (in those cells there still rest some numbers left over from the solution of preceding problem,. In order to prevent the machine from "getting cycled", we shall have to introduce the operation of comparison, which will count up the performed number of multiplications and will bring the machine to a stop at a due time. The final appearance of the program will be this:

k	P.Ch.	"0"	-	β	0				
k+ 1	P.Ch.	"1"	-	a	1				
k + 2	X	x + 1	a	a	x_1	x_1x_2	...	$x_1x_2...x_{50}$	
k + 3	CK	k + 2	"1IA"	k + 2					
k + 4	+	"1"	β	β	1	2		50	
k + 5	≠	β	"50"	k + 2					
k + 6	Stop								

In here, just as in the preceding paragraph, the counter counted up the number of cycle passages and the sum was compared with 50. This operation can be executed by another more convenient way. By the time when all 50 numbers $x_1 x_2, \ldots, x_{50}$ will have been multiplied, the command k+2 will look as follows:

k+2	x	c + 51	a	a

i.e. the cell k+2 will contain (the binary number of 51 is 110011) the number:

$$\underbrace{000001}_{6}\quad\underbrace{000000110011}_{12}\quad\underbrace{10\ldots0}_{12}\quad\underbrace{10\ldots0}_{12}$$

Instead of introducing the counter β and comparing its data with 50, we can compare the content of the cell k+2 with the above-written number and at a moment when both numbers get equal, the operation of comparison will pass and the machine will stop. In other words, this way we compare the code of "variable" command

k+2	x	c + i	a	a	i = 1, 2, ..., 51

being formed in the cell k+2 with the code of "standard" command

x	c + 51	a	a

which is placed, for example, in the cell k+10. Presently this program will be as follows:

k + 1	[illegible]	„1"	:	a
k + 2	X	c + 1	a	a
k + 3	CK	k + 2	„1IA"	k + 2
k + 4	X	k + 2	k + 10	k + 2
k + 5	Stop			
		.		
k + 10	X	c + 51	a	a

Using this method, we have further shortened the program by two commands. Although, it is true, we had to occupy the cell k + 10 by the "standard" command.

The same method can be used for the construction of a program capable of counting up the number of elements of sequence $x_1, x_2, \ldots, x_{100}$, whose modulus is equal to, or less than some positive number (it is assumed, just as above, that x_i is placed in the cell c+i):

k + 1	p < h	"0"	-	a
k + 2	|<|	"ε"	a + 1	k + 4
k + 3	+	"1"	a	a
k + 4	CK	k + 2	"1IIA"	k + 2
k + 5	CK	k + 2	k + 8	k + 2
k + 6	Print	α	—	—
k 7	Stop			
k + 8	<	"ε"	c + 101	k + 4

The command k+2 compares the number ε with the number stored in the cell c+i, that is with x_i. When $\varepsilon \geq |x_i|$, the comparison passes and to the cell α is added 1. When $\varepsilon < |x_i|$, the comparison refers us to the command k+4.

In both cases it is the command k+4, which transforms the command k+2, that is executed. One is added to the second address of command k+2 and then the address c+i becomes c+i+1. Once the command k+2 is transformed, the code of the command k+2 is compared with the code of the standard command resting in the cell k+8.

Once the code of command is a positive number, the addition of 1 from the second address to the cell increases this number. Consequently, as long as the second address of command k+2 will contain the number of cell c+i, less than the number of cell c+101, the comparison k+5 will return it to command k+2.

After the centuple execution of command k+2, the cell α will have the desired number. At t is, the command k+2 will assume the same appearance as the standard command stored in the cell k+8. Therefore, the command of comparison k+5 will let pass, the number that arose in the cell α will get printed and the machine will come to a stop.

Assuming, just as before, that the element x_i is placed in the cell c+i, we can, using the addition of commands, construct a program capable of selecting from among n numbers $x_1, x_2, \ldots, x_n$ the smallest one (as mentioned above, element x_i is located in cell c+i):

k + 1	П Ч	c + 1	—	a
k + 2	<	a	c + 2	k + 4
k + 3	П Ч	c + 2	--	a
k + 4	СК	k + 2	„1IIA"	k + 2
k + 5	СК	k + 3	„1IA"	k + 3
k + 6	<	k + 3	k + 8	k + 2
k + 7	Stop			
k + 8	П Ч	c + n + 1	-	a

In here, the command k+1 transfers the number x_1 into the cell α, whereupon x_1 is being compared with x_2. When x_2 is $\leqslant x_1$, then the command of comparison passes and x_2 is placed into the cell α. When x_2 is $> x_1$, the command k+4 will take over the control from the command k+2 and the cell will have the x_1. Hence, out of the numbers x_1 and x_2, the cell α will get the smaller one. Furthermore, the commands k+4 and k+5 transform the commands k+2 and k+3 in such a way, that at the second passage of the latter commands, the number resting in cell α will now be comparable with x_3. If and when the element x_3 is less than, or equal to that number, it will be turned over to cell α. That means: out of the numbers x_1, x_2, x_3 the smallest one will be formed in the cell α. Henceforth, the cell will have the smallest number out of the numbers x_1, x_2,....., x_n.

Introduction at a desired time of command k+6, that compares the variable command k+3 with the standard command k+8, brings the process to a halt. For sake of brevity we have spoken about the comparison of commands, whereas, actually not the commands themselves, but their codes, were being compared.

The program would be only slightly more complicated if we should set an additional task: to have in the cell δ the series of columns which are used for registering the third address of the command, an address c+i of the smallest element x_i. In the former program this smallest element is held in cell ; for singling out a cell with the smallest element x_i, we shall make use of the operation of logical multiplication. The command k+3 transfers the element x_i from the cell in which it is stored to cell α. If we logically multiply the command k+3 by the number

$$\underbrace{000000}_{6}\ \underbrace{11\ldots1}_{12}\ \underbrace{00\ldots0}_{12}\ \underbrace{00\ldots0}_{12} = 11\ldots1IA$$

and place the product into cell δ, we shall have there, in place of the first address, the c+i address of that element, which at that

STAT

time is being transferred to cell ω. The content of cell δ shall be shifted by 24 columns to the right. This shift is made at the end of the program, in order to save time and effort.

k + 1	Pch	c + 1	-	a
k + 2	∧	k + 1	„11...1IA"	δ
k + 3	<	a	c + 2	k + 6
k + 4	Pch	c + 2	-	a
k + 5	∧	k + 4	„11...1IA"	δ
k + 6	CK	k + 3	„1IIA"	k + 3
k + 7	CK	k + 4	„1IA"	k + 4
k + 8	<	k + 4	k + 11	k + 3
k + 9	→	δ	24	δ
k + 10	Stop			
k + 11	Pch	c + n + 1	-	a

It is now evident, that if the sequence $x_1, x_2, \ldots, x_n$ has several (equal among themselves) of the smallest elements $x_i, x_j, \ldots, x_k$, then the cell will have in it the newly formed address of the element having the greatest index from i, j, ..., k.

3-6. Samples of more complex problems.

Let us examine an example containing several cycles, which are performed in the program alternately. Suppose we have got to compute the value of functions sh x and ch x at some point x, by means of expansion in the series:

$$\text{sh } x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots,$$

$$\text{ch } x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots$$

For this, we can construct a program computing the general member $\frac{x^n}{n!}$ (n = 0, 1, 2....) of both series and adding the members with uneven powers into some cell β. Firstly, this program at the same time computes the value of both functions and, secondly, it is bound to be considerably shorter than two separate programs, one of which would compute sh x, the other ch x. Thus, sh x will be formed in cell β, and ch x in cell β_1. In the course of computation, we shall form in cell γ + 1 quantity n, in cell γ + 2 quantity $\frac{x^n}{n!}$.

In order to bring about the alternate addition of members $\frac{x^n}{n!}$,

one time into cell β , and another time into cell β_2 , we can use the following simple way: into a cell - let us name it controlling cell - we preliminarily place 1. Every time, after computation of the next number $\frac{x^n}{n!}$, the content of the controlling cell is multiplied by -1 and afterwards is compared with 0. In as far as in cell δ we have an alternate formation of 1 and -1, the comparison command will either let us pass (in this case we shall sum up the members with uneven powers), or it will send us off to another part of the program (where we shall form up the sum of members with even powers).

The computations will be stopped as soon as the next member in the row becomes less than, or equal to some quantity ε , which determines the precision of the computation.

Figure 3-2 shows the program schematically. In it, the number stored in cell δ is designated as δ .

Program of computation sh x and ch x

I	k + 1	P.Ch.	„0"	-	β_1	0; in β_1 is formed sh x
	k + 2	P.Ch.	„1"	-	β_2	1; in β_2 is formed ch x
	k + 3	P.Ch.	„0"	-	$\gamma+1$	0; in $\gamma+1$ is formed n
	k + 4	P.Ch.	„1"	-	$\gamma+2$	1; in $\gamma+2$ is formed $\frac{x^n}{n!}$
	k + 5	P.Ch.	„1"	-	δ	1; δ — controlling cell
II	k + 6	+	„1"	$\gamma+1$	$\gamma+1$	1 $\quad$ 2 $\quad$ 3
	k + 7	×	a	$\gamma+2$	$\gamma+2$	x $\quad$ $\frac{x^2}{1}$ $\quad$ $\frac{x^3}{2!}$
	k + 8	:	$\gamma+2$	$\gamma+1$	$\gamma+2$	$\frac{x}{1}$ $\quad$ $\frac{x^2}{2!}$ $\quad$ $\frac{x^3}{3!}$
	k + 9	|<|	$\gamma+2$	„ε"	k + 16	
III	k + 10	×	„−1"	δ	δ	−1 $\quad$ + 1 $\quad$ −1
	k + 11	<	„0"	δ	k + 14	
IV	k + 12	+	$\gamma+2$	β_1	β_1	$\frac{x}{1}$ $\quad$ $\frac{x}{1!} + \frac{x^3}{3!}$ sh x
	k + 13	PU	k + 6			
V	k + 14	+	$\gamma+2$	β_2	β_2	$1 + \frac{x^2}{2!} \quad \ldots$ ch x
	k + 15	PU	k + 6			
VI	k + 16	Stop				
	a	x				

The next example demonstrates a method of program-plotting, that is very often applied in solving of various problems. Let a program be constructed for computation of value of function th x in n points x+h, x+2h, ..., x+nh.

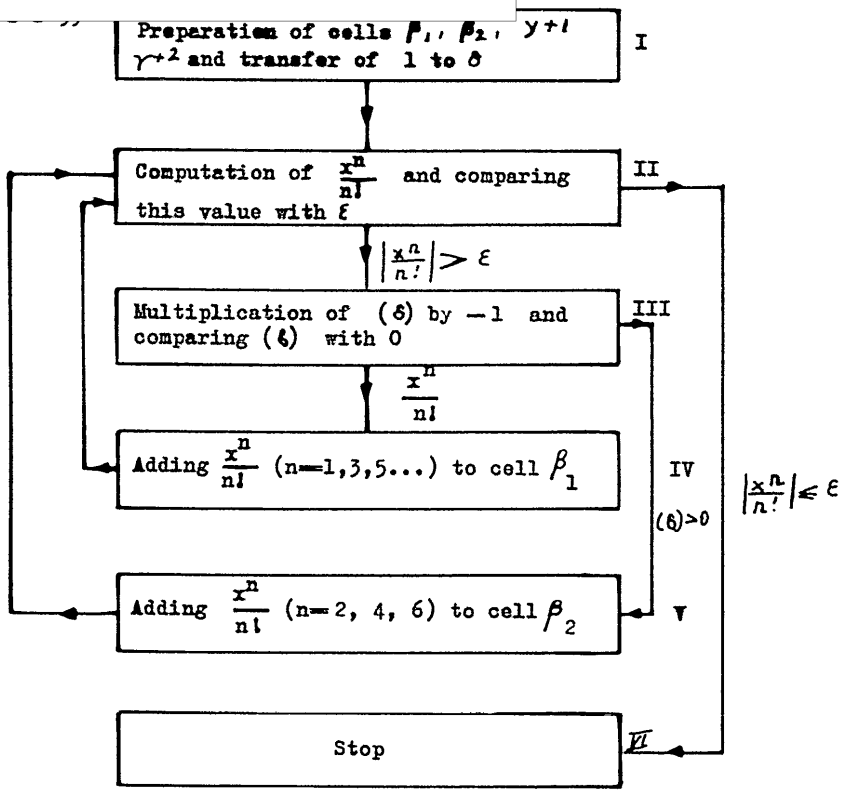


Figure 3-2.

Since

$$\text{th } x = \frac{\text{sh } x}{\text{ch } x} ,$$

the program shall include the following stages of computation:

1. Formation of next argument x+ih and the addition of 1 into the counter that determines number i of the tabulated values of the argument.

2. Computation sh(x + ih) and ch(x + ih).

3. Division of the found values sh(x + ih) and ch(x + ih) by each other, printing the result and comparing i with n.

4. If $i \leqslant n$, then the control is transferred toward the beginning of the program, if $i > n$, then the machine comes to a stop. How, the block diagram of the program will look like it is shown in Figure 3-3.

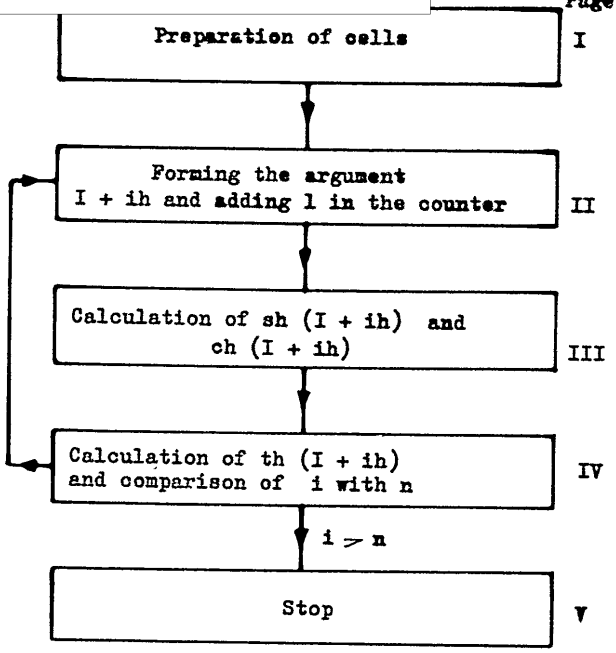


Figure 3-3.

If we attach to every group of commands I - V the name "operator", we shall see, that the operator III fully coincides with the program of computation of sh x and ch x, in our possession.

Let us, for sake of brevity, name the program that computes the value of th x "the basic program" and the one that computes sh x and ch x "the subprogram". Including the subprogram into the basic program, we must foresee the following:

Firstly, by plotting the basic program, we must take into consideration the way of distribution of points within the subprogram, that is, to take into account in which cells is stored the subprogram itself, in which cells rests the argument x (cell α) and where are being formed sh x and ch x (cells β , β_1 , etc.).

Secondly, at the end of the operator II, we must introduce a command that switches the count over to the beginning of the subprogram, i.e. to cell k+1. At last, namely thirdly, we must foresee the counting switch-over from the end of the subprogram to the beginning of the operator IV. Should the subprogram have already been introduced into the memory device of the computor, then the preparation of such switch-over can be performed automatically, leaving it to the basic program.

Therefore, the command that replaces the last command of the subprogram (operation "Stop" in cell k+16) by a command returning the control to the interrupted place of the basic program, should be placed before the command transferring the control from the

operator II to the subprogram. Such a command must be constructed beforehand and introduced into the memory device, together with the basic program. Now we have the following picture (the basic program is placed into the cells with numbers p+1, p+2,...., and the subprogram is placed into the cells with numbers k+1, k+2,....):

I	p + 1	PCh	"0"	-	γ+3
	p + 2	PCh	"x"	-	a
II	p + 3	+	"1"	γ+3	γ+3
	p + 4	+	"h"	a	a

The γ + 3 contains the counter of number i of tabulated points.

Preparation of cell a , in which argument x + ih will be formed.

1	2		n
x+h	x+2h	...	x+nh

The next command will have to transfer the control to cell k+1 of the subprogram, but, prior to that, the content of cell ξ earmarked for accomodation of the command returning the count to the beginning of the operator III, must be transferred into cell k+16. This command can be expressed only after the construction of the basic program. The rest of the program looks as follows:

Instead of III	p +5	P Ch	ξ	-	k+16
	p +6	PU	k+1	-	-
IV	p +7	:	β_1	β_2	β_1
	p +8	Print	β_1	-	-
	p +9	<	γ +3	"n"	p +3
V	p +10	Stop			
ξ		PU	p+7	-	-

The control went back to the beginning of subprogram

The command transferring the control from the subprogram's end to the basic program

Now it is evident, that the control has to be transferred from the subprogram to cell p+7. Exactly such a command is put in cell ξ. At every move to a new point, the content of cell ξ is again transferred to cell k+16. But, this process can be limited to one occurrence, if we interchange the places of commands p+3 and p+5 and replace p+3 in the third address of command p+8 by p+4. The thus constructed program is correct, yet somewhat wasteful in regard to worktime involved. The placement of the command transferring the content of cell ξ to cell k+16 immediately before the command transfers the control to the subprogram, was premeditated, because such placement is necessarily unavoidable in case some subprogram should alternately be called upon from various points of the basic program.

This method has the name "the method of standard subprograms". It will be examined in more details in the following chapter.

3-7. Samples of programs for the machine M-3.

Paragraph 2-11 dealt with the basic characteristics of the universal digital computing machine M-3, and with its structure of codes and commands (Figure 2-42). The M-3 is a machine with a fixed comma (decimal point), capable of operating numbers of negative (less than one) moduli. In other words, the sequence of every number is considered to be zero and, therefore, is not represented in the machine altogether.

When, as a result of arithmetic operations, there arises by the modulus a number equal to, or larger than one, then the discharge circuit becomes overfilled and the machine comes to a stop. Therefore, numbers of moduli in excess of one must be represented in the machine on a certain scale. We shall return to the problem of choice of proper scale coefficients later. Now we shall recall some specific data of this machine's code. List of commands of the M-3 can be found in Supplement 1. The M-3 is a twin-address machine, whose commands look as follows:

Operation code	I address	II address

By the first address are stored the subtrahend, the item, the multiplier and the divisor, and by the second address the item, the multiplicand, the dividend and the minuend. The result of every operation (with the exception of operations with moduli of numbers) can be recorded into the memory device (without any complementary command, by the second address) and get printed. If the result is recorded by the second address, the number located therein is "choked-up". Should it be needed in subsequent operations, it should be "saved" in advance, through the transplacement to another memory cell.

In the M-3, the result of every operation is retained at the beginning of the next operation in the register of the arithmetic knot. Owing to this, the result of the preceding operation can be made use of in a subsequent operation by analogy with the number standing by the second command address, i.e. represent the item, the minuend, the multiplicand and the divident. In this case, the command performed by the machine becomes, as a matter of fact, a single-address. By using the result of preceding action in the capacity of a number, we reduce the number of resortings to the memory device (magnetic drum) and shorten the time needed for the execution of the command.

The result of an operation making use of the result of a preceding operation can, in turn, be or not be recorded by the second address. If not, then the second address must be occupied with zeros.

Exemplifying the program for the machine M-3, we shall, instead of the numerical codes of operations, use their symbols (see Supplement 1). At that, when the result of operation is not being recorded into the memory device, a comma must be put after the symbol of operation, for instance +,. Meanwhile, when the result of the preceding operation, retained on the register of the arithmetic knot, is made use of, then there is an arrow put before the operation's symbol, for instance ↓ x.

Every arithmetic operation can be performed with moduli of numbers. Yet, the result of such an operation can not be recorded into the memory device without the application of a complementary command. For example, the command

| - |	α	β

means, that the module of a number stored in cell α has been subtracted from the module of a number stored in cell β, without the recording into the memory device. Similarly the command

↓ | - |	α	-

signifies that the module of the number stored in cell α has been subtracted from the module of the result of the preceding operation, without the difference having been recorded into the memory device.

The operation of conditional transfer is carried out by the sign of the result of the preceding operation and the record

c+1	...	...	...
c+2	UP	α	β

signifies, that when the sign of preceding operation is minus (the result of execution of command c+1), the next command to be executed after the command c+2 is the command resting in the cell α; when the sign is plus, then the next command to be executed is that resting in the cell β. It should be noted, that after the execution of the command UP, the register of the arithmetical knot will retain the result of the operation that proceded this command.

The conditional transfer of control, contingent upon the result of comparison of two numbers, calls for the use of two commands:

|~|	"p"	"q"
UP	α	β

The command from the cell α will be due for execution next, when $|\text{"q"}| < |\text{"p"}|$, and the command from the cell β will be next for execution when $|\text{"q"}| \geqslant |\text{"p"}|$.

An algebraic comparison of numbers of the operation of conditional transfer must be preceded by subtraction. Let us observe, that when $a > 0$, then the difference

$$(-a) - (-a) = -0$$

will appear in the machine, bearing the negative sign. Therefore, when the signs of the numbers subjected to comparison are unknown, we, in order to determine the equality of numbers a and b, shall have to examine and establish the signs of differences a-b and b-a. It is important to keep this in mind, for the operation of conditional transfer checks only one sign of the result of preceding operation.

The M-3 machine has no separate printing operation, for every operation whose result is recorded by the second address, can then have that result printed. Such an operation, which includes the printing process will be designated by placement of letter Π (P) after the symbol of operation, for example, the command

$+\Pi$	α	β

denotes that the operation effects the addition of numbers situated in cells α and β, that the sum is recorded in cell β and, at the same time, is printed.

In view of the fact that the M-3 is a machine with a fixed comma (decimal point), performing all the arithmetical operations without normalization, it calls for no introduction of a special operation for transformation of commands, which, in this machine, can be accomplished by the usual operations of addition and subtraction. For example, the addition to any command of the number

$$1I,\ 1III = \underbrace{0,0\ldots0}_{6}\ \underbrace{0\ldots01}_{12}\ \underbrace{0\ldots01}_{12} = 2^{-18} + 2^{-30}$$

results in an increase of both addresses of that command by 1.

Let us, for example, examine the program of extracting the square root from number x, for this machine. The equation

$$y = \sqrt{x}$$

can be written down as $f(x, y) = y^2 - x = 0$ and can be solved with

STAT

UNCLASSIFIED

the use of Newton's iterational formula (see Chapter 9):

$$y_{n+1} = y_n \frac{f(x, y_n)}{f'_y(x, y_n)}$$

which leads to the following redurrent correlation:

$$y_{n+1} = \frac{1}{2}\left(y_n + \frac{x}{y_n}\right)$$

for determination of square root of number x. In the given formulae y_n and y_{n+1} designate the "n" and the "n+1" approximate values of quantity y. It is easy to prove that when

$$|y_{n+1} - y_n| < \varepsilon,$$

then

$$|y - y_{n+1}| < \varepsilon,$$

since

$$y_{i+1} - y_i = \frac{1}{2}\left(\frac{x}{y_i} - y_i\right) =$$

$$= R(x, y_i),$$

$$y_{i+1} = y_i + R(x, y_i),$$

it would be sufficient to construct a program computing the quantity $R(x, y_i)$ and comparing its modulus with ε. When $|R| > \varepsilon$, then $R(x, y_i)$ must be added to y_i, whereupon the computation of $R(x, y_{i+1})$ must be made. Conversely, when $|R| \leq \varepsilon$, then the computation can be discontinued. For insuring the universal character of the program, a maximal number capable of being reproduced by the M-3 should be taken as a zero approximation, in order to preclude the relation $\frac{x}{y_o}$ formed at the first stage of computation from going out beyond the expanse of the columns. At the same time, should the x be small, such a choice would require a great number of iterations. Practically, in order to save time, the choice of a zero approximation of y_o must be harmonized with the quantity of number x. Besides, the requisite that $x < y_i$ must be implemented, which is easily admissible on account of $\sqrt{x} > x$ at $x < 1$. The program for the extraction of the square root will be as follows:

Extraction of square root.

k + 1	PCh	„y_0"	β	y_0
k + 2	:;	β	a	$\frac{x}{y_0}$ $\frac{x}{y_1}$
k + 3	↓ −,	β	−	$\frac{x}{y_0} - y_0$ $\frac{x}{y_1} - y_1$
k + 4	↓ ×	„ $\frac{1"}{2}$	γ	$\frac{1}{2}\left(\frac{x}{y_0} - y_0\right)$ $\frac{1}{2}\left(\frac{x}{y_1} - y_1\right)$
k + 5	↓ +	β	β	$y_0 + \frac{1}{2}\left(\frac{x}{y_0} - y_0\right)$ $y_1 + \frac{1}{2}\left(\frac{x}{y_1} - y_1\right)$ ...
				$-y_1$ $-y_2$
k + 6	(−I ,	γ	„ε"	Print $\sqrt{x}$
k + 7	ω P	k + 2	k + 8	
k + 8	+ n	„0"	β	
k + 9	Stop			
a	x			

Here, cell α holds the number x, the expression $R(x, y_i)$ is formed in cell γ, whereas the answer accumulates in cell β.

The result of action is not recorded in the commands k+2, k+3 and k+6. Therefore, the numbers standing in the second address of these commands experience no choking-up and can be made use of in the next passage of the cycle. The result of command k+4 is recorded into the cell by the second address, since it will be required for comparison with ε (command k+6). But in the next operation (command k+5), at the addition of number $\frac{1}{2}\left(\frac{x}{y_i} - y_i\right)$ to the contents of cell β., that number, for speed-up of action, is taken not out of cell γ, but from the register, where it was retained.

Constructing the above program of the extraction of square root from number x_0 we assumed that $x < 1$. On the strength of this, all numbers obtained in the process of computation are likewise less than 1 and, therefore, do nottrespass the limits allowed for by the M-3. In the next example we shall operate with numbers that by their moduli do exceed 1. Let us compose a program of computation of value of function e^x for some meaning $|x|$, $|x|<1$, employing the method of expansion into the series

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots.$$

Here $e^x > 1$. In Paragraph 3-3, the n-th member of this series was obtained out of the preceding member, by dividing that member

by n and multiplying by x. But in this case, this cannot be done and we shall have to do the following: In place of 1, the machine holds number $\frac{1}{M}$, where M constitutes a relatively large scale coefficient, and in place of numbers 1, 2, 3,..., by successive addition of $\frac{1}{M}$, are created numbers $\frac{1}{M}$, $\frac{2}{M}$, $\frac{3}{M}$ one from another. The member of the series $\frac{x^n}{n!}$ arises out of the foregoing $\frac{x^{n-1}}{(n-1)!}$ through its multiplication by $\frac{x}{M}$ and division by $\frac{n}{M}$. In that way, all numbers formed in the process of computation turn out to be less than 1, by the module. The program assumes the following structure:

Computation of e^x, $|x|<1$

k + 1	PCh	„1/M"	β
k + 2	PCh	„1/M"	γ + 1
k + 3	PCh	„0"	γ + 2
k + 4	×	a	γ + 1
k + 5	+	„1/M"	γ + 2
k + 6	:	γ + 2	γ + 1
k + 7	×	„1/M"	γ + 1
k + 8	+	β	β
k + 9	¬	„ε/M"	γ + 1
k + 10	UP	k + 11	k + 4
k + 11	+P	„0"	β
k + 12	Stop		
a	x		

The program is simple and calls for no special explanations. Let us only note, that in the command k+7 the member $\frac{x^n}{n!}$ is multiplied by 1/M. This is so, in order to be able to sum up all these members in the cell β, which forms e^x/M.

The command k+11 performs the printing of the result, i.e. the number e^x/M. As the machine M-3 has no separate operation for printing, so a zero is added to the content of cell β in the k+11 command and the sum is then printed.

Now we shall examine a program in which not the contents of cells are transformed at the next passage of the cycle, as it was the case with the two foregoing programs, but where the commands themselves undergo transformations. Let us compute the value of polynomical

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

for the value of some argument x. For simplification of the program

we may apply the method of computation introduced by Gorner, where the polynomical is represented by

$$P(x) = \{\ldots[(a_n x + a_{n-1})x + a_{n-2}]x + \ldots + a_1\}x + a_0 .$$

The computation of polynomical will now consist of consecutive addition into some cell β of the polynomical's coefficients (starting off with the senior one) and multiplication of contents in β by x. In order that we can introduce into the machine the coefficients a_i and argument x, they must be divided by a number M, so that $\left|\frac{x}{M}\right|$, $\left|\frac{P(x)}{M}\right|$, $\left|\frac{a_i}{M}\right|$ and all the interim results of computations should be less than 1.

Argument $\frac{x}{M}$ goes into cell α, coefficients a_n/M, a_{n-1}/M, ..., a_o/M go into cell p+0, p+1, ..., p+n respectively, whereas the result accumulates in cell β.

Computation of polynomial

k + 1	PCh	p + 0	β	$\frac{a_n}{M}$	
k + 2	$\times_9$	a	β	$\frac{a_n x}{M^2}$	$(a_n x + a_{n-1})\frac{x}{M^2}$
k + 3	÷,	1" / M	—	$\frac{a_n x}{M}$	$(a_n x + a_{n-1})\frac{x}{M}$
k + 4	+	p + 1	β	$\frac{a_n x}{M} + \frac{a_{n-1}}{M}$	$[(a_n x + a_{n-1})x + a_{n-2}]\frac{1}{M}$ ... Print P(x)
k + 5	+	"1IA"	k + 4		
k + 6	⊢,	k + 4	Y		
k + 7	UiP	k + 8	k + 2		
k + 8	+P	"0"	β		
k + 9	Stop				
	+	p + n	β		

a	$\frac{x}{M}$
p + 0	$\frac{a_n}{M}$
p + 1	$\frac{a_{n-1}}{M}$
....	....
p + n	$\frac{a_o}{M}$

Here the command k+5 forms in the first address of the command k+4 the numbers of the cells p+2, p+3, ... Consequently, the command k+4 will be successively adding into cell β the coefficients a_{n-1}/M (the coefficient a_n/M has been already put into cell β by the command k+1). In order that the multiplier $1/M$ in the first power could enter all items that are being accumulated in cell β, the result of the preceding action in the command k+3 is divided by $1/M$. The control of computation is performed by comparing the variable command k+4 with the standard command written down in the cell γ. As soon as a number of the cell p+n+1 is formed in the first address of command k+4, the difference $(\gamma) - (k+4)$ becomes negative, the control is turned over to cell k+8 and the machine comes to a stop. (We must recall, that the number stored in cell α is designated as "α"). The computation ceases only then, when the first address of command k+4 is occupied by the number of cell p+n+1, though the last coefficient a_o is stored in the cell p+n. This is so, because the command k+4 undergoes transformation only after it was executed. Consequently, as long as its first address contains the number p+n+1, no execution of command k+4 takes place. This circumstance deserves particular attention, because it often gives rise to misunderstandings and errors.

The problem can be made more complicated and then will call for a program of computation of value of polynomical $P(x)$ in the (m+1)-th points $x_o, x_1, \ldots, x_m$, differing from one another by the same quantity Δx. For this it is sufficient to have the above described program executed m+1 times, every time increasing the argument by the quantity $\Delta x/M$. For counting the number of passed points $x_1, x_2, \ldots$, not 1 should be added into the counter, as we did in Paragraph 3-6 (this cannot be done now), but the unity of the last column, i.e. $1/2^{30}$, whereupon the contents of the counter should be compared with the number $\frac{m+1}{2^{30}}$. The unity of the last column has been taken here, for the sake of certainty. The same result can be secured by adding into the counter 1 of any i-th column, i.e. the number $1/2^i$ and comparing the counter's contents with the number $\frac{m+1}{2^i}$, provided that the last number does not go out beyond the network of columns. One important remark: it was found before, that by the time of completion of computation of value of polynomical $P(x)$ at a point x the first address of command k+4 contained number p+n+1. But in order to be able to use this program for computation of the polynomical at the next point x, the first address of command k+4 must again contain the number p+1. In other words, the initial structure of the variable command k+4 must have been reconstructed prior to execution of computation of the polynomical at a next point. This can be accomplished by either of the two ways. The first way is of subtraction from the

command k+4 the number that was added to it in the course of the program's run. When we were computing the value of the polynomical at one point, we had n-times added to the first address of command k+2 a 1 of the first address. Consequently, altogether we had added a number $n \cdot \frac{1}{2^{18}}$. Also, in order to have reconstructed the initial structure of command k+4, it is sufficient to have executed the command

—	"$n\frac{1}{2^{18}}$"	k + 4

A program applying this method will be as follows:

k + 0	PCh	"2^{-30}"	γ + 1	Preparation of counter
k + 1	. . . . .	. . . . .	. . . . .	These commands are the same as in the foregoing program and for this reason are not shown here
. . . . .	. . . . .	. . . . .	. . . . .	
k + 8	. . . . .	. . . . .	. . . . .	
k + 9	-	"$n2^{-18}$"	k + 4	Reconstruction of command
k + 10	+	$\frac{\Delta x}{h}$	a	Switch-over to the next value of argument
k + 11	+	"2^{-30}"	γ + 1	Addition into the counter of the last column's 1.
k + 12	|—|,	+ 1	$\frac{m+1}{2^{30}}$	Check of number of tabulated points
k + 13	UP	k + 14	k + 1	
k + 14	Stop			

Another method of reconstruction of the variable command includes the following: The original structure k+4 is written out in some γ +2 cell:

γ +2	+	p+1	β

The command k+4 can be simply reconstructed by switching-over the content of cell γ +2 into cell k+2. Consequently, the program applying this method will be similar to that described above, with the only difference being that the command k+9 will now appear as

k+9	PCh	γ + 2	k + 4

By the way, it is more expedient to have this command placed not after, but before the passage of the cycle. We have purposely dwelt upon the reconstruction of a "spoiled" program, because this

matter has a great significance in the plotting of concrete mathematical problems. We shall return to this subject in the next chapter.

3-8. Conversion of numbers from decimal to binary system of computation and vice versa.

Herein we shall examine the process of conversion of decimal system of calculation to binary and vice versa, in application to the machine M-3.

Let us have a positive decimal number

$$A = 0, \alpha_1, \alpha_2, \ldots\ldots \alpha_n,$$

that is less than one. The α_i is a decimal figure, assuming one of the values 0, 1, 2, ...9. We know that

$$A = \frac{\alpha_1}{10^1} + \frac{\alpha_2}{10^2} + \ldots + \frac{\alpha_n}{10^n} .$$

Let us represent every figure α_i in the form of four-column binary number $\beta_i^I \beta_i^{II} \beta_i^{III} \beta_i^{IV}$. We get a binary 4n-column number

$$A' = 0, \beta_1^I \beta_1^{II} \beta_1^{III} \beta_1^{IV} \beta_2^I \beta_2^{II} \beta_2^{III} \beta_2^{IV}, \ldots\ldots, \beta_n^I \beta_n^{II} \beta_n^{III} \beta_n^{IV}$$

If, for instance, A = 0,38109, then the binary number A' will be

$$A' = 0, \underbrace{0011}_{3} \ \underbrace{1000}_{8} \ \underbrace{0001}_{1} \ \underbrace{0000}_{0} \ \underbrace{1001}_{9}$$

The number A' is called the binary-decimal code of number A and, self-evidently, is not equal to it, but with its help, we can find the binary reading of the A.

That reading is:

$$A = \frac{\alpha_1}{16} + \frac{\alpha_2}{16^2} + \ldots + \frac{\alpha_n}{16^n} .$$

In order to create a binary reading of the number A, it is sufficient to multiply $\frac{a_i}{16^i}$ by $\frac{16^i}{10^i}$ and then have the obtained products for i = 1, 2, ..., n. added together. Certainly, all numbers $\frac{16^i}{10^i}$ must appear in binary notation and all actions be made by the binary system of calculation.

So we get a number

$$A = \sum_{i=1}^{n} \frac{a_i 16^i}{16^i 10^i} = \sum_{i=1}^{n} \frac{a_i}{10^i} ,$$

which is expressed by the binary system.

STAT

Separation of number $\frac{a_i}{16^i}$, out of the binary code A' can be accomplished merely by logical multiplication of A' by the 4n-column binary number

$$p = 0,\ \underbrace{00....01111\ 00....00}_{4i}$$

Inasmuch as the below-mentioned program is constructed for use by the machine M-3, seven columns must be retained in the decimal number $A = 0,a_1 a_2, ...,a_n$. Indeed, the binary-decimal code of number A contains 4n-columns and the cell of this machine's memory device contains 30 columns. If and when number A contains a lesser number of columns, then the lacking columns must be replaced by zeros. For example, the binary-decimal code of the above-mentioned number A = 0,38109 is

$$0,\ \underbrace{0011}_{3}\ \underbrace{1000}_{8}\ \underbrace{0001}_{1}\ \underbrace{0000}_{0}\ \underbrace{1001}_{9}\ \underbrace{0000}_{0}\ \underbrace{0000}_{0}\ 00$$

Actually, at first the $\frac{a_7}{16^7}$ is separated by way of logical multiplication of number A' by

$$p = 0,\ \underbrace{00....01111}_{28}\ 00$$

which in its 25-28 columns, has numbers one and in other columns zeros. After this, $\frac{a_7}{16^7}$ is divided by $\frac{10}{16^7}$ (we cannot multiply it by $\frac{16^7}{10}$ since $\frac{16^7}{10} > 1$). Now we have a number $\frac{a_7}{10}$. For separation of columns designating the figure a_6, it is sufficient to have number A' shifted by four columns to the right and multiply it by number p. Inasmuch as the machine M-3 has no shift operation, it is substituted by multiplication by 1/16 and then we have number $\frac{6}{16^7}$. If we divide it by $\frac{10}{16^7}$ we obtain a number $\frac{6}{16^7}$. All that is left to be done now, is to add it to $\frac{a_7}{10}$, having multiplied the latter beforehand by $\frac{1}{10}$. Having performed this operation five times, we can get the final number A expressed in the binary notation. This program is so simple, that we do not even bring it up.

Program of conversion of numbers from the decimal to the binary system of calculation for the Machine M-3.

Cell number	Operation	IA	IIA	Remark
k+1	PCh	a+9	a+2	"Clearing" of cell a+2, which accumulates answer
k+2	PCh	a+9	a+4	"Clearing" of counter
k+3	x	a+5	a+2	Multiplication of $\frac{a_n}{10^{n-i}} + \frac{a_{n-1}}{10^{n-i-1}} + \ldots + \frac{a_{i+1}}{10}$ by $\frac{1}{10}$ (at the first stage this operation is unnecessary)
k+4	Λ,	a+1	a+3	Separation of $\frac{a_i}{16^7}$
k+5	↓:,	a+8	-	Formation of $\frac{a_i}{10}$
k+6	↓+	a+2	a+2	Accumulation of answer
k+7	x	a+6	a+1	Shift of number subjected to conversion by four columns to the right
k+8	+	a+7	a+4	Addition of 1 into counter
k+9	↓|−|,	a+10	-	Comparison of counter's content with 7
k+10	PCh	k+3		

a+1	Binary-decimal code of converted number
a+2	Formation of binary notation of converted number
a+3	p—0, $00\ldots0^{24}111100$
a+4	Counter
a+5	$\frac{1}{10}$ — 0.0001100110011001100110011001100110
a+6	$\frac{1}{16}$ —0,000100,..0
a+7	$\frac{2}{16}$ —0,00100...0
a+8	$\frac{10}{16}$ —0,$00\ldots0^{28}101000$
a+9	0—0,00...0
a+10	$\frac{13}{16}$ —0,110100...0

For realization of this program, the binary-decimal code of number A is perforated in the tape (see Paragraph 2-8) and is put into cell a+1. The address of the cell which after the completion of conversion is supposed to take over the control, must be put into

the second address of command k+10. The binary recording of the number takes place in cell a+2.

Let us now consider a reverse operation, the way of conversion of positive binary number

$$A = 0, \quad \beta_1 \beta_2 \ldots \beta_n$$

to decimal designation. In the decimal system of calculation, the number A appears as

$$A = 0, \quad \alpha_1 \alpha_2 \ldots \alpha_n = \frac{\alpha_1}{10} + \frac{\alpha_2}{10^2} + \ldots + \frac{\alpha_n}{10^n},$$

where the numbers α_i are not yet known and must be ascertained. For this we multiply number A by $\frac{10}{16}$ and get

$$\tilde{A} = \frac{\alpha_1}{16} + \left(\frac{\alpha_2}{10} + \frac{\alpha_3}{10^2} + \ldots + \frac{\alpha_n}{10^{n-1}}\right)\frac{1}{16}.$$

Inasmuch as the number in parentheses is less than one, the second item is placed in the column's beginning with the fifth to the n-th, whereas the $\frac{\alpha_1}{16}$ is located in the first four columns of number A. Separation of these columns results in definition of the binary expression of figure α_1 of the first decimal column of number A. In order to find out column α_2, we must subtract number $\frac{\alpha_1}{16}$ from number $\tilde{A}$, to obtain

$$\tilde{A} = \left(\frac{\alpha_2}{10} + \frac{\alpha_3}{10^2} + \ldots + \frac{\alpha_n}{10^{n-1}}\right)\frac{1}{16},$$

having multiplied it by $\frac{10}{16}$, we get:

$$\tilde{A} = \frac{\alpha_2}{10^2} + \left(\frac{\alpha_3}{16^2} + \ldots + \frac{\alpha_n}{10^{n-2}}\right)\frac{10}{16^2},$$

Now the columns from 5 to 8 are presently occupied by number $\frac{\alpha_2}{16^2}$ and the second item is located to the right from the eighth column. The binary expression α_2 of the second decimal group of number A can now be found out by separation of columns 5-8. Repetition of this operation n times produces the binary expression of all n columns $\alpha_1, \alpha_2, \ldots \alpha_n$, i.e. the binary-decimal code of number A.

In the M-3 machine, only 28 senior columns can be used by the binary-decimal code, i.e. n = 7. But it can occur that from some place all columns of the number undergoing conversion are equal to zero. For elimination of unnecessary cycles, calculation is stopped as soon as the difference between $2^{-29} + 2^{-30} = 0, \underbrace{00\ldots.011}_{30}$ and the number undergoing conversion becomes positive, i.e. as soon as all the remaining columns of the number undergoing conversion with

exception of its last two columns (29th and 30th) become equal to zero.

Program of conversion of numbers from the binary to the decimal system of notation for the machine M-3

k+1	PCh	a+7	a+2	Clearing of cell a+2
k+2	PCh	a+3	a+8	Transfer of number p = 0, 111100...0 to work cell a+8
k+3	x	a+5	a+1	Multiplication $\left(\frac{a_{i+1}}{10} + \ldots + \frac{a_n}{10^{n-i}}\right)$ by $\frac{10}{16}$
k+4	↓∧	a+8	a+9	Separation of $\frac{a_i}{16^i}$ to work cell a+9
k+5	↓ +	a+2	a+2	Accumulating the answer
k+6	↓ −	a+1	a+1	Separation of $\left(\frac{a_{i+1}}{10} + \ldots + \frac{a_n}{10^{n-i}}\right)\frac{1}{16^i}$
k+7	x	a+4	a+8	Shift of number p by 4 columns to the right
k+8	| −|	a+1	a+6	Comparison of number being converted with number $2^{-29} + 2^{-30}$
k+9	UP	k+3	-	

a+1	Binary notation of number subjected to conversion
a+2	Formation of binary-decimal code of number subjected to conversion
a+3	p = 0,111100....0
a+4	$\frac{1}{16}$ = 0,000100....0
a+5	$\frac{10}{16}$ = 0,101000....0
a+6	$2^{-29} + 2^{-30}$ = 0,00....011
a+7	0 = 0,00....0
a+8	Work cell for storing 0, $\underbrace{0....011110....0}_{4i}$
a+9	Work cell for storing $\frac{a_i}{16^i}$

3-9. Separation of integral and fractional parts in machines with floating and fixed commas.

The significance of separation of the integral part will be explained at examination of the problem of computation of function e^x for large values of x. As we know, the series

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots$$

fits for every value of x, but when x is sufficiently large, we practically have to sum up a too large number of members before we arrive at a stage where the remainder of the series becomes less than the prescribed error. Let us apraise, for instance, at which n the inequality

$$\left|\frac{x^n}{n!}\right| < 0{,}01 ,$$

if $x = e^5 \approx 146$. According to Stirling's formula

$$n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$$

n has to concur to the equation

$$(e^5)^n = 0{,}01 \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$$

$$e^{5n} = 0{,}025 \left(\frac{n}{e}\right)^n \sqrt{n}$$

or after logarithming

$$5n = n \lg n - n + \tfrac{1}{2} \lg n + \lg 0{,}025 ; \quad 12n = (2n + 1)\lg n - 7.4 .$$

Assuming than $n = e^6 = 396$, we shall in the last left equation have $12e^6$ and in the right one $12e^6$, which leads to the conclusion, that we should apply about 400 members of the series to have a member $\frac{x^n}{n!}$ with a modulus less than 0,01, when x = 146. Consequently, should we try to find out e^x using the method of expansion into the series, we should, at large values of argument x, perform a huge number of operations, which would result in a great waste of time and effort. Meanwhile we know, that the plotting of programs has to be done in a most expedient and time-saving way. The longer the machine is put to operation, the more probable would be the presence of errors. Besides, the great cost of contemporaneous electronic machines makes it imperative, that they should be made use of as efficiently as possible. These considerations force us to look for another, quicker way of computation of e^x.

For this purpose we may employ the method of separation of the integral part and have the argument x in the form of

$$x = [x] + \{x\} ,$$

wherein $[x]$ is the integral number and $\{x\}$ is the proper fraction.

Now $e^x = e^{[x]+\{x\}} = e^{[x]}e^{\{x\}}$.

The multiplier $e^{\{x\}}$ is calculated quickly by way of expansion into the series, as $(x)<1$ and, consequently,

$$\frac{\{x\}^n}{n!} < \frac{1}{n!},$$

i.e., already at n = 10

$$\frac{\{x\}^n}{n!} < \frac{1}{3\cdot628\cdot800} \approx 2.7\cdot10^{-7}$$

(it is known that the series $1 + \frac{1}{1!} + \frac{1}{2!} + \ldots$ tallies very quickly). There remains to have the quantity $e^{\{x\}}$ multiplied by e (or by $\frac{1}{e}$ if $[x]<0$) exactly $[x]$ times, which is very simple because $[x]$ is an integer. In our hypothetical machine we can expand the argument into an integral and fractional part using one command (see Paragraph 1-7), but this can not be done in the machine M-3. The fact that in here the member $\frac{x^n}{n!}$ is taken with the scale coefficient M, so that $\left|\frac{x^n}{n!}\right|\cdot\left|\frac{1}{M}\right| < 1$, does not change the situation, because this member, for ascertainment of the end of calculation, is compared not with ε but with ε/M. Consequently, it is necessary to separate the integral part of the argument x. This can be accomplished by the following: let x/M be stored in cell α and $x > 0$. Number $\frac{1}{M}$ is placed in some cell δ and then is consecutively subtracted from the content of cell α, until in cell α there remains a number less than $\frac{1}{M}$. At every such subtraction the content of cell δ is multiplied by e. By doing so, we get simultaneously the number $\frac{\{x\}}{M}$ in cell α and the number $e^{[x]}/M$ in cell δ. We shall get the answer after having calculated $e^{\{x\}}/M$ in accordance with the program described in the preceding Paragraph(using it as a subprogram) and multiplied the result by $e^{[x]}/M$.

p + 1	P Ch	$\frac{1}{Me}$	δ	$\frac{1}{Me}$				
p + 2	:	$\frac{1}{e}$	δ	$\frac{1}{M}$	$\frac{e}{M}$	...	$\frac{e^{[x]-1}}{}$	$\frac{e^{[x]}}{M}$
p + 3	—	$\frac{1}{M}$	a	$\frac{x-1}{M}$	$\frac{x-2}{M}$		$\frac{\{x\}}{M}$	$\frac{\{x\}-1}{M}$
p + 4	UP	p + 5	p + 2					
p + 5	↓+,	$\frac{1}{M}$	—					$\frac{\{x\}}{M}$
p + 6	↓ :	$\frac{1}{M}$	a					$\{x\}$
p + 7	P Ch	p +1 2	k +1-1					
p + 8	PU	k + l	$\frac{}{\beta}$					$\frac{e^{[x]}}{M}, \frac{\{x\}}{M}$
p + 9	X,	$\frac{\delta}{1}$						$\frac{e^{x}}{M}$
p + 10	↓ :	$\frac{1}{M}$	β					
p + 11	Stop							
p + 12	RU	p + 9	—					
a	x/M							

We shall have to say a few words to explain this program. We had to subtract 1/M from number x/M until the difference became less than 1/M (let us remember that $x > 0$). Meanwhile, the control by the first address can be transferred by the operation of conditional conversion of p+4 only then, when in cell α there arises a negative number. In view of this, at first we transfer into cell δ 1/Me (and not 1/M), whereas in the command p+5 we add 1/M to the content of cell α (the command of conditional transfer of control p+4 does not change the condition of the register of the arithmetical knot).

Separation of the integral part of the positive number x can be easily accomplished when we take the power of number 2 to serve as a scale M: $M = 2^k$. Then the integral part x will be recorded in the senior columns k of the cell and can be separated by way of logical multiplication by number

$$\underbrace{11\ldots1}_{k}\ 00\ldots0$$

At a long series of calculations, the selection of $M = 2^k$ as the scale is less convenient than selection of $M = 10^p$, since the division of x by scale M is made prior to introduction of number x into the machine.

The operation of separation of integral and fractional parts can be applied for calculation of value of function sin x, when x is large. Should we try to accomplish the calculation of sin x by way of expansion into the series, we should have to perform summations of a too large number of members. Therefore, we should better use the correlation

$$\sin(2k\pi + x) = \sin x$$

and represent the argument in the form of:

$$x = 2k\pi + x_o .$$

The simplest way of so doing is to have the argument x divided by 2π, then take the fractional part of this relation and have it multiplied by 2π:

$$x_o = \left\{\frac{x}{2\pi}\right\} 2\pi .$$

Once we have ascertained the value of x, we can perform the calculation of sin x_o with the use of the program described in Paragraph 3-3. It will be now as follows:

p + 1	:	a	"2π"	a
p + 2	[ ]	a	β	a
p + 3	X	a	"2π"	a
p + 4	PU	k + 1	—	—
a	x			

$$\frac{x}{2\pi}$$

In cell ∞ is created $\{x/2\pi\}$, in cell β is created $[x/2]$

$$x_o = \left\{\frac{x}{2\pi}\right\} 2\pi$$

Referring to the subprogram of calculation of sin x (Par. 3-3)

For the expedition of the calculation process it would not be sufficient to have the argument reduced by the modulus to a value less than 2π. Using some trigonometrical correlations, it might be useful to have the calculation performed so as to have the argument's value, in the value of the modulus, less than $\frac{\pi}{4}$, but we shall not dwell on this.

Other methods, related to those described in this paragraph can be likewise useful in the calculation of value of function lg x. The ordinary formula of expansion into the series

$$\lg(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots - 1 < x \leq 1$$

is absolutely unusable in this case, since, when the value of x is close to 1, the members of this series subside like $\frac{1}{n}$ and in order that this member could become less than, for example 10^{-4}, we should have to calculate 10^4 members. For reduction of the volume of cal-

culation work we could substitute x by -x and the have the formula:

$$\lg(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \ldots.$$

Upon subtraction herefrom of the foregoing formula, we should have

$$\lg \frac{1-x}{1+x} = -2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots\right)$$

This series is twice as quick because it contains exclusively members with uneven powers. Should we now have to calculate $\lg y$ for some value $y > 0$ and designate y as

$$y = e^p z$$

wherein p is an integer (positive, negative or zero) and

$$\frac{1}{e} < z \leq 1$$

assuming that

$$\frac{1-z}{1+z} = x, \quad 0 < x < \tfrac{1}{2},$$

then

$$\lg y = \lg e^p z = \lg e^p + \lg z = p + \lg z = p + \lg \frac{1-\frac{1-z}{1+z}}{1+\frac{1-z}{1+z}} =$$

$$= p + \lg \frac{1-x}{1+x} = p - 2\left(x + \frac{x^3}{3} + \ldots + \frac{x^{2n-1}}{2n-1}\right) -$$

$$- 2\left(\frac{x^{2n+1}}{2n+1} + \ldots\right) = p - S_n - \Delta_n;$$

$$\Delta_n = 2\left(\frac{x^{2n+1}}{2n+1} + \ldots\right) < \frac{2x^{2n+1}}{2n+1}(1+x^2+x^4+\ldots) = \frac{x^{2n+1}}{2n+1} \cdot \frac{2}{1-x^2}.$$

Since $\frac{2}{1-x^2} < 3$, then at $\frac{x^{2n+1}}{2n+1} < \frac{\varepsilon}{3}$ the remainder Δ_n in the value of the modulus is less than ε. In this case $|x| < \frac{1}{2}$, consequently at $n = 6$

$$\frac{x^{2n+1}}{2n+1} \leq \frac{1}{2^{13} \cdot 13} < 10^{-5},$$

that is, for accomplishing the same (or a still higher)degree of precision it is now sufficient to take only 6 members instead of 10^4 members.

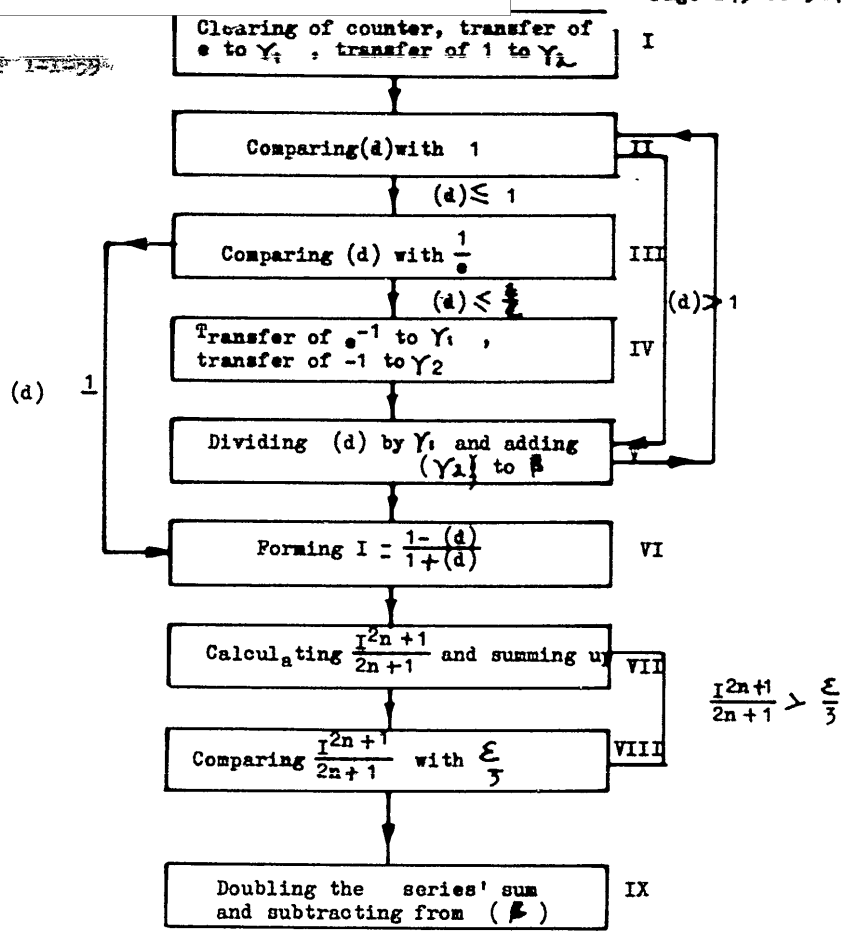


Figure 3-4

For the reduction of y to $e^p z$ (in order to determine numbers p and z) we can place y into cell α and then employ one of the following operations:

if $y > 1$, we divide the content of cell α by e, until the quotient becomes equal to or less than 1, accompanying every division by feeding number 1 into counter β ;

if $y \leq \frac{1}{e}$, we divide the contents of cell α by e^{-1} until the quotient becomes more than $\frac{1}{e}$, accompanying every division by feeding number -1 into counter β .

Doing so, we shall have z in cell α and number p in counter β . Then, by the formula

$$x = \frac{1-z}{1+z}$$

we determinate the number x, for which we separate the series

$$x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots$$

until the next calculated member becomes less than $\varepsilon/3$. The result is doubled and subtracted from p. By this way in

cell β is formed lg y. The logical diagram of this program is shown by Figure 3-4 (cell α contains number y).

Below is shown the basic part of this program (operators I-V).

I	k + 1	PCh	„0"	—	β
	k + 2	PCh	„e"	--	γ_1
	k + 3	PCh	„1"	--	γ_2
II)	k + 4	<	"1"	a	k + 8
III	k + 5	<	„$\frac{1}{e}$"	a	k + 11
IV	k + 6	PCh	„e^{-1}"	—	γ_1
	k + 7	PCh	„-1"	—	γ_2
V	k + 8	:	a	γ_1	a
	k + 9	+	γ_2	β	β
	k + 10	PU	k + 4	—	—
VI	k + 11	. . . . .	. . . . .	. . . . .	. . . . .
	. . .	. . .	. . .	. . .	. . .

FOURTH CHAPTER.

The Programming of Mathematical Problems.

4-1. Program circuit.

In the preceding chapter, using a number of examples, we demonstrated the basic methods of program-plotting. In this chapter we shall expound the programming methods and the formulation of mathematical problems with the use of computers.

As we have repeatedly noted before, in order to employ the computer for solving any desired problem, we should choose some numeral method, through which we might reduce the solving process to a series of arithmetical operations with numbers. Such choice is predicated by considerations of desired precision, operational speed, simplicity and capability of the given machine of coping with the set problem, and so on. In the majority of cases this is a difficult task, constituting an analytical approach of approximation. Although these factors do have an influence on the methods of programming, they have no immediate relation thereto and for this reason, we shall not dwell on them.

Also, in order to solve a set mathematical problem, we, shall we say have chosen a certain numeral method and now we have to construct a program that would be capable of executing this method in the machine. With this in mind, we must first of all try to conceive a clear picture of the whole process of calculation, i.e. explicitely establish, by which mathematical formulae the computation must be conducted, in which succession and under which circumstances should we employ one or another formula, how many times and for which numbers the given formula should be applied, and so on. Furthermore, we must determine which numbers should be taken out of the machine, and which numbers are needed merely for execution of subsequent calculations. If the latter include numbers needed in subsequent calculations, they must be retained in separate cells.

Constructing the calculation chart, we break the whole computation process down into a series of so-called operators of computation, everyone of which makes application of one or more mathematical formulae. As a rule, one such operator may contain such formulae, which are made use of simultaneously in the whole process of computation. Besides, one operator may employ such formulae which deal with the same numeral material, and therefore need the same information. If

two different mathematical formulae have, for instance, a comparatively extensive general part, then it can be separated into one operator. At the same time, such formulae which do not meet one another simultaneously in the course of computation, or are needed now together, now separately, must be contained in separate operators. Likewise, the formulae applied for determination of same quantities, which, however, depending on circumstances are in various situations applied individually, must also be contained in different operators. It is not simple to convey the meaning of the term "operator", it will be elucidated by examples brought up in the coming paragraphs.

We shall be designating the computation operators with capital Latin letters A, B, C,... at times assigning to them indices A_j, B_{jk}, which will signify that these operators change depending on some indices i, j, k, ...

Now we are able to describe the whole computation system with the use of operators, i.e. indicating which operators, in which succession, how often and depending on what conditions should be carried out so as to insure the full realization of the computation process. The preceding chapter provides some of the simplest examples of computation. For example, Paragraph 3-7 presents a program of computation of a polynomical by Gorner's method. If in that program we designate the operator that feeds the coefficient a_i into cell β and multiply the content of that cell by x, with A_i, then the whole calculation scheme will look as follows:

$$\prod_{i=0}^{n} A_i = A_0 \cdot A_1 \cdot \ldots \cdot A_n ,$$

wherein the sign $\prod_{i=0}^{n}$ denotes: the product of operators A_i for i = 0, 1, ..., n. The simplicity of this scheme is explained by our beforehand knowledge of the number of repetitions of the cycle during the calculation process. But, in Paragraph 3-3 we also examined such programs wherein the number of cycle repetitions was determined by the program itself. In this case, it is not sufficient for composition of a calculation scheme to have only calculation operators. It should as well contain the so-called logical operators, which check the fulfillment of some of the conditions usually brought about with the use of an operation of conditional transfer of control. For example, composing a calculation scheme for determination of e^{x_0}, we should have to introduce the calculation operator A, that calculates the member $\left|\frac{x_0^n}{n!}\right|$ and feeds that member into cell β, as well as the logical operator λ, comparing $\left|\frac{x_0^n}{n!}\right|$ with ε. Then, the computation

STAT

scheme will be as depicted on Figure 4-1

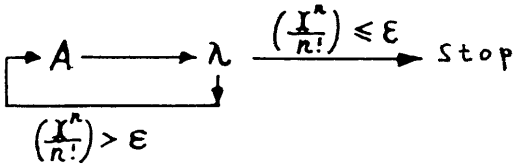


Figure 4-1.

These two examples alone make it now evident, that the calculation scheme does not exhaust the meaning of the whole program, for apart from commands needed for the implementation of calculation operators and logical operators, the program contains a series of other commands required for the preparation of cells, readdressing the commands, etc. In order to be in a position to describe the program in all details, we have to, on the basis of calculation scheme, construct a program's circuit. To put it bluntly, the calculation scheme differs from the program's circuit in that the latter describes the whole program, whereas the former encompasses only the arithmetical commands of that program. We may say, that the program's circuit, apart from strictly arithmetical operations, must describe everything else that pertains to the control over those operations. Consequently, while plotting a program's circuit, we shall have to introduce not only the calculation operators, but also a number of other operators. The following operators belong to the category of the most frequently encountered operators:

R_i - operator of reconstruction; it reconstructs the initial shape of those commands which depend on index i, and which undergo transformation in the calculation process.

r_i - operator for preparation of counter; it feeds into the counter the initial value of the index i.

F(ni) - operator of transformation or readdressing; it transforms the commands depending on index 1 by the n number of 1.

f(1) - operator which adds one to the content of the counter of index i.

λ_i - logical condition checking the quantity of index i.

T - operator of conveyance; out of the whole set of constants, this operator selects the needed group and feeds it into the work cells. It also prepares the cells whose contents undergo transformation during the program run, for example the work cells, the cells accumulating the answer, etc.

With the use of these operators we are able to construct a logical program's scheme. For example, designating the operator feeding a_i into cell β as T and the operator that increases the index i of operator A_i by one as Δ F(i), we shall have the program circuit for the calculation of polynomicals, previously examined in Paragraph 3-7, in a different form,as shown on Figure 4-2.

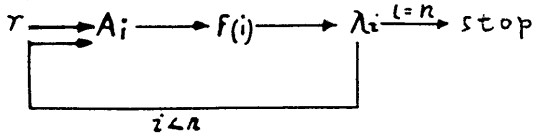


Figure 4-2.

The block-diagrams we used for description of the program in Paragraph 3 are more illustrative, yet they do not describe the program just as clearly and exactly as the logical operator schemes do. They will be dealt with in greater detail in the next paragraphs,but now we leave them alone and confine ourselves to making a few general remarks about the methods of problem programming.

Firstly, the construction of a logical program's circuit should be performed with the aim possibly shortening the time of calculation to the minimum, which, among other things, can be secured by a reduction of the number of commands in the program. Particular attention should be paid to the commands contained in the frequently repeated cycles. At times it can be expedient to reconcile oneself to some increase of the total number of commands in the given program, provided that this increase takes place at the cost of such portions of the program, which are to be executed but once or so and for that reduces the number of commands in the cycles.

Furthermore, let us note that before we get down to composition of an exact program circuit, we must have apportioned the capacity of the memory device, that is to say, allocate cells to all the constants, which either are contained in the conditions of the problem (determinant's elements, for example), or are required for its solution by the machine (the comparison constants, for example). Moreover, some cells must be allocated beforehand for quantities arising in the process of calculation (for example, when the determinant's elements are unknown, but we have formulae by which they will be calculated), for use as the work cells, etc. The work cells are those cells which store the results of intermediary calculations. At last we must allocate some cells for accommodation of the program itself. Distribution of memory mechanism, particularly important for composition of such operators as the operator of readdressing, calls upon certain experience and cannot be always made up exactly at once, since it is impossible to foretell exactly the total number of commands.

STAT

It can be efficiently made, after some practice, though. If it is clear from the beginning, that the whole program (including the constants, work cells, etc) cannot be placed within the internal memory unit of the machine, a part thereof is put into the exterior memory device. The time required for the solution of a problem will be reduced, if we manage to utilize the exterior memory device, as rare as possible.

In the Paragraph 3-7 we have already noted, that the reconstruction of commands undergoing transformation in the course of calculation can be accomplished by one of the two methods. Firstly: from the code of the given program we can subtract the number that had been added thereto in the course of the program's run. Secondly: we can keep the initial, the so-called standard structure of the program in a separate cell and feed it into where it is needed in the program. The second method can be particularly recommended, for its application diminishes the chance that either the program-plotter or the machine itself should commit an error. The original structure of the command can be executed at the very beginning of the program. In such a case, the machine can be stopped at any stage of program's run and we can resume the solution of the problem anew, beginning with the first command. The same is true when we have a malfunction. If, however, the standard commands are being fed in after their transformation, or in such a case when the reconstruction of commands is being performed by subtraction of added numbers, at every stop of the machine in the middle of the program run and at every malfunction, the whole program must be fed into the machine anew. The programs in which the reconstruction of commands precedes their transformation are usually called self-reconstructing programs. They are very convenient for use in solving of such problems which call for handling of a great number of variants. This method finds detailed illustration in Paragraph 4-3.

It is expedient to work out the program circuit by stages. The whole count is then broken down into enlarged operators and the general program circuit is then made up of those operators, whereupon separate circuits of every such enlarged operator are worked out. Sometimes such introduction of sub-circuits into the principal circuit can be multi-stage and considerably facilitates not only the working out of the circuit, but also provides a better survey of the whole circuit. This method is used in Paragraph 5-2.

The working out of the program can be conveniently accomplished in two steps: at first we make up a program of the type described in Chapter 3. That type includes the so to say symbolic programs, wherein the command addresses are shown conditionally as k+1, k+2, ... , the

STAT

work cells are substituted with cells α , β , γ , .., and the numbers themselves, in inverted commas (in quotation marks), appear in place of their addresses, and so on. Once we have worked out such a program we can get down to the final allotting of the capacity of the memory device. Such program can be encoded, put on perforated tape and fed into the machine, when required.

The majority of mathematical problems include such frequently arising processes as computation of roots, operations with complex numbers, calculation of values of special functions, etc. It would be inexpedient to work out such programs every time anew. Therefore, such processes use to be programmed beforehand and the machine is supplied with a library of such programs, which are usually called "the standard subprograms", on account of their independence from specific qualities of some concrete problem. Making up the program for the given problem, we can easily include into it the already available standard subprograms, just as we did in Paragraph 3-6. The carefully worked out standard subprograms prove to be very economical with respect to the time of the machine's operation.

4-2. <u>The program for solution of common differential equations by the Runge-Kutta method.</u>

An examination of the ways of finding a solution of a common differential equation by the Runge-Kutta method, can serve as an example illustrating the methods set forth in the preceding paragraph.

Let us find a solution to equation

$$\frac{dy}{dx} = \Phi(x, y) ,$$

satisfactory to the conditions that

$$y(x_0) = y_0; \quad x_0 \leqslant x \leqslant X.$$

We know (see Chapter 9) that according to the Runge-Kutta method the segment (x_0, X) is broken up into $x_1, x_2, \ldots . x_n = X$, $x_{i+1} - x_i = h$ and takes the value of function $y(x)$ at point x_{i+1} as

$$y_{i+1} = y_i + k, \; i = 0, 1, 2, \ldots, n-1 ,$$

where

$$k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4); \; k_1 = h\Phi(x_i, y_i);$$

$$k_2 = h\Phi\left(x_i + \frac{h}{2}, \; y_i + \frac{k_1}{2}\right); \; k_3 = h\Phi\left(x_i + \frac{h}{2}, \; y_i + \frac{k_2}{2}\right);$$

$$k_4 = h\Phi(x_i + h, \; y_i + k_3).$$

STAT

The preparation of a calculation scheme includes the introduction of the following calculation operators:

L - operator, which adds h/2 to the current argument
K' - operator, which adds $k_1/2$ to y_i
K'' - operator, which adds $k/2$ to y_i
K''' - operator, which adds k_3 to y_i
K - operator, which calculates the quantity k
Φ - operator, which calculates the value of function (x, y)
H - operator, which calculates $k_i = h$ (x, y)

With the aid of these operators we can very easily make up the calculation scheme

$$\prod_{i=1}^{n} (\Phi NLK'\Phi NK''\Phi NLK'''\Phi NK)_i.$$

Prior to working out the program circuit, we must allot the memory cells. The values x and y, for which we calculate the function Φ are located respectively into the cells α and β, whereas the value Φ is formed in the cell δ. In order to preclude the choking-up of the old value Φ by the new one, numbers k_1, k_2, k_3 must be transferred to cells $\delta_1, \delta_2, \delta_3$ (the fourth value of Φ, i.e. the k_4 may be left alone). Furthermore, inasmuch as to the argument x_i is invariably added the same quantity h/2, this addition may be invariably made to the same cell α. Since, however, the y_i gets every time a different quantity added thereto, y_i must be stored up in one more cell γ.

Having performed such allotment of the memory and beforehand introduced clarity into the functions of the above mentioned operators and having introduced some new operators, we can now work out the program circuit. The new operators will be as follows:

P - the operator feeding x_o into α and y_o into β ;
H - the operator calculating h/2;
j - the operator calculating the counter j ;
R - the operator reconstructing the operator T_j ;
Φ - the operator calculating the value Φ (x, y) by x and y resting in cells α and β : the value Φ (x, y) itself is formed in cell δ ;
N - the operator forming $k_j = h$ (j = 1, 2, 3, 4) in cell δ ;
T_j - the operator transferring $k_j = \delta_j$ (j = 1, 2, 3);
L - feeds h/2 into cell α ;
K' - forms up $\frac{k_j}{2}$ (divides the contents of cell δ in two)
K'' - adds the content of cell δ (i.e. $\frac{k_1}{2}$, $\frac{k_2}{2}$, k_3) to y_i (stored in

STAT

cell γ) and places the result into cell β ; these two operators K' and K'' replace by themselves the afore-introduced operators K', K'' and K''');

K - the operator calculating and printing y_{i+1} ; the value y_{i+1} being fed into β and into γ ;

λ_j - the operator checking the value of index j;

F(j) - the operator transforming T_j;

f(j) - the operator feeding 1 into counter j;

λ_x - the operator comparing the current argument x_{i+1} of calculated value y_{i+1} with X.

Presently the program circuit will be as shown on Figure 4-3.

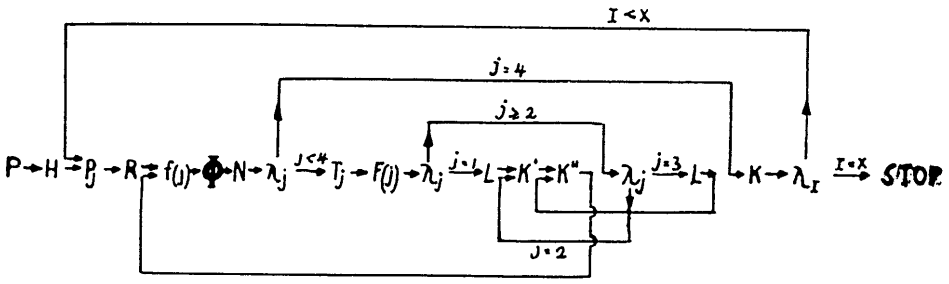


Figure 4-3.

Having worked out the program circuit and knowing exactly all the operators, we can now, without difficulty, make up the program itself. At this, we shall not be interested in the function Φ and we shall assume, that the program of calculation of Φ (x, y) is stored in the cells from ℓ+1 to ℓ+n, and the command returning us to the interrupted portion of the program is stored in cell ℓ+n.

The program for the solution of common differential equations by the Runge-Kutta Method.

P	k+1	PCh	"x_o"	-	α	
P	k+2	PCh	"y_o"	-	β	
P	k+3	PCh	"y_o"	-	γ	
H	k+4	:	"h"	"2"	p_1	$\frac{h}{2}$
P_j	k+5	PCh	"0"	-	p_o	Clearing of counter j
R	k+6	PCh	k+31	-	k+11	
f(j)	k+7	+	"1"	p_o	p_o	
	k+8	PU	+1	-	-	Transfer of control to the calculation program Φ (x, y). The last command of this program transfers the control to cell k+9
N	k+9	x	δ	"h"	δ	
j	k+10	<	"3"	p_o	k+21	
T_j	k+11	PCh	δ	-	δ_j	
F(j)	k+12	SK	k+11	"1IIIA"	k+11	
λ_j	k+13	<	"1"	p_o	k+18	
L	k+14	+	p_1		α	
K'	k+15	:	δ	"2"	δ	$k_j/2$
K"	k+16	+	γ	δ	β	$y_i + k_j/2$ or $y_i + k_3$
	k+17	PU	k+7			
λ_j	k+18	<	p_o	"3"	k+15	
L	k+19	+	p_1	α	α	
	k+20	PU	k+16			
K	k+21	+	δ_1	δ	δ	$k_1 + k_4$
K	k+22	+	δ_2	δ_3	δ_3	$(k_2 + k_3)$
K	k+23	x	δ_3	"2"	δ_3	$2(k_2 + k_3)$
K	k+24	+	δ	δ_3	δ	$k_1 + k_4 + (k_2 + k_3)2$
K	k+25	x	δ	"$\frac{1}{6}$"	δ	k
K	k+26	+	γ	δ	β	y_{i+1}
K	k+27	PCh	β	-	γ	y_{i+1}
K	k+28	Print	β			
λ_x	k+29	<	α	"X"	k+5	
	k+30	Stop				
	k+31	PCh	δ	-	δ	Initial appearance of operator T_i

In the sense as delineated at the end of preceding paragraph, [redacted] be called a standard subprogram of solution of a

STAT

common differential equation by the Runge-Kutta Method. Having been worked out and adjusted, it can be incorporated in the perforated tape, which will be then stored in the library of standard sub-programs. Every time when we should have to solve an equation of this type, it would be sufficient to make up its right section, place the first command of this program in cell 2+1 and place at the program's end the transfer of control into cell k+9. This program is then also incorporated in the perforated tape, whereupon both tapes, independently from each other are fed into their respective positions in the memory device.

In chapter 7 we shall consider the logical program circuits for the solution of systems of common differential equations by the Runge-Kutta Method.

4-3. The Program of Calculation of a Determinant. Transformation of Commands in Several Cycles.

General ways of working out a program of mathematical problems can be well illustrated by the example of calculation of the determinant.

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} \cdots & a_{nn} \end{vmatrix}$$

For the solution of this problem we can apply Gauss's method, according to which, all determinant elements located below the main diagonal, turn into zeros and, consequently, the determinant becomes equal to the product of the diagonal elements. The turning of the above-mentioned elements into zero takes place in accordance with the theorem, to the effect that out of all elements of one line of the determinant, it is possible to subtract the elements of another line, multiplied by any chosen coefficient. In order to turn element a_{21} into zero, it is necessary to form a coefficient a_{21}/a_{11} and then perform the subtraction from the elements of the second line the elements of the first line and multiplied by this coefficient.

Element a_{31} turns into zero at subtraction from the elements of the third line of respective elements of the first line, multiplied by coefficient a_{31}/a_{11}.

Thus, it is possible to reduce to zero all the elements of the first column situated below the diagonal. In order to turn into zero the elements of the second column, we must perform the following: we make up the coefficient a_{32}/a_{22} and from the elements of the third line subtract the respective elements of the second line, multiplied by this coefficient. At that, the element a_{32} turns into zero. In order to turn into zero the element a_{42}, we subtract from the elements of the fourth line the respective elements of the second line multiplied by the coefficient a_{42}/a_{22}, and so on, until all other elements of the second column become zero, too (the first element of the second line is a zero already, so that such a subtraction will do no harm to the elements of the first column). Having turned into zero all the elements of the second column, we take up the third one, and so forth. As soon as all the elements situated below the diagonal within the 1st, 2d,..., i-1st columns and the first j-i-1 element $a_{i+1,i}$, $a_{i+2,i}$,..., $a_{j-1,i}$, standing in the i-th column below the diagonal element, have been turned into zero, we must turn into zero the element a_{ji}. (The first index denotes the number of the line, the second index denotes the number of column, see Fig. 4-4).

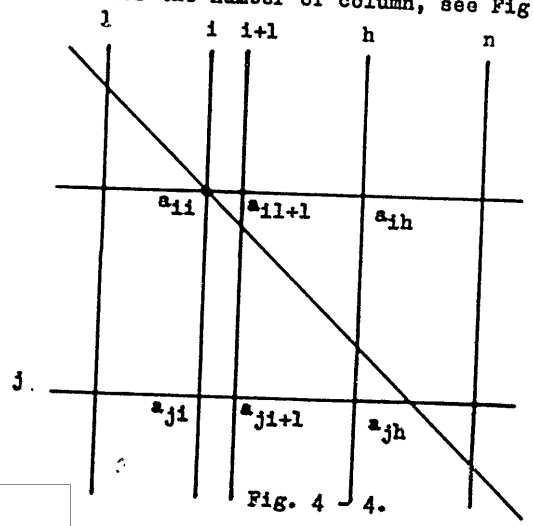


Fig. 4 - 4.

For this we calculate the coefficient $a_{ji}a_{ii}$ and from the elements of the j-th line subtract the corresponding elements of the i-th line, multiplied by this coefficient. In order to save the time, it is expedient to subtract only from the elements $a_{j,h}$ $(h=i+1, i+2,\ldots,n)$, since we know in advance, that the elements a_{j1}, $a_{j2},\ldots,$ a_{ji-1}, a_{ji} will at this subtraction turn into zero.

For convenience we may introduce into the scheme of calculation the following calculation operators:

B_{ij}-operators calculating the coefficient a_{ji}/a_{ii}

A_{ijh}-operator calculating the new value of the element

$$a'_{jh} = a_{jh} - \frac{a_i}{a_{ii}} a_{ih},$$

which is located at the intersection of the j-line and the h-column.

With the use of these operators we can express the calculation scheme, upon reduction to triangular mode; as

$$\prod_{i=1}^{n} \prod_{j=i+1}^{n} B_{ij} \prod_{h=i-1}^{n} A_{ijh}.$$

Upon the introduction of operator C_i, which multiplies the diagonal elements, we can make up the symbolic calculation scheme

$$\left(\prod_{i=1}^{n} \prod_{j=i+1}^{n} B_{ij} \prod_{h=i+1} A_{ijh}\right) C_i.$$

Now we have to allot the memory cells and, in particular, earmark the cells for the determinant's elements. It is convenient to have the determinant's elements situated in successive series of cells of the memory device, in accordance with their turn on the lines of the determinant, as it is shown at the end of the below-described program. Consequently, at the passage of one element to its neighbor on the line, the number of cell changes by one, whereas at the passage of one element to its neighbor in the column, the number of the cell changes by n ones.

STAT

 STAT

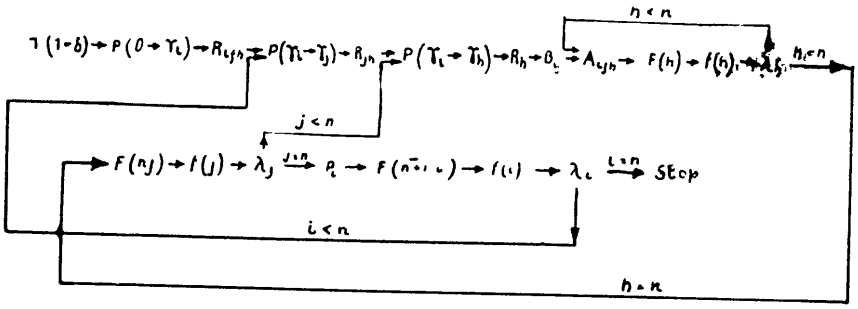
Before we set about the making up the program circuit, we must invite the reader's attention to one very important remark. As it is evident from the symbolic scheme of calculation, the operators B_{ij} and A_{ijh} undergo a change not in one cycle, but in several cycles (for each of the indices i,j,h, in the scheme of calculation, there is a corresponding program cycle).

It may occur, that every command of these operators depends on only one of the indices i,j,h,. Then, their transformation and reconstruction take place by the usual way. However, such operators often contain commands depending at the same time on several indices. Hence, they undergo changes in several cycles. Hereafter we shall explain the way of transformation and reconstruction of such commands. At the time being, we are going to make a preliminary step and examine, for that purpose, the command k + 15 from the below-described program. Its initial expression is indicated at the end of the program, in cell β_3. The commands k+5, k+9 and k+12 convey it, respectively, into cells β_3' β_3'' and, at last, into its work place in the program, i.e. into cell k+15. Fixing the value $i=i_0$ and taking at first j as $j=i_0+1$, we shall see that index h will alternately have had all values from i_0+1 to n. This transformation is made by the command k+17. In order to give index j the value i_0+2, it is necessary to have reconstructed in the command k+15 the value h, which is equal to i_0+1 and increase j by 1. Since j is a line number, the change of j by 1 corresponds to change of addresses depending on j, by n unities. This command's code with $i=i_0$ and $j=i_0+1$ is stored up in cell β_3'', consequently, the index j in it can be changed by 1 (which is performed by command k+21) and the control can be transferred to command k+12, which performs the transfer of β_3'' to cell k+15 (actually, command k+23 transfers the control to command k+10, because it is not only the command k+15 that depends on index j). Assuming that indices j and h have become equal to n and it is necessary to impart to i the value i_0+1 (inasmuch as i is a number of the diagonal element, changing of i by 1 corresponds to changing of respective addresses by n+1). Since the cell β_3'' has already underwent the transformation, the changing of index i takes

STAT

place in the cell β_3 (performed by command k+27) and the control is being transferred to the command that sends the contents of this cell onto the work place. Actually, the command k+30 transfers the control to cell k+7, command k+9 transfers the content of cell β_3 to cell β_3 (index i must be changed once more, but this time from the value i_0+2) and, at last, command k+12 transfers the contents of cell β_3 to the work place. The command k+5, which transfers the initial expression of command k+15, i.e. the contents of cell β_3 to cell β_3 makes the whole program self-reconstructing. From this analysis we can deduce a very useful rule. If a command undergoes transformation in several cycles, then, at the beginning of every cycle it is transferred to some cell, for which it must be taken out of that cell in which it had been stored up at the beginning of the preceding cycle. At this, at the first cycle it is taken out of the cell which stored its initial expression, whereas at the last cycle it is transferred not to reserve cell, but to its work place in the program. Within the cycle, the transformation of this command takes place not at the work place (with the exception of the last cycle), but in that cell in which it was stored up at the beginning of the given cycle.

Now we can construct a program circuit wherein the counters of indices i, j, j, are respectively designated with Yi, Yj, Yh (Fig. 4 -)



This scheme is used for the working out the program, with due consideration of characteristic peculiarities of reconstruction operators R_{ijh}, R_{jh} and R_h, which were described above.

A Program for Determination of a Determinant

T (1 → k)	k + 1	PCh	"1"	-	δ	
P(0 → γ_j)	k + 2	PCh	"0"	-	γ_1	
R_{1jh}	k + 3	PCh	β_1	-	β'_1	
	k + 4	PCh	β_2	-	β'_2	
	k + 5	PCh	β_3	-	β'_3	
	k + 6	PCh		-	k + 24	
P(γ_i → γ_j)	k + 7	PCh	γ_i	-	γ_j	
Rjh	k + 8	PCh	β'_1	-	k + 13	
	k + 9	PCh	β'_3	-	β''_3	
P(γ_1 → γ_h)	k + 10	PCh	γ_i	-	γ_h	
	k + 11	PCh	β'_2	-	k + 14	
	k + 12	PCh	β''_3	-	k + 15	
	k + 13	X X	X	XXX	XX	"a_{21}", "a_{11}", P_1
A_{ijh}	k + 14	X X	X	XXX	XX	P_1, "a_1 P_2
	k + 15	X X	X	XXX	XX	"a_{22}", P_2, A_n
	k + 16	SK	k + 14	"IIIA"	k + 14	
	k + 17	SK	k + 15	"1I,IIIA"	k + 15	
f (h)	k + 18	+	"1"	γ_h	γ_h	
γ_h	k + 19	<	γ_h	" n-1"	k + 14	
F(nj)	k + 20	SK	k + 13	"nIA"	k + 13	
	k + 21	SK	β''_3	"nI,IIIA"	β''_3	
F(i)	k + 22	+	"1"	γ_j	γ_j	
A_j	k + 23	<	γ_j	"n-1"	k + 10	
Di	k + 24	X X	X	XXX	XX	"a_{11}",

STAT

	k+25	SK	β'_1	"(n+1)I,IIA"	β'_1
$F(\overline{n+1\,i})$	k+26	SK	β'_2	"(n+1)IIA"	β'_2
	k+27	SK	β'_3	"(n+1)I,IIIA"	β'_3
	k+28	SK	k+24	"(n+1) IIA"	k+24
f(i)	k+29	+	"1"	γ_i	γ_i
λi	k+30	<	γ_i	"n-1"	k+7
	k+31	Stop			

B_{ij} β_1	:	a_n+n+1	a+1	P_1	(i,j)
A_{ijh} β_2	x	p_1	a+2	p_2	(i,h)
A_{ijh} β_3	-	a+n+2	p_2	a+n+2	(i,j,h)
C β_4	x	δ	a+1	δ	(i)

$a+1\ a_{11}$, $a+2\ a_{12}$, ... , $a+n\ a_{1n}$ } The first determinant's line

$a+n(n-1)+1\ a_{n1}$, $a+n(n-1)+2\ a_{n2}$, ... , $a+nn\ a_{nn}$ } The n-th determinant's line

$a+r+1\ a_{21}$, $a+n+2\ a_{22}$, ... , $a+2n\ a_{2n}$ } The second determinant's line

γ_i-counter of index i
γ_j-counter of index j
γ_h- counter of index h

.....

The initial form of commands. Dependence on an index is indicated in the parantheses

STAT

Dotted line shows the limits of the cycles. Alternate commands are not shown (they are replaced by shading), but to the right of the program is shown their initial expressions. The answer accumulates in cell δ .

4-4. Solution of Algebraic and Transcendental Equations

Let it be known, that an interval $(\omega_0, \tilde{\omega})$ there is a finite number of roots of algebraic or transcendental equation $D(\omega) = 0$, and that the distance between the adjacent roots is not less than 2 h. It is desired to determine the smallest of those roots. For this, we calculate the values of function $D(\omega)$ at points ω_0, $\omega_1 = \omega_0 + h$, $\omega_2 = \omega_0 + 2h$, ..., ... and compare their signs at two successive points. By the strength of our assumptions, the smallest root of equation $D(\omega) = 0$ is located inside of the first of those intervals $(\omega_i, \omega_i + h)$, at the ends of which the function $D(\omega)$ has different signs. This interval is divided in two and out of the two we select the one at which the function changes its sign. Repeating this process a sufficient number of times, we can make, the interval holding the root as small as we please and, consequently, find out the desired root with any degree of accuracy.

The program listed below was used for the determination of critical speeds of the rotors of turbogenerators (see chapter 5). Solution of this problem by the machine BESM made high demands upon the scope of the program, which compelled us to make the program somewhat more complicated. It will be useful for the reader to get acquainted with such a complicated, although not a long program. Following the customary line actions(described in Par. 4-1), we shall start off with the construction of a calculation scheme. At the first stage of the calculation we determine the value of the function D at a point (ω_0) and retain the value $D(\omega_0)$. Then we add step h to ω_0 and calculate the value of D at new meaning of argument $\omega_1 = \omega_0 + h$. If the sign of D underwent no change, the newly found value of the function is retained, and to the last value of the argument we add step h and compare the sign of the retained value of function D with the sign of the value found at new value of argument.

This process is continued until we get to the first interval (ω_i, ω_i+h), at whose ends the function D has different signs.

After that, the calculation is carried out in accordance with the following rules (see Fig. 4 - 6)

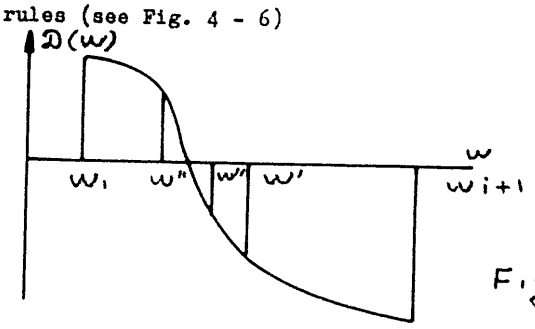


Fig. 4-6

1. At every new stage the step is divided in two.

2. The two consecutive values $D(\omega)$ are compared with each other and the newly found value is discarded when the sign underwent a change, or is retained when it has not.

3. If the sign $D(\omega)$ underwent a change, the next step is subtracted from the last value of the argument, otherwise it is added to the last value of the argument.

In other words, in this calculation scheme, the value of function at the newly created point is compared with the last of those values, which has the same sign as its predecessor has.

For the working out the program circuit, we may introduce the following operators:

T- transfer of initial point ω_o to the work cell;

R- reconstruction of step;

P_i- preparation of counter i (counting of number of points in which is calculated function D);

P_j- preparation of counter j (counting of number of divisions);

P_r- preparation of positive value of controlling parameter r (at $r > 0$ step is undivisible, at $r < 0$ divisible);

f(r)- inparting to parameter r a negative value;

f(t)- working out the controlling parameter t (product of two values of D): at $t > 0$ step is added, at $t < 0$ step is subtracted);

D- computation of function $D(\omega)$;

T'- transfer of the found value of function D to the work cell;

L- division of current step in two;

B'- addition of current step;

B"- subtraction of current step;

Π (Latin P) - printing the result;

f (i), f (j) - increase of indices i and j by 1.

Now we can make up the program circuit (Fig 4 - 7).

At this, it is not needed to break up the operator D into separate commands, since the concrete expression of function D (ω) is now immaterial. We assume, that the program of calculation of value of function D at some point, is stored in a group of cells that begins with the cell numbered with p. The last command of this program returns the calculation to the command k+10. The complexity of this program calls for a few explanatory remarks, that follow below:

The Program for the Solution of Equation D (ω)=0

T	k+1	PCh	ω_0-h	-	β
R	k+2	PCh	"h"	-	δ
Pi	k+3	PCh	"$\frac{1}{2}$"	-	γ_1
Pj	k+4	PCh	"$\frac{1}{2}$"	-	γ_j
Pr	k+5	PCh	"1"	-	γ_2
λ r	k+6	<	γ_2	"0"	k+15
T'	k+7	PCh	a	-	a'
B'	k+8	+	β	δ	β
	k+9	PU	p	-	-
f(i)	k+10	+	"1"	γ_i	γ_i
λ i	k+11	<	γ_1	"2"	k+7
f(t)	k+12	x	a	a_1	γ_t
λ_t	k+13	<	"0"	γ_t	k+6
f(r)	k+14	-PCh	"1"	-	γ_2
L	k+15	:	δ	"2"	δ
f(i)	k+16	+	"1"	γ_j	γ_j
λj	k+17	<	"n"	γ_j γ_t	k+21
λ_t	k+18	<	"0"		k+7
B'	k+19	-	β	δ	β
	k+20	PU	p	-	-
Π	k+21	Print	β	-	-
	k+22	Stop			

Transfer of control for determination of D

a - formation of new value D;

a'- retention of old value D;

β - formation of current argument ω ; at end of calculation in cell β is formed root.

δ - formation of current step

γ_i - counter of index i

γ_j - counter of index j

γ_r - cell for storing the parameter r

γ_t - cell for storing the parameter t.

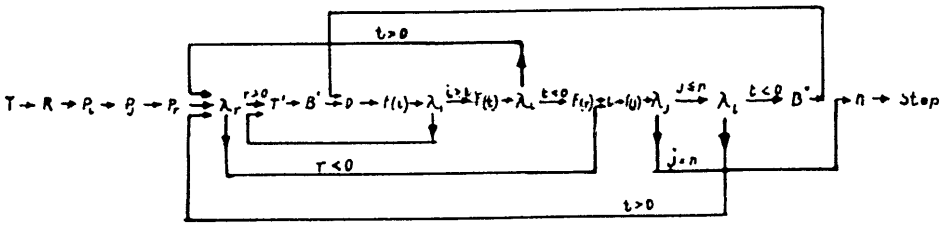


Fig. 4 - 7.

Firstly, the command k+1 transfers to cell β not the initial value of argument ω_0, but ω_0-h, for the operator B' is executed the first time prior to the execution of operator D. Secondly, the counters γ_i and γ_j get halves, not zeros. This is no, in order to insure that in the command k+11 the numbers $1\frac{1}{2}$ and 2 could be compared before the numbers 2 and 2, and then followed by the comparison of $2\frac{1}{2}$ and 2. Thirdly, despite out interest in the separation of only the first point ω_0, at every transition to a new value of argument, the counter γ_i gets 1 added therein. Thus we need to perform the comparison k+11 only one time, but it in no way disturbs the further run of the program. Let us agree to regard the just considered program (excluding the command k+1) as one operator.C. This operator will be used in the working out of a program of calculation of critical speeds of the rotors of turbogenerators, tocome up in the next chapter.

4-5. Storage of Functions in the Machine

Very often, for the solution of technological and natural scientific problems we have to know the values of special functions, such as trigonometric and hyperbolic functions of Bessel and other functions. Performing the calculation manually, the operator can take such

STAT

STAT

values from special tables, which is not the case when the calculation process is entrusted to an automatic computation machine. Therefore, the program of a problem's solution must be provided with subprograms of calculation of the functions by some formulas. The Par. 3 - 3 has, for instance, contained the programs of calculation of function e^x and sin x by way of their expansion into series, by the powers of x. Par. 3 - 6 included the description of programs of calculation of functions shx and chx. As it was pointed out in par. 3 - 8, the application of such formulas should be exercised with utmost care and performed in such a way as to secure the highest possible degree of shortening the time needed for calculation. This is particularly true with respect to such problems, whose solution is liable to call for frequent use of subprograms for special functions. Sometimes it is in our behoof to secure a reduction of computation time by using one value of argument instead of another through the application of some analytic correllations, as for example,

sin $(2k\pi+x)$ = sin x;

$\lg e^p z = p+\lg z$, etc. (see par. 3 - 9)

In this respect, the formulas representing special functions in the series according to Chabyshev's polynomicals which were made use of with success at the working out the subprograms for the BESM machine, can be very useful, indeed. Sometimes, for the calculation of values of functions at various values of argument it is useful to apply different formulas.

However, very often the values of functions are given not in the form an analytic formulas, but in the form of tables or graphs resulting from experimental work. In such a case, it will be necessary, on the basis of data on the graph, to make up a table, as the computers can handle only numeral data. In order that we could determine the value of functions at an arbitrary value of argument, with the use of tables showing the values of fundtions only at some fixed values of argument, we should have to use the interpolation formulas. This can be done in two different ways. If the problem indicates that the argument is capable of changing within relatively small limits, or if it is known

STAT

that the function can change smoothly, its value can be calculated by one and the same interpolation polynomial. In this case it is useful to have the coefficients of interpolation polynomials calculated beforehand and then, upon introducing them into the machine, conduct the calculation of the value of the polynomial with the use of the Gorner's method, described in Par. 3 - 7. If this can not be done, a table of values of function can be introduced into the machine, whereupon the machine itself will construct an interpolation polynomial and then use it in the calculation of the value of function. In this case, the most difficult task would be the finding out of an entry onto the table with a set value of argument x. Let in n+1 equidistant points $x_0, x_1=x_0+h, x_2=x_0+2h, \ldots, x_n=x_0+nh$, be set the values of function $y_0, y_1, \ldots, y_n$. They are respectively located in cells 1+0, 1+1, 1+2, ..., 1+n. Let us assume, that the employed interpolation formula is that of the k-th sequence, wherein the value of function at a certain point x is expressed by the table values of $y_i, y_{i+1}, \ldots, y_{i+k}$, where the value of y_i corresponds to point x_i, which satisfies the inequations $x_i \leqslant x < x_{i+1}$. It is desirable to find out the value of y_i with the use of the set value of x, wherein $x_0 \leqslant x \leqslant x_{n-k}$. Let us examine the difference $x-x_0$, which is equal to the length of segment (x_0, x). The whole part of relation $\frac{x-x_0}{h}$ is equal to the number of points $x_1, \ldots, x_i$, located to the left of point x (Fig. 4 - 8).

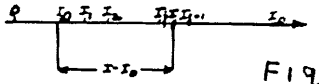


Fig. 4-8

If to this number we add number 1 (number of cell storing x_0), then the number $\frac{x-x_0}{h}+1$ will be equal to the number of that cell 1+i, which is storing y_i. For working out a program capable of selecting number y_i and conveying it into some work cell a, we can employ the operation "ant'ye", which places the whole part of a number into the junior columns of the third address.

STAT

k + 1	-	,x'	,x_o'	β
k + 2	:	β	,h'	β
k + 3	[ ]	β	γ	γ
k + 4	←	δ	24	δ
k + 5	ЗК	k + s	δ	k + 6
k + 6	РСh	1 + i	-	a
k + s	РСh	1	-	a

The junior columns of cell δ contain $\left[\frac{x-x_o}{h}\right]$

The number $\left[\frac{x-x_o}{h}\right]$ is placed into the junior columns of 1st address of cell δ

Cell k+s stores the command transferring the contents of cell ℓ to cell a. With the use of operation of addition of commands, to the first address of cell k+s is added number i and the result is placed into cell k+6. Thereby, in cell k+6 is being formed the command transferring the unknown quantity y_i to work cell a. Meanwhile, the first address of that command gives the number of the cell storing the y_i. Determination of values $y_{i+1}, y_{i+2}, \ldots y_{i+k}$, needed in the construction of an interpolation formula, is not difficult any more.

Likewise, the program of calculating interpolation polynomials can, in this case, proceed from a notion of representation of this polynomial with the use of finite differences constructed by the machine itself, as well as directly, with the values of function $y_i, y_{i+1}, \ldots, y_{i+k}$. With regard to the number of commands, the last method is more economical.

Now let the same problem be solved by a machine with fixed comma, and let all the numbers be stored in the machine's memory device having a scale $\frac{1}{M}$. If we now multiply the relation $\frac{1}{M} \cdot \frac{x-x_o}{h}$ by $\frac{1}{2^p}$ and divide the product by $\frac{1}{M}$, we get a number $x' = \frac{1}{2^p} \cdot \frac{x-x_o}{h}$, wherein p is chosen in such a way as to have $\frac{1}{2^{p-1}} \leqslant \frac{1}{M} \leqslant \frac{1}{2^p}$.

Substituting $\frac{1}{2^p}$ for scale $\frac{1}{M}$ we have accomplished, that at present the whole part of number x' is stored in the p senior columns of the cell. By way of separation of the whole part through logical multiplication by $\underbrace{11\ldots1}_{p}00\ldots0$ and shift to a needed number of columns,

the whole part of number x' finds itself in the junior columns of the first address. Further construction of the program will be made after the analogy of the foregoing case.

The table method of task-setting is, at time, made use of for functions, for which the analytical formulas are too cumbersome, or call for a great number of operations. It can be conveniently employed then, when there is no need to secure very precise calculation or when it is known beforehand, that the function's argument change within narrow limits.

Let now, at last, the values of function be set at n+1 non-equidistant points. In order to find an entry onto the table, it is necessary, in this case, to have placed into the memory device both the value of the function and the value of argument. Let points $x_0, x_1, \ldots, x_n$ be placed into cells $1, 1+1, \ldots, 1+n$ and the respective values of function be placed into cells $1+n+1, 1+n+2, \ldots, 1+2n+1$. For finding out an entrance onto such a table, with the set x, it is necessary to subject x to a succession of comparisons with points $x_0, x_1, x_2, \ldots$, until we find that first point x_i, for which $x_i \leqslant x$. The number of the cell storing y_i is determined by a separation of address of that point and subsequent addition to that address of n+1 unities of a corresponding address.

CHAPTER V

THE ASCERTAINMENT OF CRITICAL SPEEDS OF THE ROTORS OF TURBO-GENERATORS

5-1. Setting the Problem

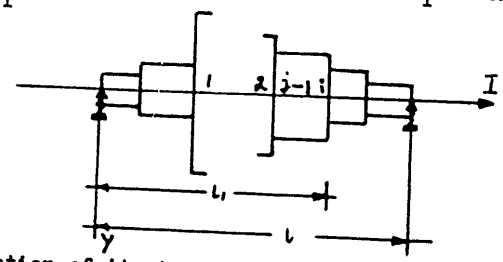
It is known, that during rapid revolving of hafts develops the following phenomenon: with the increase of angular velocity ω after a while it reaches such a magnitude $\omega = \omega_1$, at which the shaft does not maintain the rectilineal form and begins to knock. Then, after a while, while the angular velocity is still on the increase, the knocking stops, only to start again at $\omega = \omega_2$, etc. Speeds $\omega_1, \omega_2, \omega_3 \ldots$ are called critical speeds and it is very important,

at calculations of various revolving shafts, to provide for such limits, which would differ from the work speed by not less than 20 - 30%. This chapter sets forth the process of calculation of critical speeds of the rotors of powerful turbogenerators, depending on the vibrations of its own supports, carried out by digital automatic computers. (This problem was brought up by a member of NIIEP of the USSR State Planning Commission, Candidate of Physico-Mathematical Sciences I. Ye. Sakharov).

The shaft knocking is explained by practical impossibility to attain the complete coincidence of the shaft's geometrical axle with one of the main axles of inertion, which results in giving rise to appearance of disturbing forces. When the frequency of those forces, numerically equal to the number of the shaft's revolution per second, becomes equal to the number of its own lateral fidgeting of the shaft, then arises the phenomenon of resonance and the shaft begins to oscillate.

The turbogenerator's rotor can be regarded as a pivot consisting of n straight circular cylinders of various diameters having the same axle (Fig. 5 - 1). Let us align axle x along the axle of pivot and designate with y_i (x,t) the disalignment of the axle of the i-th cylinder at point x at the moment of period t.

Let the supports of the shaft be placed at points x=0 and x=l, and the boundary between the i-th and the i+1 sections be located at point $x=l_i$ and $s=\frac{x}{l}$. If B_i is the rigidity and m_i is the mass of an i-th cylinder



then the equation of its transverse vibrations will be as follows:

$$a_i^4 \frac{\partial^4 y_i}{\partial s^4} + \frac{\partial^2 y_i}{\partial t^2} = 0, \quad a_i^4 = \frac{B_i}{l^4 m_i}. \qquad \text{(Fig. 5 - 1)}$$

STAT

At this, the supports are regarded as oscillatory systems with one degree of freedom. The boundary conditions are as follows:

on the left support (s=0)

$$S_1 = S_\lambda;$$

$$S_1'' = 0;$$

$$-B_1 S_0''' = a_\lambda S_\lambda \left(1 - \frac{\omega^2}{\omega_\lambda^2}\right); \qquad (5\text{-}6)$$

on the right support:

$$S_n = S_{np};$$

$$S_n'' = 0;$$

$$B_n S_n''' = a_{np} S_{np} \left(1 - \frac{\omega^2}{\omega_{np}^2}\right), \qquad (5\text{-}7)$$

where a_λ, a_{np} are the constants depending on physical qualities of the rotor and supports, whereas the ω_λ, ω_{np} are its own frequences of oscillations in the supports ($\omega_\lambda = \sqrt{1/\delta_\lambda M_\lambda}$, $\omega_{np} = \sqrt{1/\delta_{np} M_{np}}$, δ and M according to the pliability and the mass of the supports). At substitution of formulas (5-3) and (5-4) within the boundary conditions of (5-5), (5-6) and (5-7) we get 4n+2 uniform equations for ascertainment of the unknown S_λ, S_{np}, A_i, B_i, C_i, D_i, i=1, 2,...,n. For existance of non-trivial solution of this system, its determinant regarded as a function of frequency ω, must be equal to zero.

At the calculation of the critical speeds of a rotor of a turbogenerator of 150,000-kw capacity, the critical speeds of the rotor have clarified themselves as roots of the equation

$$D(\omega, \omega_\lambda, a_\lambda, \omega_{np}, a_{np}) = 0, \qquad (5\text{-}8)$$

which constitutes the determinant of the 46th order (table 5-1, see the inset at the end of book). Here, at this rotor, the ω_λ, a_λ, ω_{np}, a_{np} are constants depending on the dynamic pliability of the supports. By imparting to ω_λ, a_λ, ω_{np}, a_{np} various values, we can establish the dependency of the critical speeds of the rotor on the physical qualities of the supports.

Its own vibrations of the rotor-support system develop according to the law of

$$y = S(s)\sin\omega t; \qquad (5\text{-}2)$$

$$y_\lambda = S_\lambda \sin\omega t; \quad y_{np} = S_{np}\sin\omega t; \qquad (5\text{-}3)$$

wherein is its own frequency of oscillations in the rotor-support system and y_λ and y_{np} are displacements of the supports centers of gravity. The equation (5-1), after introduction into it of (5-2) will be this $S_i^{IV} - \frac{\omega^2}{a_i^4} S_i = 0$.

We shall seek for a solution of this equation in the form of:

$$S_i(s) = A_i \operatorname{ch}\frac{s}{a_i}\sqrt{\omega} + B_i \operatorname{sh}\frac{s}{a_i}\sqrt{\omega} + C_i \cos\frac{s}{a_i}\sqrt{\omega} + D_i \sin\frac{s}{a_i}\sqrt{\omega}; \quad s_{i-1} < s \leqslant s_i = \frac{l_i}{1}. \qquad (5\text{-}4)$$

The constants s_λ, s_{np}, A_i, B_i, C_i, D_i can be ascertained from the boundary conditions, which for the boundary s_i between the i-th and the i+1th sections are expressed by equality of deflections, angles of inclination tangential to the axle, bending momonts and cutting forces:

$$\begin{aligned} S_i &= S_{i+1}; \\ S_i' &= S_{i+1}'; \\ B_i S_i'' &= B_{i+1} S_{i+1}''; \\ B_i S_i''' &= B_{i+1} S_{i+1}'''. \end{aligned} \qquad (5\text{-}5)$$

The boundary conditions on the supports are expressed by equality of deflections in the rotor and the displacements of the supports, by absence of bending moments at the rotor's ends and by equality between the cutting forces at the rotor's ends and the forces prevailing upon the rotor by the supports.

5-2. Solution of the Problem by the BESM Machine

The calculatory examination dealt with 7 different supports, which resulted in 28 variants of the problem, i.e. 28 set groups of numeral values of parameters

$$\omega^i_\lambda, \quad a^i_\lambda, \omega^i_{np}, a^i_{np},$$

which called for the solution of equation

$$D\ (\omega, \omega^i_\lambda, a^i_\lambda, \omega^i_{np}, a^i_{np}) = 0. \quad (5\text{-}9)$$

Approximate determination of roots of such an equation is, strictly speaking, quite a difficult task. In this case, however, the task is greatly simplified. Considering the physical factors, we know that the lowest of its frequencies surpasses 50 sec -1 and that two successive such frequencies are away from each other farther that 25 sec -1. Besides, only the first four of its own frequencies are of practical interest. This being so, we can solve the equation $D(\omega)=0$ employing the method described in Par. 4-4, whose step $h=25$. The permissible calculation error makes up 1% of the true value, i.e. exceeds 0.5, since the minimal root is more than 50. It means, that after the septuple division of step h in two, the error will be equal to $\frac{h}{2^7} = \frac{25}{128}$ i.e., it will reach the desired degree of accuracy. Beginning the calculation with $\omega_o = 50$, we determine four successive roots for one problem's variant, then go over to the next, and so on, until we have checked all the 28 variants. The calculation scheme will be as follows:

$$\prod_{i=1}^{28} T_i \prod_{k=1}^{4} C_k,$$

wherein C_k is the operator finding out the k-th root of equation $D(\omega)=0$; T_i is the operator transferring the constants ω^i_λ, a^i_λ, ω^i_{np}, a^i_{np} into the work cells.

Working out the program, we can make use of operator C, introduced at the end of Par. 4 - 4. If, in the structure of that operator (Fig. 4-4) we replace the command "stop" by the command that transfers STAT

the control to operator R, the thus obtained program can be employed for ascertainment of roots of equation $D(\omega)=0$.

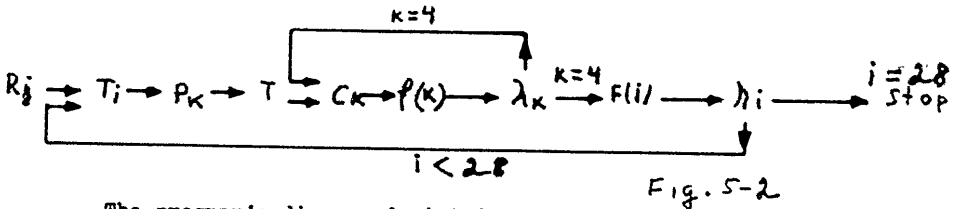


Fig. 5-2

The program's diagram depicted on Fig 5-2 contains the following operators:

R_i - reconstruction of operator T_i

T_i - transfer of ω^i_{λ}, a^i_{λ}, ω^i_{np}, a^i_{np} to the work cells

P_k - preparation of counter k (number of calculated roots)

T - reconstruction of initial point

f(k) - addition of 1 to the contents of counter k

λ_k - comparison of k with 4

F_i - transformation of operator T_i

λ_i - comparison of i with 28 (comparison of variable command with standard command).

The scheme of operator C_k includes operator D, which calculates the value of function $D(\omega)$ at any desired point. The expression of this function has not been rendered concrete in Par. 4-4 and, therefore, operator D has not been broken down into separate commands. But in this case, it is exactly the program of calculation of value of function $D(\omega)$ that constitutes the most difficult portion of the problem. (Junior G. Gendzhoyan scientific worker of the Institute of Mathematics and Mechanics of the AS of Armenian SSR, colaborated in working out this program).

Generally speaking, the calculation of the value of determinant consists of the following: for every determinant's element we allot a special cell in the memory device, then we ascertain the value of those elements at a given ω; the obtained values are inscribed into the respective cells, whereupon the determinant is being calculated by one of the known methods. Operator D can be conveniently expressed by two operators D' and D": D' calculates the determinant's elements and allots them to their respective memory cells, whereas D" calculates the determinant itself. At first we are going to describe operator D".

As it was pointed out above, this problem was solved by the BESM machine, whose memory device contains 1,024 cells. At the same timSTAT the number of the determinant's elements reaches $46^2 = 2,116$. Should we assign a separate memory cell to every determinant's element, then,

for calculation of determinant's value at one point, we should have resorted to the services of the external memory device, which would have considerably complicated the problem and would have required a much longer time for its solution. We performed the calculation of the determinant's value by the Gauss method described in Par. 4 - 3. Inasmuch as only 350 out of 2,116 determinant's elements are different from zero, the necessity for allocation of a separate cell for every determinant's element fell away. Furthermore, the very fact that the majority of the determinant's elements are equal to zero, ensured the possibility of prompt application of the Gauss method. Actually, only 856 cells as indicated in Fig 5 - 2 were allocated to determinant's element. This allocation can be explained by the following: in order that we could preclude the necessity of division by numbers close to zero, the method described in Par. 4-3 was somewhat modified. Prior to division of elements of i-th line by a_{ii}, we had singled out from among them the highest by the modulus element a_{ik} and transplaced the i-th and the k-th columns (such transplacement changes the determinant's sign to the inverse). After that permutation, seven cells in every column were chosen and earmarked for the elements located below the diagonal. Indeed, if, for example, the largest element in the fourth line is the element located in the 45th square, then the fourth column is permutated with the ninth and in the fourth column under the diagonal element seven elements different from zero are being formed. But, from the 40th column onwards, it is sufficient to allocate to the same elements only 46-i cells, wherein i is the number of the column. Also, in the i-th column a number of elements equal to min $\{7,46-i\}$ must be turned into zero. Further, subtraction of elements of the i-th line from corresponding elements of the j-th line is expedient only there, where they differ from zero. Not more than five elements different from zero can be located in every line, right of the diagonal element, at one time. Exactly this number of memory cells was allocated to those elements. In order to conceive the meaning of purpose of cells located outside of determinant itself, we shall have

STAT

to return to the program of calculation of determinant, contained in Par. 4 - 3.

In that program, all transformations of commands are performed with help of constants containing n (or n+1) unities of one or another address, wherein n is the determinant's order. This is so, because of any two arbitrarily taken consecutive diagonal elements of the determinant being located in cells whose numbers differ by n+1 unities. In the determinant under consideration, the addresses of two consecutive diagonal elements located in the 7th to 42d column differ by 13 unities. To maintain this system permanently and in order to simplify the system of readdressing, 13 cells are allotted to every determinant's line (with the exception of the first and the last lines, where the cells located outside the determinant are unnecessary). It is evident from this consideration, that the cells located outside the determinant may contain any numbers, for we leave them altogether alone.

After all that has been said above, it is now clear wherein the program of calculation of a determinant, applied for the solution of a problem, differs from that contained in Par. 4 - 3. Firstly, in every line we singled out the largest, element, determined its address (applicable program was discussed in Par. 3 - 5), whereupon we transplaced the column containing that element with the column containing the diagonal element. Secondly, in all constants of readdressing, instead of number n (in this case n=46) figured number 13. Thirdly, we turned into zero only a number of elements equal to min $\{7,46-i,\}$ where i was number of column, and subtracted from every line only five elements. Consequently, at every passage from one column to another, number min $\{7,46-i\}$ had to be calculated anew and then be employed as a comparison constant. Likewise, instead of constant n that was used in Par. 4 - 5, we used constant 5. At this, both counters had to be cleaned. Since this method of calculation of a determinant differs from that described in Par. 4 - 3 in no principal points, we are not going to present it here, broken down into the

STAT

commands.

There remains now, to make up a scheme of operator D' which calculates the value of determinant's elements themselves and arranges them among respective memory cells. At first operator G_i (Fig. 5 - 3) places a zero in all 586 cells. By doing this, it also cleans all useless cells, but this can be accomplished merely in four commands. A program that would place zeros only into the needed cells, would be too cumbersome and complex. The groups of 16 elements, located on table 5 - 2 in squares I, III,..., XIX, differ from one another only in coefficients s_i/a_i, i=1,2,...,10 standing under the signs of functions sin, cos, sh and ch. Likewise, the groups of 16 elements located in squares II, IV,...XX, distinguish themselves only by differences in coefficients standing under the signs of these functions and before them. Calculation and arrangement of elements of these 20 squares are as follows: Constants $\frac{s_1}{a_1}, \frac{s_2}{a_2}, \frac{s_2}{a_2}, \frac{s_2}{a_3}, \ldots, \frac{s_{10}}{a_{10}}, \frac{s_{10}}{a_{11}}, \frac{s_{11}}{a_{11}}$ are placed in the above shown order, into 21 consecutive memory cells. Likewise, constants $\frac{a_1}{a_2}, \left(\frac{a_1}{a_2}\right)\frac{^2B_2}{B_1}, \left(\frac{a_1}{a_2}\right)\frac{^3B_2}{B_1}, \ldots, \frac{a_{10}}{a_{11}}$

$$\left(\frac{a_{10}}{a_{11}}\right)\frac{^2B_{11}}{B_{10}}, \left(\frac{a_{10}}{a_{11}}\right)\frac{^3B_{11}}{B_{10}}$$

are placed into 30 consecutive cells. The current value of argument ω we feed into the subprogram of calculation of a root, designated with 1". The operator T_j multiplies $\frac{s_j}{a_j}$ or $\frac{s_j}{a_{j+1}}$ by $\sqrt{\omega}$ and the control is transferred to operator I", calculating the values of functions sin, cos, sh, ch. The operator transferring the elements of squares I,III,..., XIX to respective cells is designated as K'_t, t=1,2,...,10 and the operator doing the same with the elements of squares II, IV,...,XX is designated as K''_t, t=1,2,...,10. The controlling parameter r is placed after the operator T". Depending on sign of this parameter, the calculation is referred either to operator K'_t or to operator K''_t. In the last instance, the operator

STAT

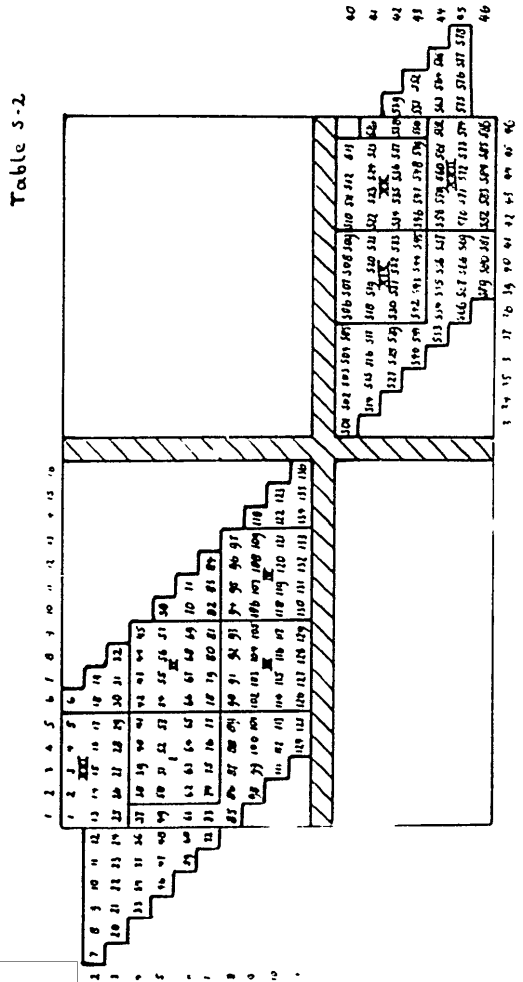
K''_t is preceded by operator T_e, which conducts the constants

$$\frac{a_t}{a_{t+1}}, \left(\frac{a_t}{a_{t+1}}\right) \quad \frac{^2B_{t+1}}{B_t}, \left(\frac{a_t}{a_{t+1}}\right) x \frac{^3B_{t+1}}{B_1}$$

in the work cells.

The work of operator K'_t, its transformation and reconstruction, should be described in some more details. Let operator T" feed the values of functions sin, cos, sh and ch into respective cells, P_1, P_2, P_3, P_4. Operator K't consists of 16 commands whose initial expression has been fed into cells n+1, n+2,..., n+16 (in their third address rest the numbers of cells corresponding to the determinant's elements, according to table 5-2).

Table 5-2



STAT

n+1 П Ч p_4 - 38

n+2 П Ч p_3 - 39

.....

n+16 П Ч p_2 -77

Scheme D' opens up with operator R, which transfers the initial expression of these commands to their work places in the program, designated as k+1, k+2,..., k+16. This operator contains five commands

1 + 1 П Ч 0 - a

1 + 2 П Ч n + 1 - k + 1

1 + 3 СК ℓ + 2„1I,IIIA" ℓ + 2

1 + 4 + „1" a a

1 + 5 < a 16 ℓ + 2

Thus, operator K'_1, which will have the elements of square I arranged among the cells, will be placed in k+1 ÷ k+16. Now operator K'_1 must be transformed so as to be like K'_2, in which form it will perform the arranging of the elements of square III. For that, to the third address of cells k+1, k+2,..., k+16 must be added 52 unities, so as, for example, the command k+1 will then take the shape of П Ч P_4—90.

This transformation is performed by operator F (t), which can be constructed the same way as operator R. Similarly, the operator K''_t is constructed. The simultaneous reconstruction of K''_t and of all other commands included in operator D' is assigned to operator R, whereas the transformation of K''_t is assigned to operator F (t). All that is left to be done now, is to have the elements within the rectangles, marked in table 5 - 2 with numbers XXI and XXII, placed into their places. This is performed by operator L, the programming ofwhich is simple. Its introduction into the program becomes clear from the scheme of operator D' in Fig. 5 - 3.

STAT

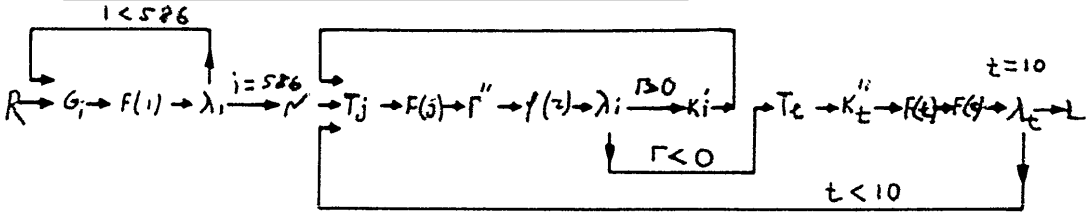


Fig. 5-3

The scheme contains the following operators:

R - reconstruction of operators G, T_j, K'_t, K''_t and parameter r.

G_i - cleaning of cells

Γ' - calculation of root

T_j - transfer of $\frac{s_j}{a_j}$ or $\frac{s_i}{a_{j+1}}$ to work cell

Γ'' - calculation of sin, cos, sh and ch

f(r) - change of sign of controlling parameter to inverse (at first $r < 0$)

K'_t - arranging of elements contained in squares I, III,..., XIX

K''_t - arranging of elements contained in squares II, IV,..., XX

L - arranging of elements contained in rectangles XXI and XXII

F - transformation of corresponding operators

λ - check-up of number of performed cycles

Preparation of schemes of operators D' and D'' constitutes the last stage of completion of the whole program. From the contents of this paragraph it is clear, that the working out of the program circuit for this problem was performed by a few stages. At first an enlarged scheme was prepared, which included such operators as D, which calculates the determinant's value. That operator was represented by the two operators D' and D'', with a separate program circuit worked out for each of the two. Likewise, the program itself can be broken up into several separate programs, each such program describing one operator and being made up of conditional addresses and cells. The next step is the distribution of memory cells and the assembly of separate operators of the overall program for the solution of the given problem, consisting not of conditional addresses but of concrete memory cells. In conlcusion let us recount the way by which

STAT

STAT

the memory was actually distributed, in this problem.

The first five cells of the memory device of the BESM machine occupied as operational cells by the standard subprograms, could not be made use of. The determinant's elements were placed into cells 11-596. The program itself contained 251 commands and was accommodated in cells 601 - 851. The standard commands and the constants used in the program (constants of comparison, readdressing, etc) occupied 81 cells, from 861 to 941. The constants

$$\frac{a_j}{a_{i+1}},\ \frac{s_i}{a_i},\ \frac{s_i}{a_{i+1}},\ \left(\frac{a_i}{a_{i+1}}\right)^{\lambda}\frac{B_{i+1}}{B_i},\ \left(\frac{a_1}{a_{i+1}}\right)^{3}\ \frac{B_{i+1}}{B_i}$$

occupied 51 cells, from 950 to 1,000.

Out of the remaining 50 cells, 31 were used as work cells. 112 constants ω^i_λ, a^i_λ, ω^i_{np}, a^i_{np} were accommodated in the external memory device, which was resorted to 28 times (operator T_i on Fig. 5-2). Calculation of every variant lasted for about 2 minutes. During that time the machine was performing about 1,000,000 operations.

CHAPTER VI

CALCULATION OF STABILITY OF AUTOMATIC REGULATION SYSTEMS BY DIGITAL COMPUTERS

6-1. Basic Data

One of the principal problems that arise at designing the systems of automatic control and regulation is the problem of ensuring their stability. The calculation and the analysis of complex dynamic systems necessitates a great effort of calculation. The use of automatic digital computers can perform such complex calculations with good success.

The first calculation method for determination of limits of stability by automatic computers was worked out by the Academy of Sciences of the Ukrainian SSR (L. 24). In this chapter we offer a description of the program used for calculation of stability of systems of automatic regulation used by the calculation laboratory

STAT

STAT

of NIIEP of the State Planning Commission for the computer M-3. In the process of working out this program a number of characteristic peculatiries of methods adopted by the AS Ukr SSR found their implementation. At the same time, the below described program contains a series of essential differences from other programs.

The contemporaneous methods of study of stability of dynamic systems are based on the Lyapunov's theory of stability of motion. The motion of a dynamic system can be represented by a system of differential equations:

$$\left.\begin{aligned} \frac{dx_1}{dt} &= f_1\ (x_1, x_2, \ldots, x_n); \\ \frac{dx_2}{dt} &= f_2\ (x_1, x_2, \ldots, x_n) \\ &\cdots\cdots\cdots\cdots \\ \frac{dx_n}{dt} &= f_n\ (x_1, x_2, \ldots, x_n) \end{aligned}\right\} \qquad (6-1)$$

wherein $x_1, x_2, \ldots, x_n$ are deflections of variable quantities (generalized coordinates-angles, currents, tensions, etc), from their values in a set-in regime; $f_1, f_2, \ldots, f_n$ are known non-linear functions of deflections of variable $x_1, x_2, \ldots, x_n$. Equations (6-1) have been named by Lyapunov "the equations of perturbed motion".

In the state of a settled regime, the deflections of variables and their derivatives are equal to zero, namely:

$$x_1 = 0;\ x_2 = 0; \ldots;\ x_n = 0 \qquad (6-2)$$

$$\left.\begin{aligned} f_1\ (x_1, x_2, \ldots, x_n) &= 0; \\ f_2\ (x_1, x_2, \ldots, x_n) &= 0; \\ &\cdots\cdots\cdots \\ f_n\ (x_1, x_2, \ldots, x_n) &= 0. \end{aligned}\right\} \qquad (6-3)$$

The zero-solution (6-2) of a system of differential equations in the Lyapunov's theory is called "the unperturbed motion". The unperturbed motion is called stable in relation to quantities $x_1, x_2, \ldots, x_n$ if at any arbitrarily given positive number ε, no matter how small it might be, there can be chosen another positive number $\tau(\varepsilon)$

STAT

so that at every perturbation $x_{10},\ldots, x_{no}$, satisfying the condition

$$|x_{1o}| \leq \gamma, \quad (6\text{-}4)$$

the perturbed motion at any $t > 0$ should secure the inequality

$$|x_i(t)| < \epsilon. \quad (6\text{-}5)$$

Otherwise, the perturbed motion is unstable. When, along with the inequalities (604) and (6-5) the equality

$$\lim_{t \to \infty} x_i = 0 \quad (i=1,\ldots,n). \quad (6\text{-}6)$$

is likewise valid, then the unperturbed motion is called asymptotically stable.

There exist methods enabling us to analyse the stability of dynamic systems without the integration of differential equations (6-1). The most powerful of them is the socalled Lyapunov's direct method (or Lyapunov's second method) based on qualities of some functions of variables $x_1, x_2,\ldots,x_n$ ("Lyapunov's functions"). Lyapunov's direct method enables us to examine the stability of nonlinear dynamic systems at any large deflections of the variables, if the initial differential equations of the system (6-1) retain their validity.

However, in many problems the finding out of Lyapunov's functions with the application of the direct method is id frought with great difficulties. Therefore, the work is often limited to examination of a system's stability (stability of unperturbed motion) (6-2) at small deflections of variables, with the application of linear differential equations ("equations of first approximation").

In a number of important practical cases, the functions standing in the right part of an equation of a dynamic system (6-1) can be arranged in series, by the whole positive powers of deflections of variables $x_1,\ldots,x_n$, if those deflections are sufficiently small.

Then the equations (6-1) assumes the form:

$$\left.\begin{aligned}
&\frac{dx_1}{dt} = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + N_1(x_1,\ldots,x_n);\\
&\frac{dx_2}{dt} = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + N_2(x_1,\ldots,x_n);\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\frac{dx_n}{dt} = a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + N_n(x_1,\ldots,x_n),
\end{aligned}\right| \qquad (6\text{-}7)$$

where $N_i(x_1,\ldots, x_n)$ are functions $x_1, x_2,,\ldots,x_n$ containing no less members than the second order of infinitesimal;

$a_{11},\ldots, a_{nn}$ are stable and where

$$a_{ik} = \left(\frac{\partial f_i}{\partial x_k}\right)_{x_1 = x_2 = \ldots = x_n = 0} \qquad (6\text{-}8)$$

The linearized equations of system, or equations of the first approximation

$$\left.\begin{aligned}
&\frac{dx_1}{dt} = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n ;\\
&\frac{dx_2}{dt} = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n ;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&\frac{dx_n}{dt} = a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n
\end{aligned}\right| \qquad (6\text{-}9)$$

are obtainable by way of discarding (6-7) of members in the right parts which exceed the first order of infinitesimal. To the system of differential equations corresponds to the characterizing equation

$$\lambda^2 + p_1\lambda^{n-1} + \ldots + p_{n-1}\lambda + p_n = 0, \qquad (6\text{-}10)$$

whose coefficients can be had by way of opening up the characterizing determinant:

$$\begin{vmatrix} a_{11}-\lambda & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22}-\lambda & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33}-\lambda & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn}-\lambda \end{vmatrix}, \quad (6\text{-}11)$$

made up of coefficients of the system of equations (6-9).

A. M. Lyapunov was the first person who established the conditions permitting examination of stability of unperturbed motion (6-1) by equation of its first approximation. According to Lyapunov, when the linearized differential equation (equation of first approximation) of the system possesses a characterizing equation the essential parts of the roots of which are negative, then the unperturbed motion $x_1=0;\ x_2=0,\ldots,\ x_n=0$ is asymptotically stable, irrespective of the members $N_i\ (x_1, x_2,\ldots,x_n)$ which are higher than the first order of the infinitesimal in the equations (6-7). If, however, at least one root has a positive essential part, then the unperturbed motion is unstable irrespective of the members $N_i(x_1, x_2,\ldots, x_n)$ being higher than the first order of infinitesimal.

It is possible, that in some instances an essential part of one or of several roots is equal to zero, whereas those of other roots are negative. Then we can not judge the stability with application of equations of the first approximation and must take into consideration the non-linear members $N_i(x_1, x_2,\ldots,x_n)$.

There exist criteria which enable us to ascertain the negativeness of essential parts of all the roots of a characterizing equation, without having to solve it. Among them are the well known satisfactory criteria of negativeness of essential parts of roots of equations of type (6-10); the algebraic criteria of Gurvits and Raus, and the frequency criteria of Naykvist, Mikhailov and others.

The best criteria for use in calculation of stability with the help of an automatic digital computer are the algebraic criteria. This is so, because they operate with material numbers, whereas the decision on the stability factors are made depending on the type of

sign of numbers obtained during the calculation. Out of a series of algebraic criteria, the best criteria for application in calculation exceeding the fourth power is the criterium of Raus, which provides a uniformity of calculations. We shall use it later and therefore we shall consider it for a while, now.

Let a linear system have a characterizing equation (6-10) of an n-th power.

Performing the calculation for the n+1 lines of table (6-1) in order to have all the roots of equation (6-10) with the negative essential parts, it is necessary and sufficient to have all the numbers standing in the first column of the table (p_1, b_1, c_1,...) positive.

Table 6-1

$p_0 = 1$	p_2	p_4	p_6	. . . . .
p_1	p_3	p_5	p_7	. . . . .
$b_1 = \frac{p_1 p_2 - p_0 p_3}{p_1}$	$b_2 = \frac{p_1 p_4 - p_0 p_5}{p_1}$	$b_3 = \frac{p_1 p_6 - p_0 p_7}{p_1}$	$b_4 = \frac{p_1 p_8 - p_0 p_9}{p_1}$	. . . . .
$c_1 = \frac{b_1 p_3 - p_1 b_2}{b_1}$	$c_2 = \frac{b_1 p_5 - p_1 b_3}{b_1}$	$c_3 = \frac{b_1 p_1 - p_1 b_4}{b_1}$	$c_4 = \frac{b_1 p_9 - p_1 b_5}{b_1}$	. . . . .
$d_1 = \frac{c_1 b_2 - b_1 c_2}{c_1}$	$d_2 = \frac{d_1 b_3 - b_1 c_3}{c_1}$	$d_3 = \frac{c_1 b_4 - b_1 c_4}{c_1}$	$d_4 = \frac{c_1 b_5 - b_1 c_5}{c_1}$	. . . . .
. . . . .	. . . . .	. . . . .	. . . . .	. . . . .

Usually, examining the stability of a dynamic system, we have to find out the limits of change of one or more structural parameters (coefficients of strengthening, permanent time etc.), at which the system retains its stability. There arises the problem of ascertainment of the limits of the area of stability in the plane of two parameters upon which depend the coefficients of differential equations (6-9) and the coefficient of characterizing equation.

STAT

Let, for example, the curve 1 on Fig. 6-1 bound the limit of the area of stability in the plane of two parameters δ, μ and let all the points located within this area have the values δ, μ, at which the system is stabled.

The value δ and μ must be chosen in such a way, as to secure some reserve of stability and a quick fading away of the transitory process in the system.

Ya. Z. Tsypkin and P. V. Bromberg introduced the term "degree of stability", which is applied to designate the least quantity of the essential parts of roots of characterizing equation, or, in other words, a distance h from a hypothetic axis on the plane of roots (Fig. 6-2) to the nearest root of characterizing equation (L.25). In a number of instances, the degree of stability can approximately characterize the rate of fading away of the transitory process.

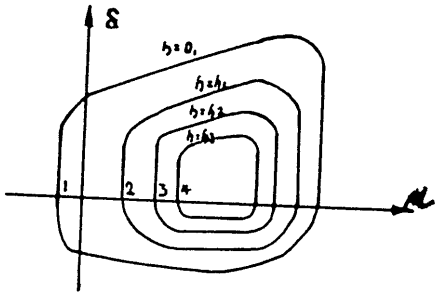


Fig. 6-1.

The right choice of structural parameters is facilitated by the construction within the area of stability of lines corresponding to values δ and μ, at which the degree of stability has the values $h=h_1, h_2, \ldots$ At the border of the area of stability the degree of stability h=0 (Fig. 6 - 1).

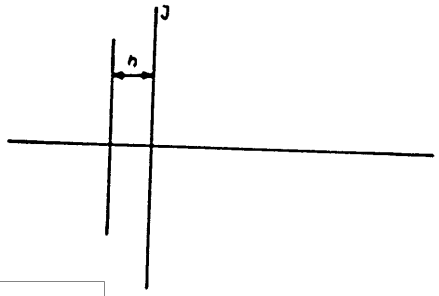


Fig. 6-2.

STAT

 STAT

The line connecting the points possessing an equal degree of stability h can be determined by the same criteria which determine the boundaries of the area of stability, provided that we apply the initial equations

$$\begin{aligned}
\frac{dx_1}{dt} &= (a_{11} + h)x_1 + a_{12}x_2 + \ldots + \\
&\quad + a_{1n}x_n; \\
\frac{dx_2}{dt} &= a_{21}x_1 + (a_{22} + h)x_2 + \ldots + \\
&\quad + a_{2n}x_n; \\
&\ldots\ldots\ldots\ldots\ldots \qquad (6\text{-}12) \\
\frac{dx_n}{dt} &= a_{n1}x_1 + a_{n2}x_2 + \ldots + \\
&\quad + (a_{nn} + h)x_n,
\end{aligned}$$

which differ from equations (6-9) only in that, that to every coefficient a_{ii} was added a positive number h.

6-2. Scheme of General Program Calculation of Areas of Stability and Lines of Equal Degrees of Stability in a Plane of Two Parameters:

The program of calculation described below cuts down to a minimum the volume of preparatory work required prior to introduction of a calculation task into the computer. All that is left to be done by the operator, is to make up a linear system of differential equations of the first order, in its usual form:

$$\begin{aligned}
\frac{dx_1}{dt} &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n; \\
\frac{dx_2}{dt} &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n; \\
&\ldots\ldots\ldots\ldots\ldots \qquad (6\text{-}13) \\
\frac{dx_n}{dt} &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n
\end{aligned}$$

and determine the numeral values of coefficients of these equations, or dependency (which can be non-linear) of coefficients on two parameters δ and μ, in whose plane the boundary of the area of stability

STAT

is under construction. The rest of the work of determination of coefficients of a characterizing equation, of the boundaries of the area of stability and of lines of equal degrees of stability is left to the computer itself.

Using a computer with fixed comma, we must introduce into the expressions of equations (6-13) appropriate scale multipliers, in order to prevent the quantities δ and μ, within the vital range of changes of parameters, from getting out of the expanse of values

$$-1 \leqslant \delta \leqslant +1 \qquad -1 \leqslant \mu \leqslant +1. \qquad (6\text{-}14)$$

The area (6-14) of the plane of parameters δ and μ is subdivided into equal squares with sides $\Delta\delta = \Delta\mu = \Delta$, which usually are chosen as being equal to 0.05 or 0.1 (Fig. 6 - 3).

The calculation of the boundary of the area of stability is performed as follows: all the knot points of the network depicted on Fig. 6 - 3 are examined in some succession and the system's stability is examined at every given point at the values of δ and μ corresponding to that given point. The automatic process of calculation of the boundary of the area of stability by the computer consists of two stages:

a. Search of the first point (the summit of the square) located within the area of stability;

b. A run along the border and determination of coordinates δ, μ of the square summits adjacent to the boundary from within. The areas's boundary is constructed by these summits, or, should a higher precision be desired, by the points located at a distance of a half length of a square, as shown of Fig. 6 - 3. The areas 's boundary runs among the neighboring summits of squares located within and outside of the area.

The search of the first point is performed from below upwards, along the vertical lines of the network (Fig. 6 - 3) and stability is successively examined separately for every network's knot point.

STAT

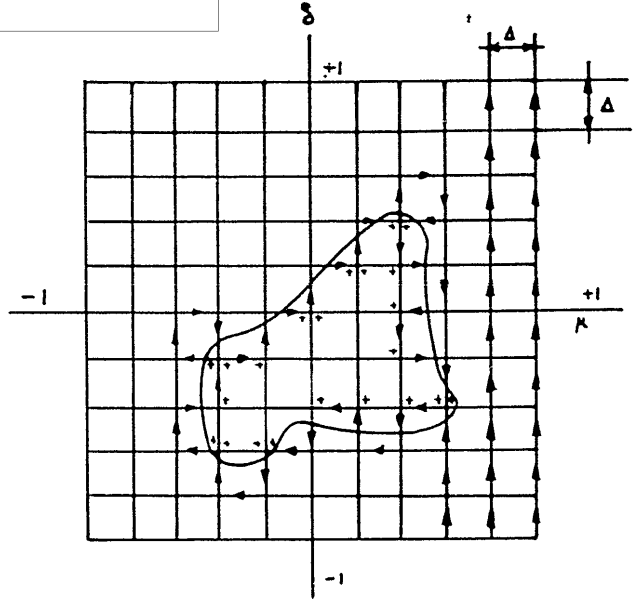


Fig. 6-3

As a rule, the calculation begins from the farther right lower point ($\delta = -1, \mu = -1$). If that point is located outside of the area of stability, the parameter δ gets an accretion and the stability is examined once more, and so on, until δ becomes equal to +1. After this the calculation is made for the point on the next vertical line to the left, beginning with the lower point, and so on.

The run along the boundary of the area of stability commences as soon as the first point located within the area of stability has been found.

The run, by the square summits, is made as follows: At every step of the run, one parameter δ or μ gets an accretion $\pm \Delta$ and the system of stability is examined. If at the given step of change of δ or μ the area's boundary is crossed and the point gets into the area of stability, the subsequent steps are made in such a way as to run along the square sides anti-clockwise (considering the direction of step at which the area's boundary was crossed). Conversely, if at a given step the boundary is crossed and the point got out of the area of stability, then the parameters δ and μ are changed in such a way, as to run clockwise along the square sides. Thus, the direction of run in the elementary square is changed at every crossing of the boundary of the area of stability. At every boundary crossing (at entering into, or coming out of the area of stability) the values of parameters δ and μ for border points in the area of stability are

STAT

not printed.

The run can be made by other, simpler ways, but the above described method has been accepted as universal, since it permits to determine the areas's boundary also then, when the area has a complex shape, which can exist, for example then, when the coefficients of the characterizing equation are depending on the parameters δ and μ in an unlinear manner.

The calculation of stability for every given point is carried out in the following succession:

a. Ascertainment of numeral quantities of the equations' coefficients (6 -13) when $\delta = \delta_1$, and $\mu = \mu_1$, corresponding to the point under examination.

b. Ascertainment of coefficients of the characterizing equation by way of exposure of the characterizing determinant (6-11).

c. Inspection of execution of the Raus' criterian of stability.

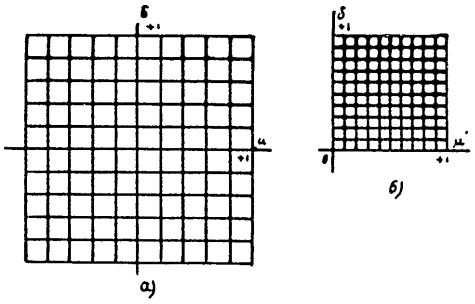


Fig. 6-4

The construction of a logical program's circuit for execution of automatic search and inspection of the boundary gets simplified, if we pass over from such coordinates as δ and μ to such δ' and μ' where the areas of values of δ и $(-1 \leq \delta \leq +1, -1 \leq \mu \leq +1)$ (Fig. 6 - 4, a), we are interested in, would correspond in the plane δ', μ' to square $0 \leq \delta \leq +1, \quad 0 \leq \mu' \leq +1$.

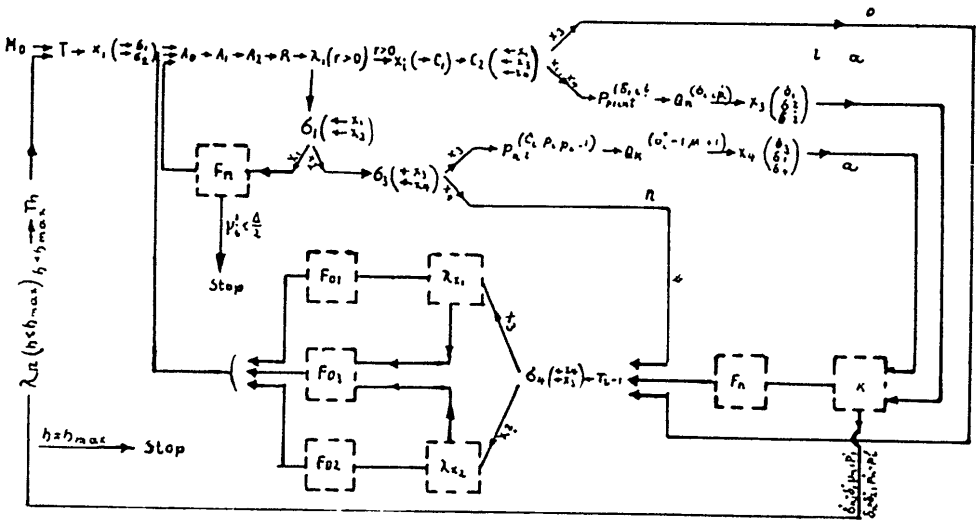
which is located entirely within the first quadrant (Fig. 6 - 4, b).

Coordinates δ, μ are bound to each other by the following dependency:

$$\left.\begin{aligned}\delta &= 2\delta' - 1;\\ \mu &= 2\mu' - 1.\end{aligned}\right\} \quad (6\text{-}15)$$

Step $\frac{\Delta}{2}$ in plane δ', μ' corresponds to step Δ in plane δ, μ.

Coordinates δ', μ' are used by the operators of the logical scheme which control the search and inspection of the area's boundary, i.e. control the selection of the next calculation point in the area of stability. When the coordinates δ' and μ' of a new point are selected, they undergo transformation into corresponding coordinates δ and μ, and the machine performs the calculation of coefficients of the characterizing equation, and checks on the criterian of stability. The boundary of the area of stability gets printed at initial coordinates δ and μ.



The run along the boundary of the area is finished.

Fig. 6 - 5.

A simplified logical program circuit is shown on Fig. 6 - 5 (the program was worked out with the colaboration of G. S. Gekchan). The logical conditions corresponding to operations of conditional transfer are designated with $\lambda[\varphi]$, where φ is the condition determining the ramification of the calculation process. The operators containing logical conditions essential for the search and inspection of the area's boundary but, for sake of brevity not presented separately, are shown surrounded with the dotted line.

STAT

 STAT

A number of ramifications are, in the process of calculation, performed by the operators of unconditional transfer σ'_i. The auxiliary operators x_j, x_{j+1},... place into the operator σ_i the respective command of unconditional transfer and direct the control in the required direction. Designations

$$x_n(\rightarrow \sigma_i) \quad \text{and} \quad \sigma_i \begin{pmatrix} \leftarrow x_n \\ \leftarrow x_m \end{pmatrix} \begin{matrix} \nearrow x_n \\ \searrow x_m \end{matrix}$$

signify that the auxiliary operator x_n prevails upon the ramification operator σ_i and the latter transfers the control in one or another direction, depending on which of its auxiliary operators worked last.

The logical scheme contains a series of operators. The auxiliary operators H_o, T, x_1 prepare the scheme for the beginning of the search of the boundary of the area of stability at the outset of the operation. The H_o places into the work cell the quantity of the degree of stability $h = 0$ (it corresponds to the boundary of the area of stability $h=0$). The T conveys the coordinates δ, μ of the first point, from which the search begins, into the work cells and forms the corresponding values δ' and μ'. The A_o performs the transformation of coordinates δ' and μ' into corresponding coordinates δ_i and μ_i, for the current point of plane(according to formulas (6-15)). The A_1 calculates the coefficients of differential equations (6-13) at current values of δ_i and μ_i.

The operator A_2 calculates the coefficients of the characterizing equation at current values δ_i and μ_i, and the operators R and λ_1, perform the calculations required for the Raus' criterion (according to table 6-1) and check on the agreement of signs in all elements of the left column of the table.

If, at the given δ_i, and μ_i the Gaus' criterian is no performed, then the operator σ_1, previously prepared to search by the auxiliary

STAT

operator x_1, transfers the control to the search operator F_n. The F_n chooses the next paint for search, making one step Δ upwards, along the ordinate, if only that point does not go, at that, out of the area of square on Fig. 6 - 4, b. Otherwise as the next point due to be taken, will be the lower point on the next left vertical line of the network. Once the operator F_n has selected a new point and turned its coordinates over to cell $6'$, it turns the control over to the operator A_0. Once the first point of the area of stability has been found, the operator x_2 changes the content of the operator δ_1, in such a way, that the operator F_n becomes switched off and the circuit of transfer of the control $\sigma_1 - \sigma_3$ becomes switched on. Then follows the inspection of the boundary of the area of stability.

If at that inspection the current point is located within the area of stability, the control goes to the ramification I of the logical circuit. Otherwise, if the current point is located outside of the area of stability, the control goes over to ramification II. Each of the above-named ramifications consists of two circuits a and b. Immediately after the crossing of the boundary, the control goes over to circuit Ia (current point within the area of stability), or to circuit IIa (current point outside of the area of stability) and then, after the next step of inspection, goes over to Ib or IIb respectively and is exercised by that circuit until the next border crossing.

The circuit a of the ramification I contains operator $P^{(\delta_i, \mu_i)}$, that prints the coordinates δ_i, μ_i, of the current point, and operator $Q_R^{\delta_i', \mu_i'}$, that transfers the transformed coordinates of the current point to the work cells of operator K, which exercises the control over the completion of inspection of the area of stability. Besides, this ramification contains an auxiliary operator x_3, that prepares the control circuits in such a way, that after the next step the calculation would go through the circuit Ib (if the next current point like the preceding one is located within the area of stability), or

STAT

through the circuit IIa (if the next point is situated outside of the area of stability). The operator x_3, prevailing upon the operator σ_4, prepares the transfer of the control to the operator that controls the step selection at the run against the clock F_{o1}.

The circuit IIa contains analogical operators, with the only difference being that not the coordinates of the current point, but those of the preceding point i-1st are printed and transferred to operative cells of the operator K. The auxiliary operator x_4 prepares the transfer of the control to the operator that selects the next step while the inspection of the boundary is made clockwise F_{o2}.

From the group of operators Ia (or IIa), the control goes over to the operator supervising the completion of the boundary's inspection, and to the operator of direction F_H. (Editor's Note: The small Russian "n" stands probably for napravlenie - direction). The operator K, that supervises the completion of inspection of the boundary, constitutes a logical circuit, that functions a s follows: At the beginning of the inspection are memorized the coordinates δ'_1, μ'_1, and δ'_2, μ'_2 of the first two different border points. The boundary's inspection ends, when those points are encountered in the same succession as theretofore if the coordinates of two successive border points do not fully coincide with the coordinates of the first two different border points on the inspection route, i.e. if equations $\delta'_k = \delta'_2; \mu'_k = \mu_2; \delta'_{k-1} = \delta'_1; \mu_{k-1} = \mu'_1$, are not valid simultaneously, then the inspection is over and the control goes over from operator K to the operator of direction F_H. The latter determines in which direction the inspection step crossed the boundary. This direction is determined by the logical circuit which compares the coordinates δ'_i, μ'_i of the current point of inspection with the coordinates $\delta'_{i-1}, \mu'_{i-1}$ of the preceding point. Depending on the direction of the boundary crossing (from below upwards, from above downwards, from the right to the left, from the left to the right), the operator in charge prepares the operators of inspection

STAT

F_{o1} and F_{o2}, which will then secure a change of coordinates δ', μ'.

The coordinator T_{i-1} takes over from the F_H and transfers the coordinates of the current point δ'_i, μ'_i, to the work cells of the coordinates of the preceding point . Thus, the memory device continuously retains the coordinates of the preceding point all along the inspection route.

These coordinates are used by operator F_H for the determination of direction of the boundary crossing. If the boundary is not crossed at all, then these operators are switched off; the control through the circuit Ib (or IIb) goes over from σ_2 (or σ_3) to the operator T_{i-1}. If the inspection takes place within the area of stability, then the operator σ_4 turns the control over to operators λ_{k1} and F_{o1}. If the inspection takes place outside of the area of stability, then the operators λ_{k2}, and F_{02} are switched in.

The operators F_{01} and F_{02} change the coordinates δ', μ' in such a way as to secure the route of inspection over the summits of the squares clockwise or anti-clockwise. The operators λ_{k1}, λ_{k2} ascertain whether the required step can be made without getting out of the square shown on Fig. 6-4b. If this can not be accomplished, then the control goes over to operator F_{03}, which performs the changing of coordinates δ', μ' clockwise on the inspection route along the outer contour of the square.

The coordinates δ' and μ' will be changed by operator F_{03} until the area's boundary is crossed again.

From operators F_{01}, F_{02}, F_{03}, the control goes over to A_o and, for the next point, the whole calculation cycle is repeated all over again.

Once the inspection of the boundary of the area of stability is over, the operator K transfers the control to operator T_h, which increases the degree of stability by a quantity Δh. Operator A_1, which calculates the coefficients of the system of differential equations, considers the degree of stability and adds it to the diagonal elements of the matrix of the system of differential

equations.

From operator T_h the control goes over to operator T and the boundary is determined anew, this time with the new degree of stability h. Operator λ_4 examines whether the current value of the degree of stability h has not yet reached the desired maximal reading $h_{maximal}$. When $h = h_{maximal}$, the calculation is over and the computer comes to a stop.

Calculation of coefficients of a characterizing equation by a determinant (6-11) (operator A_a) is performed by the Danilevskiy method (L.22), which is based on the fact that similar matrixes have equal characterizing polynomials, and consists of a reduction by a sequence of similar transformations of the initial matrix of the system's coefficients

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \qquad (6\text{-}16)$$

to a similar matrix, corresponding to a "normal type" by Frobenius:

$$\begin{pmatrix} -p_1 & -p_2 & \cdots & -p_{n-1} & -p_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & & 1 & 0 \end{pmatrix} \qquad (6\text{-}17)$$

It is easy to establish, that the elements of the first line of this matrix are coefficients of the characterizing polynomial

$$N(\lambda) =$$

$$-(-1)^n(\lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \ldots + p_n) \qquad (6\text{-}18)$$

of matrix (6-17), and consequently, at the same time of a matrix similar to it (6-16).

In calculating coefficients of a characterizing polynomial with the aid of a computer, the correctness of the result is

controlled by way of inspection of equality of coefficient p_1 with the sign of the matrix, i.e.

$$-p_1 = a_{11} + a_{22} + \ldots + a_{nn}, \qquad (6\text{-}19)$$

Using a computer with a fixed comma, the calculation of coefficients of a characterizing equation and Raus table is conducted through the use of a calculation subprogram with an artificial floating comma.

The computers having no high-speed memory devices (M-2, "Ural"), operate at a rate of several thousands operations per minute. Calculating the stability of dynamic systems of high orders with the use of such low work speeds, we should have to spent quite a long time on the determination of a characterizing determinant at all points on the route of boundary inspection of the area of stability. In this case it would be expedient to have the above described program subdivided in two:

a. A program of determination of coefficients of a characterizing equation by means of determination of a characteristic determinant (operator A_2); if we have an algebraic expression of dependence of coefficients of the characterizing equation on parameters μ, δ (linear, quadruple, etc.), then this program's part can be made use for getting the numerical values of coefficients determining this dependence (for example, at linear dependence, the coefficients of the characterizing equation are $p_i = A_i + B_i\delta + C_i\mu$. Discovering the characterizing determinant in cases when 1) $\mu = \delta = 0$; 2) $\mu = 0$; $\delta = 1$; 3) $\mu = 1, \delta = 0$, it is possible to find out the coefficients A_i, B_i, C_i);

b. A program of calculation of the boundary of the area of stability by means of the Raus' criterian and inspection of the boundary in such a case, when the coefficients of the characterizing equation and their dependence on parameters δ and μ are given in the numeral form.

 STAT

6-3. Example of Calculation of Static Stability of a Long-Distance Electric Power Line.

We shall examine a simple scheme of electric power transmission shown on Fig. 6-6 (L.8). The generator, by way of a distant transmission line Π, works on the bus of the receiving system ΠC (PS) of infinite capacity (U=const.).

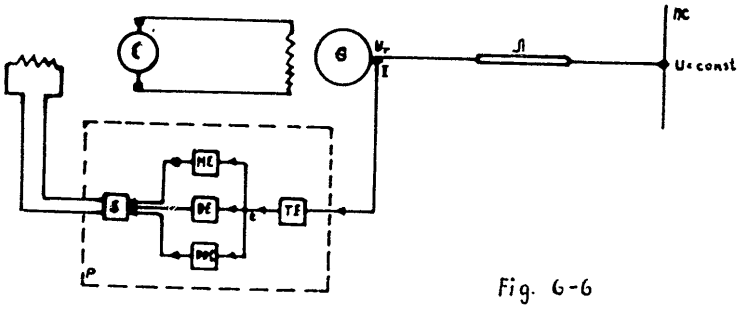


Fig. 6-6

The regulator P (R) regulates the exitation of the generator in the first current function I and in its first and second derivatives.

Under the term "static stability" of electric power transmission we understand the stability at small deflections from the state of equibalance (stability of unperturbed motion).

Let us make up a differential equation of the system under consideration, applying S. A. Lebedev's method (L.18). The movement of the rotor of a synchronous generator is expressed by differential equation

$$M\frac{d^2\theta}{dt^2} + P_d\frac{d\theta}{dt} = P^{max} - P^{эл}, \qquad (6\text{-}20)$$

wherein M is the permanent rotor's inertia;

P_d is the generator's damping moment;

P^{max}, $P^{эл}$ are the capacity developed by the turbine and the capacity of the generator's output into the network.

θ is the angle between the vector of electromotive force of backlash and the vector of voltage of the U receiving system.

For the circuit of exitation of the generator winding we have another differential equation :

STAT

$$T_o \frac{dE'_d}{dt} + E_d = E_{de'} \qquad (6\text{-}21)$$

where E_d is the longitudinal synchronous electromotive power;

E'_d is the longitudinal component of transitory electromotive power of generator;

E_{de} is an electromotive power corresponding to voltage on the exciter's binders;

T_o is the constant of time of exitation winding when the stator's circuit is disconnected.

The output $P^{эл}$ rendered by the generator into the network, depends upon the angle θ and electromotive power E_d (or E'_d). This dependence is determined by the following expressions:

$$P^{эл}_{Ed} = \frac{E_d U}{x_d + x_л} \sin\theta + \frac{U^2 (x_d - x_a)}{2(x_d + x_л)(x_q + x_л)} \sin 2\theta; \qquad (6\text{-}22)$$

$$P^{эл}_{E'd} = \frac{E'_d U}{x'_d + x_л} \sin\theta + \frac{U^2 (x'_d - x_q)}{2(x'_d + x_л)(x_q + x_л)} \sin 2\theta; \qquad (6\text{-}23)$$

The following differential equation expresses the processes occuring in the exciter:

$$T_e \frac{dE_{de}}{dt} + E_{de} = U_{p'} \qquad (6\text{-}24)$$

where T_e is the constant of time of exciter's circuit of excitation;

U_p is the outlet voltage of regulator.

The processes taking place in the regulator are described in the following equations:

 STAT

The transforming element П3 creates at its outlet a constant voltage, proportional in its quantity to the generator's current. The differential equation of this element is

$$T_p \frac{de}{dt} + e = I, \quad (6\text{-}25)$$

where I is the generator's current; T_p is the time constant of the transforming element. If the parameter of regulation is constituted not by current, but by the generator's voltage $U_г$ or angle θ, then it is necessary to place in the right part of the equation (6-25) not I, but accordingly either $U_г$ or θ.

The equation of the sensitive device of the regulator, responding to changes in electric current, including its first and second derivatives, appears in the form:

$$K_o e + K_1 \frac{de}{dt} + K_2 \frac{d^2e}{dt^2} = U_p. \quad (6\text{-}26)$$

Equation (6-26) takes no heed of time constants of differentiating contours. Measures are taken at the designing of the regulator to secure that those time constants be small.

Equations (6-20) - (6-26) must be supplemented by an equation, that would couple the stator's current I with other variables. From the vector diagram the dependence can be easily established that

$$I = \sqrt{I_d^2 + I_q^2} = f(E_d, \theta),$$

where

$$I_d = \frac{E_d - U\cos\theta}{x_d + x_л}; \qquad I_q = \frac{U\sin\theta}{x_q + x_л};$$

$$I = \sqrt{\left(\frac{E_d - U\cos\theta}{x_d + x_л}\right)^2 + \left(\frac{U}{x_q + x_л}\sin\theta\right)^2} \quad (6\text{-}27)$$

In equations (6-26)-(6-27) it is necessary to pass over from the variables themselves to their deviations from the settled values. This results in a system of equations

 STAT

$$M\frac{d^2\Delta\theta}{dt^2} + P_d\frac{d\Delta\theta}{dt} = -\Delta P;$$

$$\Delta P = \left(\frac{\partial P}{\partial\theta}\right)_{E_{d=const}} \Delta\theta + \left(\frac{\partial P}{\partial E_d}\right)_{\theta = const} \Delta E_d;$$

$$\Delta P = \left(\frac{\partial P}{\partial\theta}\right)_{E'_{d=const}} \Delta\theta + \left(\frac{\partial P}{\partial E'_d}\right)_{\theta = const} \Delta E'_d;$$

(6-28)

$$T_o\frac{d\Delta E'_d}{dt} = \Delta E_{de} - \Delta E_d;$$

$$T_\bullet\frac{d\Delta E_{de}}{dt} = \Delta U - \Delta E_{de};$$

$$T_p\frac{d\Delta e}{dt} = \Delta I - \Delta e;$$

$$K_o\Delta e + K_1\frac{d\Delta e}{dt} + K_2\frac{d^2\Delta_e}{dt^2} = \Delta U_p;$$

$$(\Delta I) = \left(\frac{\partial I}{\partial\theta}\right)_{E_d=const} \Delta\theta + \left(\frac{\partial I}{\partial E_d}\right)_{\theta=const} \Delta E_d.$$

In accordance with what we have said in Par. 6-1, the particular derivatives in (6-28) are considered as permanent coefficients. Differentiating the expressions (6-22), (6-23) and (6-27) we can establish values of their particular derivatives

$$P^{Ed}_{\theta} = \left(\frac{\partial P}{\partial\theta}\right)_{E_d=const} = \frac{E_d U}{x_d+x_\lambda}\cos\theta +$$

$$+ \frac{U^2(x_d - x_q)}{(x_q+x_\lambda)(x_d+x_\lambda)}\cos 2\theta;$$

$$P^{\theta}_{E_d} = \left(\frac{\partial P}{\partial E_d}\right)_{\theta = const} = \frac{U}{x_d+x_\lambda}\sin\theta;$$

(6-29)

STAT

$$P^{E'_d}_{\theta} = \left(\frac{\partial P}{\partial \theta}\right)_{E'_d = \text{const}} = \frac{E'_d U}{x_d + x_\lambda} \cos\beta +$$

$$+ \frac{U^2 (x'_d - x_q)}{(x'_d + x_\lambda)(x_q + x_\lambda)} \cos 2\theta; \qquad (6\text{-}29)$$

$$P^{\theta}_{E'_d} = \left(\frac{\partial P}{\partial E'_d}\right)_{\theta=\text{const}} = \frac{U}{x'_d + x_\lambda} \sin\theta;$$

$$I^{E_d}_{\theta} = \left(\frac{\partial 1}{\partial \theta}\right)_{E_q=\text{const}} = \frac{U}{x_q + x_\lambda} \frac{l_q}{l} \cos\theta +$$

$$+ \frac{U}{x_d + x_\lambda} \frac{l_d}{l} \sin\theta;$$

$$I^{\theta}_{E_d} = \left(\frac{\partial 1}{\partial E_d}\right)_{\theta=\text{const}} = \frac{I_d}{(x_d + x_\lambda) l};$$

having introduced the designations

$$x_1 = \Delta\theta; \quad x_2 = \frac{d\Delta\theta}{dt}; \quad x_3 = \Delta E'_d;$$

$$x_4 = \Delta E_{de}; \quad x_5 = \Delta e,$$

the system of equations (6-28) can be expressed in the form

$$\begin{aligned}
\frac{dx_1}{dt} &= a_{12}x_2; \\
\frac{dx_2}{dt} &= a_{21}x_1 + a_{22}x_2 + a_{23}x_3; \\
\frac{dx_3}{dt} &= a_{31}x_1 + a_{33}x_3 + a_{34}x_4; \\
\frac{dx_4}{dt} &= a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 + a_{45}x_5; \qquad (6\text{-}30) \\
\frac{dx_5}{dt} &= a_{51}x_1 + a_{53}x_3 + a_{55}x_5,
\end{aligned}$$

where

$$a_{12} = 1; \quad a_{21} = -\frac{P^{E'_d}_{\theta}}{M}; \quad a_{22} = -\frac{P_d}{M};$$

$$a_{23} = -\frac{P^{\theta}_{E'_d}}{M}; \quad a_{31} = -\frac{P^{\theta}_{E'_d} - P^{E_d}_{\theta}}{T_0 P^{\theta}_{E_d}};$$

STAT

$$a_{33} = -\frac{P'_{E_d}}{T_o P_{E_d}}; \quad a_{34} = \frac{1}{T_o};$$

$$a_{51} = \left[\frac{I'_{E_d}}{P_{E_d}}\left(P'_{\theta E'_d} - P_{\theta E_d}\right) + I_{\theta E_d}\right]\frac{1}{T_p};$$

$$a_{53} = \frac{I'_{E_d}}{P_{E_d}} P'_{E_d}\frac{1}{T_p}; \quad a_{55} = -\frac{1}{T_p};$$

$$a_{41} = \frac{1}{T_e}\left\{a_{51}\left(K_1 - \frac{K_2}{T_p}\right) + K_2 a_{31} a_{53}\right\};$$

$$a_{42} = \frac{K_2 a_{51}}{T_e};$$

$$a_{43} = \frac{a_{53}}{T_e}\left(K_1 - \frac{K_2}{T_o P_{E_d}} P'_{\theta E_d} - \frac{K_2}{T_p}\right);$$

$$a_{44} = \frac{1}{T_e}\left(\frac{K_2 a_{53}}{T_o} - 1\right);$$

$$a_{45} = \frac{1}{T_e}\left(K_o - \frac{K_1}{T_p} + \frac{K_2}{T_p^2}\right).$$

Now let us determine the areas of stability and the lines of equal degrees of stability on the plane of regulation coefficients by the first and the second derivatives K_1, K_2 in the Kuybyshev Hydropower Plant-Moscow transmission line. At a time when the capacitance compensation and the bypass chokes are switched off, the parameters of this system have the following values (in relative units). (The respective parameters are of the electro-dynamic model of the Kuybyshev GES-Moscow transmission line, constructed by MEI. The accepted basis values were P_b = 11.5 kw, U_b = 360 v, x_b = 11.3 ohm (L.7)). U_r = 1; U = 2.42; $x_{d\Sigma}$ = 0.544; $x'd_\Sigma$ = 0.253; $x_{q\Sigma}$ = 0.38; x_Λ = 2.36; P_d = 0; T_o = 5 sec; (T_e = 0.12 sec) M_{sec} = 17 sec (inertia constant attributed to the nominal active capacity).

STAT

Fig. 6 - 7, computed by the M-3 computer, shows the areas of stability and the lines of equal degrees of stability for regimes $\theta = 30^{\circ}; 60^{\circ}; 90^{\circ}; 120^{\circ}$; (at $K_0 = 2.3$).

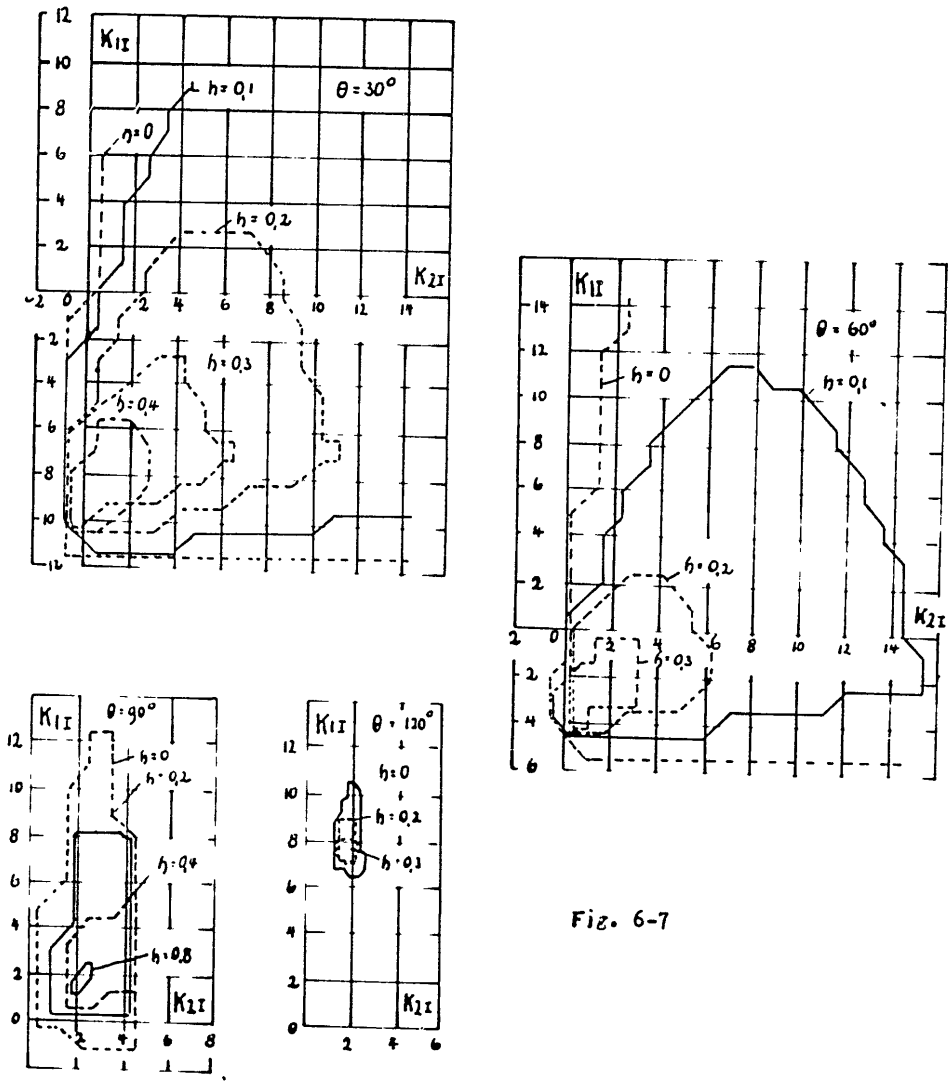


Fig. 6-7

CHAPTER VII

COMPUTATION AND EXAMINATION OF TRANSITION PROCESSES

7-1. Preliminary Remarks

Computations of transition processes constitute a significant element of design work generally and in design and examination of various electrotechnical appliances (electric transmissions, drives, systems of automatic control and regulation, etc.) in particular. In a number of cases, the selection of structural parameters is primarily influenced by demands upon the transition processes within the system under consideration.

Calculation of transition processes consists of integration of differential equations, describing the system's behavior under various initial characteristic conditions and perturbing influences.

Existing dynamic systems may contain elements with non-linear characteristics, variable parameters and/or distributed parameters. In this chapter we shall consider calculation of transition processes in systems capable of being described by common differential equations.

In the overwhelming majority of cases, no solution of differential equations of transition processes can be achieved by analytic methods. For their solution, the mathematics have at its disposal quite an extensive number of methods of numeral solution of differential equations. Widely known have become the methods by Eylor, Runge-Kutta, Adams and others.

Calculations of transition processes in non-linear dynamic systems by hand count calls for a tremendous amount of labor. But, in a great number of cases, such calculations can be performed by calculators of uninterrupted action. This is especially applicable to dynamic systems containing "typical" non-linear elements such as clearance, relay link, zone of insensitivity, etc.

Application of high-speed digital computors for the calculations of transition regimes can be rational, provided that the given system includes a high order and contains elements that are characteristic by their complex non-linear dependences

STAT

Particularly effective is the application of such computers, when on the basis of calculation of transition processes they have to calculate the solutions of logical problems, calling for the selection of optimal structural parameters and control functions. The digital computers are used in all cases when we have to secure a high accuracy of calculation of a non-stationary process.

The amount of time required for preparatory work (programming, adjustment of programs) is considerably shortened, when there are available already adjusted programs for typical problems.

The Runge-Kutta method is widely used in integrating the common differential equations by automatic digital computers, for it secures sufficient accuracy at a great uniformity of calculation. The Adams method is more complicated than the Runge-Kutta method; it requires the aid of some other additional method for the determination of values of variables at some first points of interval of integration, when a higher degree of precision of calculation is needed, or at greater intervals of integration.

Among other things, chapter IV contains a program of integration of a differential equation by the Runge-Kutta method.

In this chapter we shall consider the logical schemes of programming differential integrations by the Runge-Kutta method employing a permanent step and an automatic choice of step. As an illustration of this by a practical example, we shall examine the process of calculation of dynamic stability of a long-distance power line.

7-2. Logical Scheme of a Program of Integration of a System of Ordinary Differential Equations by the Runge-Kutta Method with a Permanent Step.

Through the introduction of auxiliary variables, the system of differential equations expressing the transition processes in a dynamic system, can be reduced to the normal form by Kosha. Let the transition process be expressed by a system of differential equations in the normal form:

$$\left.\begin{aligned}
\frac{dy_1}{dt} &= f_1\ (t, y_1, y_2, \ldots, y_m) \\
\frac{dy_2}{dt} &= f_2\ (t, y_1, y_2, \ldots, y_m) \\
&\cdots\cdots\cdots\cdots\cdots \\
\frac{dy_m}{dt} &= f_m\ (t, y_1, y_2, \ldots, y_m)
\end{aligned}\right\} \qquad (7\text{-}1)$$

at initial conditions $t=t_o$, $y_1=y_{10}$, $y_2=y_{20}, \ldots, y_{mo}$, $f_1, f_2, \ldots, f_m$ - are known, and generally the non-linear functions $t, y_1, \ldots, y_m$.

For the sake of uniformity it is convenient to regard time t in the right portion of equations (7-1) as a dependent variable

$$y_{m+1} = t \qquad (7\text{-}2)$$

and add thereto the differential equation

$$\frac{dy_{m+1}}{dt} = 1. \qquad (7\text{-}3)$$

Considering (7-2) and (7-3), the system (7-1) will assume the following form:

$$\left.\begin{aligned}
\frac{dy_1}{dt} &= f_1\ (y_1, y_2, \ldots, y_n)\ ; \\
\frac{dy_2}{dt} &= f_2\ (y_1, y_2, \ldots, y_n)\ ; \\
&\cdots\cdots\cdots\cdots\cdots \\
\frac{dy_n}{dt} &= f_n\ (y_1, y_2, \ldots, y_n),
\end{aligned}\right\} \qquad (7\text{-}4)$$

where $n=m+1$, $y_n = t$ and $f_n\ (y_1, y_2, \ldots, y_n)=1$.

According to the Runge-Kutta method, when we deal with a system of differential equations of normal form (7-4), the value of functions y_{ir+1} at end of step r + 1 of integration can be found by the values of functions $y_{1r}, y_{2r}, \ldots, y_{nr}$ at the beginning of the integration step through the application of formulas

$$y_{ir+1} = y_{ir} + \Delta y_{ir} \quad (i=1, \ldots, n); \qquad (7\text{-}5)$$

$$\Delta y_{ir} = \frac{1}{6}\ (K_{i1} + 2K_{i2} + 2K_{i3} + K_{i4}). \qquad (7\text{-}6)$$

STAT

 STAT

Runge-Kutta coefficients K_{i1}, K_{i2}, K_{i3}, K_{i4} for the system of equations (7-4)are expressed by the following:

$$\left.\begin{aligned} K_{i1} &= h.f_i\ (y_{1r}, y_{2r}, \ldots, y_{nr}); \\ K_{i2} &= h \cdot f_i\left(y_{1r} + \frac{K_{11}}{2}, y_{2r} + \frac{K_{21}}{2}, \ldots, y_{nr} + \frac{K_{n1}}{2}\right) \\ K_{i3} &= h.\ f_i\ \left(y_{1r} + \frac{K_{12}}{2}, y_{2r} + \frac{K_{22}}{2}, \ldots, y_{nr} + \frac{K_{n2}}{2}\right); \\ K_{i4} &= h.f_i\ (y_{1r} + K_{13}, y_{2r} + K_{23}, \ldots, y_{nr} + K_{n3}), \end{aligned}\right\} \quad (7\text{-}7)$$

where h is the step of integration.

At every step of integration for the determination of Runge-Kutta coefficients four calculations of the values of functions f_1, f_2,..., f_n (7-4) each time with different values of their arguments must be performed in the right parts of equations. It is natural therefore, to subdivide the calculation at every step of integration into four cycles corresponding to the determination of the first, the second, the third and the fourth Runge-Kutta coefficients K_{i1}, K_{i2}, K_{i3}, K_{i4} $(i=1,\ldots, n)$.

Once we have determined all Runge-Kutta coefficients we can find the increment and the new values of variables y_{ir+1}, y_{2r+1},..., y_{nr+1}, by applying the formulas (7-6) and (7-5). However, the program would be more compact if we subdivide the increment y_{ir} into four partial increments, corresponding to the Runge-Kutta coefficients variously weighted,

$$\Delta y_{ir} = \frac{1}{6} K_{i1} + \frac{1}{3} K_{i2} + \frac{1}{3} K_{i3} + \frac{1}{6} K_{i4} \quad (i=1,\ldots, n),$$

and at every step of integration add the obtained partial increments

STAT

to the values of the variables determined at the beginning of integration step. Table 7-1 shows the calculation process of an integration step broken down into separate elements. The calculation process consists of four uniform calculation cycles and one zero-cycle, in which, out of the values of the variables at the final integration step y_{ir+1} arguments are formed (through the transfer of the values of y_{ir+1} into the respective groups of memory cells) to be applied for determination of f_i at the first calculation cycle. Values of the coefficients of weight γ and σ, are subject to cyclic changes depending on the number "s" of the cycle of calculation. Thus, the value assumes, in succession, values, 1, 1, 2, 2, 1, 1, 2, 2 and so on. Values of weight coefficients γ, σ are interconnected by simple correlation

$$\sigma_s = \frac{1}{3}\gamma_{s+1} \quad (s=1, 2, 3, 4). \quad (7\text{-}8)$$

The last line of table 7 - 1 in parentheses contains values of the independent variable "t" for all calculation cycles.

Fig. 7-1 depicts the logical scheme of an integration program for the integration of the system from "n" ordinary differential equations of the first order, constructed in accordance with table 7 - 1. Through the use of table 7-1, we can easily follow the whole work process of the logical scheme of the program.

Five groups of cells A_o, A_1, A_2, C, D, each containing n number of cells, are selected in the memory device. The initial condition of problem y_{io} is placed in the A_o group of cells. By way of addition of partial increments to the A_1 group of cells, new values of variables y_{ir+1} are formed. At every integration step they are turned over to the A_2 group of cells, for storage. These values safekept in the A_2 group of cells, are used in the calculation of arguments for the members of the right parts of equations. New arguments $y_{ir} + \gamma_s \frac{Kis}{2}$ for the calculation of the right parts, are formed in the C group of cells. The D group of cells, at every cycle, at first contains the values of the right parts f_i and then the Runge-Kutta coefficients $\frac{K_{is}}{2}$.

STAT

Table 7-1

	"0" Cycle	First cycle	Second cycle	Third cycle	Fourth cycle
Weight coefficient γ_s		$\gamma_1 = 1$	$\gamma_2 = 1$	$\gamma_3 = 2$	$\gamma_4 = 1$
Weight coefficient δ_s		$\delta_1 = \frac{1}{3}\gamma_2$	$\delta_2 = \frac{1}{3}\gamma_3$	$\delta_3 = \frac{1}{3}\gamma_4$	$\delta_4 = \frac{1}{3}\gamma_1$
Calculation of the right parts $f_i(i=1,\dots,n)$ by the arguments' values obtained in the preceding cycle		$f_i(y_{1r}, \dots y_{kr} \dots y_{nr})$	$f_i\left(y_{1r}+\gamma_1\frac{k_1^1}{2}, \dots, y_{nr}+\gamma_1\frac{k_n^1}{2}\right)$	$f_i\left(y_{1r}+\gamma_2\frac{k_1^2}{2}, \dots, y_{nr}+\gamma_2\frac{k_n^2}{2}\right)$	$f_i\left(y_{1r}+\gamma_3\frac{k_1^3}{2}, \dots, y_{nr}+\gamma_3\frac{k_n^3}{2}\right)$
Calculation of the Runge-Kutta coefficients K_i^s $(i=1,\dots,n;\ s=1,2,3,4)$		$\frac{k_i^1}{2} = \frac{h}{2} f_i$	$\frac{k_i^2}{2} = \frac{h}{2} f_i$	$\frac{k_i^3}{2} = \frac{h}{2} f_i$	$k_i^4 = \frac{h}{2} f_i$
Calculation of arguments by which the f_i will be determined in the next cycle	y_{ir}	$y_{ir} + \gamma_1 \frac{k_i^1}{2}$	$y_{ir} + \gamma_2 \frac{k_i^2}{2}$	$y_{ir} + \gamma_3 \frac{k_i^3}{2}$	—
Calculation of partial increments		$\delta_1 \frac{k_i^1}{2}$	$\delta_2 \frac{k_i^2}{2}$	$\delta_3 \frac{k_i^3}{2}$	$\delta_4 \frac{k_i^4}{2}$
Addition of partial increments to the values of variables at the beginning of the integration step y_{jr}		$y_{ir} + \delta_1 \frac{k_i^1}{2}$ $\left(t_r + \frac{1}{6}h\right)$	$y_{ir} + \delta_1 \frac{k_i^1}{2} + \delta_2 \frac{k_i^2}{2}$ $\left(t_r + \frac{1}{6}h\right)$	$y_{ir} + \delta_1 \frac{k_i^1}{2} + \delta_2 \frac{k_i^2}{2} + \delta_3 \frac{k_i^3}{2}$ $\left(t_r + \frac{5}{6}h\right)$	$y_{ir} + \delta_1 \frac{k_i^1}{2} + \delta_2 \frac{k_i^2}{2} + \delta_3 \frac{k_i^3}{2} + \delta_4 \frac{k_i^4}{2} = y_{i,r+1}$ $\left(t_r + h\right)$

The transfer of values of the variables from one group of cells to another, the formation of new arguments of Runge-Kutta coefficients as well as addition of partial increments occure within "n" number of cycles. The control over these cyclic operation is exercised by the counter of number of cycles (cell p).

Before the beginning of every cycle a zero is placed into cell p, and after the completion of every cycle (for example, after every transfer of y_{ir} from cell A_{1i} to cell A_{2i}) a unity is added. The number thus formed in cell p is used for the transformation of commands of the given cycle, so that in the next cycle the operations would handle quantities provided with index i + 1. The control of completion of the cycle operations is exercised by way of comparison of quantities n with p (operation of conditional transfer).

The operator of formation of cyclically-changing weight coefficients y_s, can be realized, for example, by the following group

STAT

 STAT

of commands: (this method is used in the program worked out by ITM and VT of the AN SSSR)

m + 1 — „1" β β

m + 2 + γ β γ

m + 3 ⩽γ„3" m + 5

m + 4 —γ„2" γ

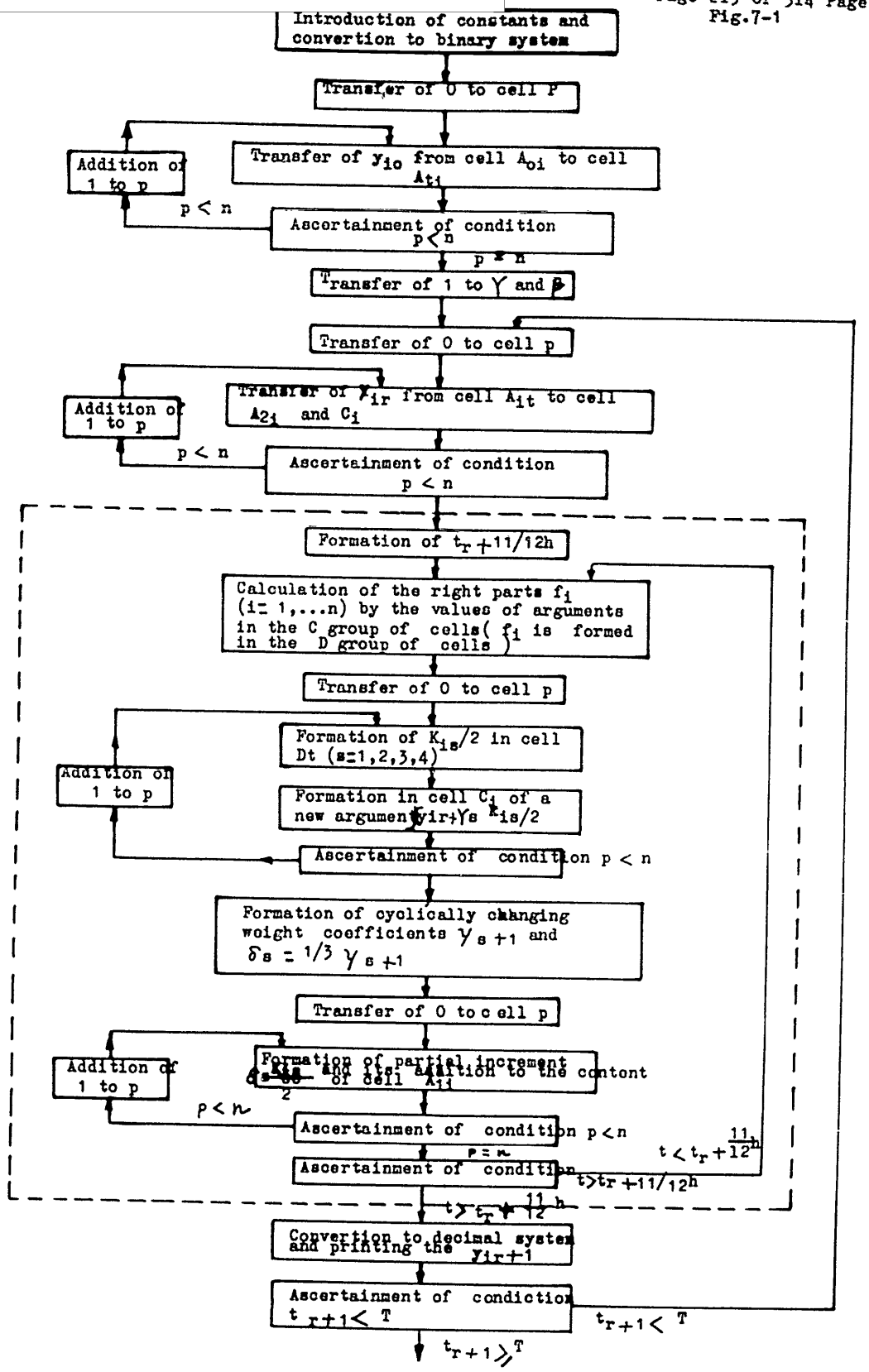
m + 5 . . . . . .

At the beginning of the program cells γ and β are each receiving a unity. The reader can establish without difficulty that in the cell γ, the operator of the cyclically changing weight coefficient successively forms values 1,1, 2, 2, 1, 1, 2, 2, and so on.

STAT

STAT

Fig.7-1



UNCLASSIFIED

STAT

STAT

At the beginning of every integration step, quantity $t_r + \frac{11}{12} h$ is formed, which is used for controlling the final calculation step, i.e. the end of four calculation cycles shown on table 7 - 1. For this, the quantity $t_r + \frac{11}{12} h$ undergoes a comparison with the independent variable values at the end of every cycle (see the last line of table 7-1). After the fourth cycle, $t = t_r + h$, and the control passes to the operator that prints the result of the calculation.

Then the condition $t_{r+1} \angle T$, (wherein T is the prescribed interval) is subjected to inspection. If $t_{r+1} < T$, then the next integration step is made. Everything is ready for that, since the cells γ and β presently contain unities, and the new values of variables y_{ir+1} are now located in those cells, which at the beginning of the preceding step had contained the quantities y_{ir}. The new values of y_{ir} are passed to the argument cells used for calculation of the right parts of equations and the calculation cycle for one integration step is repeated.

If $t_{r+1} \geqslant T$, then the calculation is completed and the machine comes to a stop.

A program corresponding to the scheme shown on Fig. 7 - 1 is a standard program, usable for integration of a system of differential equations (7-4) at any order of n and with an arbitrary form of functions f_i. For every concrete problem only the program's portion dealing with calculation of the right parts of equations would have to be constructed anew.

7-3. A Logical Scheme of a Program of Integration of a System of Ordinary Differential Equations by the Runge-Kutta Method with Automatic Choice of Step.

The choice of value of an integration step of differential equations is influenced by considerations of securing the desired degree of precision of calculation and by reduction of time needed for execution of the calculation process by the machine.

By diminishing within certain limits the integration step, we

STAT

enhance the solution's precision. However, the total number of steps to be made during the prescribed interval of integration would be increased and, consequently, the time needed by the machine for the execution of calculation would be increased, too. In order to achieve the integration with the prescribed precision in a minimal time, we must, at every part of the process, make calculations with a maximal possible step, that can satisfy the prescribed precision. Where the dependents are subject to abrupt changes, the desired step should be small, whereas where the process flows on smoothly, the desired step can be considerably increased, without cutting of precision of calculation (this problem is discussed more fully in chapter 9). For the integrating of differential equations by automatic digital computing machines, such programs are widely used which ensure an automatic choice of the size of integration step, and the desired precision of calculation.

An automatic choice of a step can be made as follows: at every integration step, the values of variables y^{h}_{ir+1}, calculated by the step h (full step), are compared with the values of variables $y^{h/2}_{ir+1}$ obtained after two integration steps by h/2 half steps. If

$$\left| y^{h}_{ir+1} - y^{h/2}_{ir+1} \right| < \varepsilon \quad (i=1,\ldots,n), \qquad (7\text{-}9)$$

where ε is a quantity indicating the calculation's precision, then the values $y^{h/2}_{ir+1}$ (obtained by half-step)are taken for the final result and the following integration step is made thereupon.

If the condition (7-9) is not observed, then the machine discards the obtained result and returns the calculation by step h. The half-step h/2 is then taken for a full step and the result obtained after one step of integration by step h/2 is compared with the result obtained after two steps of integration by step h/4, and so on. For this reason, at every step of integration it is necessary to retain in the memory device the values of the variables which they had at the beginning of the given full step, and the result of the first step of integration by the half-step. If, during the preceding integration step the step was not divided in two, then the next

STAT

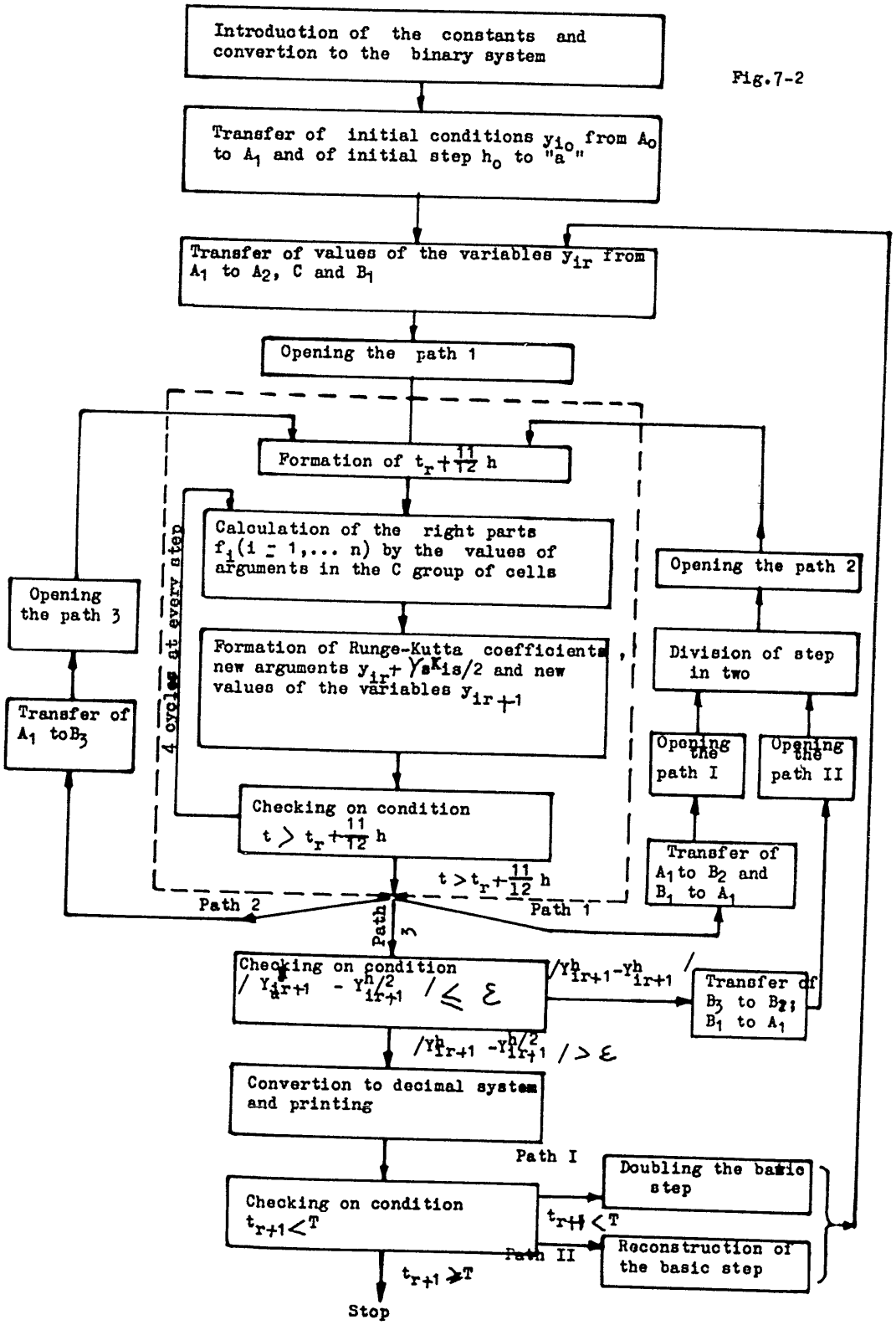
integration step becomes twice as large. Otherwise, its value remains unchanged.

Fig. 7 -2 presents a simplified logical scheme of a program with automatic choice of step. It does not show the auxiliary operators, which, with the help of counter p, control the n-action cycles. The portion surrounded with the dotted line fully coincides with the corresponding section of the logical scheme shown in greater detail on Fig. 7 - 1.

For the program with automatic choice of step, in add tion to groups of cells A_0, A_1, A_2, C and D, used for the same purpose as in the program with a permanent step (7-1), three more groups of cells B_1, B_2 and B_3, each consisting of "n" number of cells, are selected additionally in the memory device.

The B_1 group of cells is used to hold the values of the variables for the beginning of new full step. The cells of this group are used for reconstruction of initial values of the variables at step reduction, when the semi-step was taken for full step and it was necessary to make two steps of integration by step h/4. Although, the values y_{ir} for the beginning of the full step, at passing over from full step to semi-step, might be transferred from A_2 group of cells, for sake of uniformity they are taken now from this (B_1) group of cells. The B_2 group of cells contains the values of the variables obtained by the full step. The B_3 group of cells stores the values of the variables obtained after one half-step. At step deduction these values are taken for the result obtained by the full step.

Fig.7-2



The calculation cycle for every new whole integration step starts with the transfer of the new values of the variables y_{ir} obtained during the preceding resulting step in the A_1 group of cells, to the A_2, C and B_1 group of cells. Then a corresponding command of unconditional transfer with a dotted line is sent to the end of the group of operators surrounded ↓ whereby route I in the control program transfer is opened up.

This is followed by calculations that correspond to one step of integration (full step). At this, according to table 7-1, the group of operators surrounded with the dotted line, operates four times. Upon completion of these calculations the values y^{h}_{ir+1} obtained in the A_1 group of cells are transferred to the B_3 group of cells for storing, whereas the values y_{ir} placed beforehand in the B_1 group of cells are transferred to the A_1 group of cells. The latter values constitute the initial values for half-step.integration. The next operation is the opening up of route 1 (preparation of doubling the step in the next calculation cycle), followed by the division in two of the current step and opening up of route 2. Then the first step of integration is executed by half-step. Through route 2, the control passes to the operator which transfers t e values $y^{h/2}_{ir+1}$, obtained in A_1 group of cells to B_3 group of cells, for storage. Then route 3 of the control transfer is opened up and the second step of integration is made by half-step, whereupon in the A_1 group of cells new values of the variables $y^{h/2}_{ir+1}$ are formed.

If the condition (7-9) is fulfilled, then after the printing and examination to the effect that the prescribed interval of integration has not yet been passed, the basic step is doubled, whereupon the whole calculation cycle for the new step is repeated all over again. If, however, the condition (7-9) is not met, then the values $y^{h/2}_{ir+1}$, obtained after one half-step integration and stored in the B_3 group of cells, are transferred to B_2 group of cells, and are taken for the result as if obtained by a full step.

The initial conditions for the given integration step y_{ir} are reconstructed in A_1 group of cells. Route II of the control transfer is opened (the operator that doubles the basic step is switched off), the current step is again divided in two and the route 2 is opened. Then two integration steps are made by step h/4, and so forth, until the condition (7-9) is met.

Execution of integration with an automatic choice of step complicates the calculation program, but at the same time it increases the precision and permits for the prescribed precision the application of a maximal possible step, which expedites the calculation process within the prescribed degree of precision.

Performing a numeral integration of differential equations, it should be kept in mind, that an inaccuracy in determination of values of the variables y_{i1}, y_{i2}, y_{i3}, ... (i=1,..., n) can create a systematically increasing error at the following steps of integration. Therefore, even after a relatively small number of steps the found solution pretty well coincides with the exact solution, both those solutions can be wide apart after a considerable increase of integration intervals and a number of steps.

In such cases, when we apply the Runge-Kutta method with a permanent step without having preliminarily examined the influence of the step upon the precision of calculations for the given problem, we usually assume, that (Page 135) the number of integration steps should not exceed 200 - 300. During an application of calculations with an automatic choice of step, the number of steps may be increased to 600 - 800.

For the selection of quantity ε, which limits the value of the difference of the variables obtained through the use of the full step and half-step (condition 7-9)), we can employ the formula

$$\varepsilon = 15 \frac{\Delta}{N}. \quad (7\text{-}10)$$

where Δ is the admissible error in calculation; N is the number of steps.

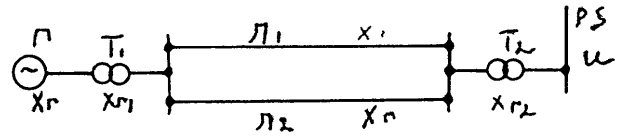
The schemes shown on Fig. 7 - 1 and 7 - 2 are characteristic by their clearness, yet they are not optimal from the standpoint of economy of memory cells and the reduction of the number of operations needed in the calculation. Thus, in the programs with an automatic choice of step, the values f_i of the right members of equations found by values of the variables y_{ir} at determination of K_{i1} for the full step, can be retained in the memory device and made use of in calculations of K_{i1} at the first integration step by the half-step.

7-4. Calculations of Dynamic Stability of Long-Distance Power Transmissions.

Application of high-speed digital computers for calculation of transition processes can be demonstrated by an example of calculation of dynamic stability of long-distance power transmissions.

In case of non-linear dynamic systems, to which the system of power transmission examined in the preceding chapter partially belongs the calculation of stability by equations of first approximation is good only then, when the deflections of variables from their values at a settled regime are small. Meanwhile, in the exploitation of power transmissions the occurance of sharp and abrupt changes of the work regimes (short-circuits, disconnection of lines, generators, etc) is not a rarity at all. The dynamic stability of power transmissions, i.e. the absence of dropping out of synchronism at sharp and abrupt disturbances of its work regime, is judged by the characteristic of transition processes in the system that take place after the influence of some characteristic (with respect to force and location of application) disturbances.

Let us, following the example of P. S. Zhdanov (L.9), examine a simplified qualitative picture of transition process in power transmission after disconnection of one of the two parallel lines. Fig. 7 - 3 shows a simplified scheme of power transmission, where Γ is generator, T_1 and T_2 transformer, Π_1 and Π_2 are two parallel, three-phase transmission lines, C are buses of the receiving system.



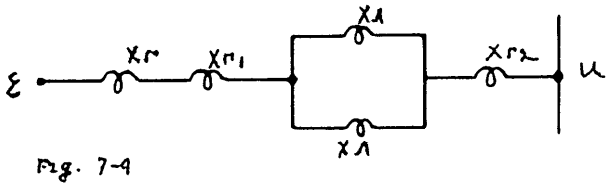


Fig. 7-4

Let the capacity of the receiving system ΠC (PS) be many times greater than the capacity of the power transmission. In this case, the magnitude and the phase of the voltage vector U of the receiving system remain unchanged at all work regimes. Neglecting the active resistance and capacities, we can make up a scheme of substitution for the examination of power transmission, depicted on Fig. 7 - 4, where the generator, the transformers and the lines are depicted in the form of corresponding reaction resistances. From the transmitting end on the scheme is applied the electromotive force of the generator E. The quantities E, U and all reactances are reduced to the same voltage range.

The total reactance of power transmissions

$$x_c = x_T + x_{T1} + \frac{x_A}{2} + x_{T2}. \quad (7\text{-}11a)$$

If however, one of the parallel lines(Π_1 or Π_2) is switched off, the reactance of the system increases, assuming the meaning

$$x_c = x_T + x_{T1} + x_A + x_{T2}. \quad (7\text{-}11b)$$

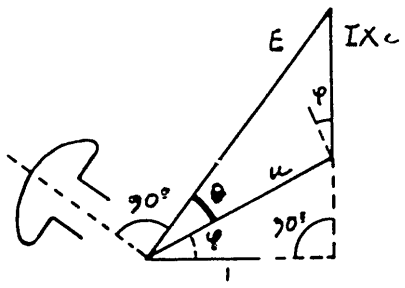


Fig. 7-5

Using the vector diagram of power transmission (Fig. 7 - 5), we can easily obtain the expression of the active capacity P being transmitted by the line to the receiving system:

$$P = UI \cos\varphi.$$

Taking notice that

$$\frac{Ix_c}{\sin\theta} = \frac{E}{\sin(90^\circ + \varphi)},$$

we get

$$P = \frac{EU}{x_c} \sin\theta. \qquad (7\text{-}12)$$

In the system of relative units, the expression (7-12) shows the capacity of a three-phase system. At the fixed state of E generator's e.m.f. and U's receiving system, the active capacity P, transmitted into the receiving system by the power line, depends exclusively on the angle between the generator's e.m.f. and the voltage at the receiving end. The angular characteristic of capacity (Fig. 7 - 6), according to (7-12), has the form of a sinusoid.

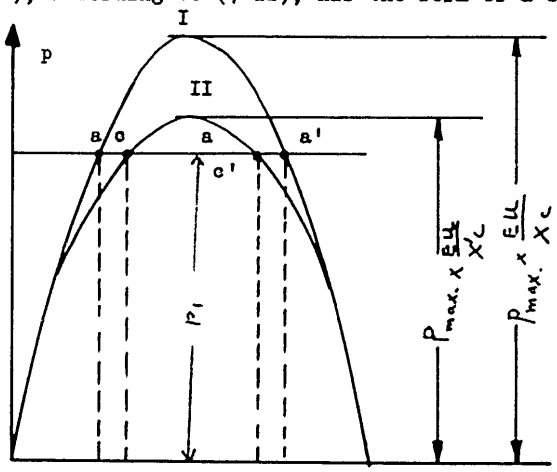


Fig.7-6

With the increase of angle θ the capacity transmitted toward the receiving system at first grows, then subsides. We may examine now the characteristic I on Fig. 7 - 6. Let the turbine develope an output P_o, where $P_o < P_{maximal.}$ Then, the characteristic I has two points a and a', at which the turbine's momentum is counterbalanced by the generator's momentum. Yet, the equibalance can be stable only at point a. Point a' on the delcining curve indicates unstable

equibalance.

The conception of dynamic stability can be explained by an example of deflection of one of the parallel lines depicted on Fig. 7 - 3. Prior to that deflection of the line, the system's characteristic is determined by curve I (Fig. 7 - 6). If the turbine develops an output of P_o, then the work regime is marked on that curve with point a.

Upon disconnecting of one of the lines, the reactance of the system grows and the power transmission gets another characteristic (curve II) with a smaller amplitude.

At the first moment following the disconnection, the angle θ retains its size and the work regime point is shifted to point b on curve II.

At this, the power of the generator drops and the surplus momentum of the turbine accelerates the generator's rotor. The angle θ grows and the work regime point undergoes a shift upwards from point b on the curve II. At this, by inertia, the system passes point c, which corresponds to the angle $\theta = \theta_{cm}$, at which moments of the turbine and of the generator become equal. The surplus moment changes the sign at the point c, once the generator's power becomes higher than the turbine's power and the rotor is dragged. The system is dynamically stable, if the process of dragging will have been over prior to reaching point c', for example at point d. Thereupon the angle θ will decrease again and the work regime point will move from d to c. The rotor's oscillations will gradually subside and a new value of angle $\theta = \theta_{cm}$ (Fig. 7 - 7 a) will set in.

If however, in the process of dragging the angle will exceed the critical value $\theta = \theta''_{kp}$ (point c'), the surplus moment of the turbine will rise anew, angle θ will steadily grow and the machine will drop out of synchronism (Fig. 7 - 7b). This instance corresponds to derangement of dynamic stability. The dynamic stability of a system can be evaluated by the character of the transition

STAT

The definition (7-12) and the characteristic on Fig. 7 - 6 are correct only to some degree. Practical calculations of dynamic stability call for taking into account the non-uniform e.m.f. of the generator, active resistance in the transmission line, the presence of patent rotor poles and other factors.

In connection with the construction in our country of powerful hydroelectric power plants and long-distance transmission lines, ensuring the dynamic stability in powerful synchronous generators transmitting electric power over long distances, acquires particular significance. The static and the dynamic stabilities of long-distance transmissions can be ensured through the application of quick-action excitations permitting considerable boost of voltage in the rotor's rings, use of excitation regulaters provided with measurement organs, which react not only upon changes in the generator's voltage (at times upon the derivative voltages, as well), but also upon the current value of a generator's current and its derivatives the angle θ and its derivatives.

One calculation example of static stability of electric transmission was examined in the preceding chapter. Calculations of dynamic stability of electric transmissions enable us to work out correct formulations of demands toward the synchronous generators, line parameters, excitation systems, equipment and instrumonts. They also enable us to determine the types and capacities of required quick-action switches and protective gadgets, the required values of built-up and of "ceiling" value of excitation voltage and to find correct ways of regulation of excitation.

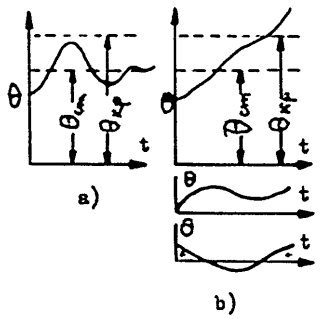


Fig. 7-7

Wide application of excitation systems and regulators, ensuring intensive boost of excitation, paved the way to the introduction of self-synchronization of generators that dropped out of synchronism, to take place after disconnecting of the short circuit section. Therefore, the calculation of dynamic stability must, at present, not only give an answer as to whether or not a disturbance of synchronism is likely to occur, but also be capable of determining the character of the settled regime that sets-in after the dynamic transition (reconstruction of synchronism or setting-up the asynchronous run). Such calculations must be made taking into account the effect of the generator's damper winding, the limited capacity of the receiving system, the non-linear factors and limitations prevailing in regulators of excitations and velocity, the degree of saturation in the excitation system and in the generator itself. We can solve such problems with the use of high-speed computors.

Below we shall examine a less complicated problem of calculation of dynamic stability in an event, when the stability is judged by the behavior of the system within the first cycle of oscillations following a short circuit. In this case, the synchronous generator feeds into a long-distance transmission line on buses of infinite capacity.

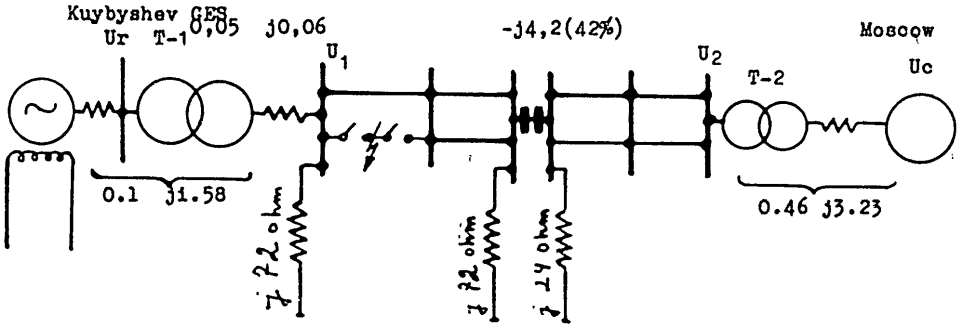


Fig. 7-8

Fig. 7 - 8 shows the principal scheme of power transmission from the Kuybyshev Hydropower Plant to Moscow, of a variant with sequential capacitance of the line inductance (42%) and bypass reactors (L.7). The transmission line consists of two parallel circuits, each subdivided into four sections. From the standpoint of dynamic stability, the most dangerous fault is the three-phase short circuit in the first line section, followed by disconnection of that section (Fig. 7 - 8). Fig. 7 - 9 shows circuits of substitutions for such instances when (a) all the sections are connected, and (b) when the first section of one circuit is disconnected. It also indicates the parameters corresponding to an electrodynamic model of the power transmission from the Kuybyshev Hydropower Plant to Moscow, constructed by the MEI (L.7).

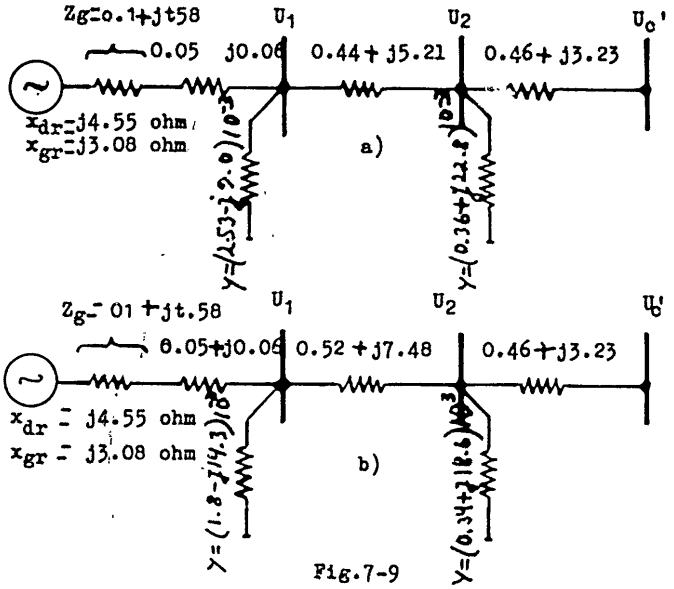


Fig.7-9

The problem states that the transmission line is in operation in a settled regime. At a moment t = 0 a short circuit affects the first section. After a period of time $t_{k.z}$ ($t_{k.z}$ k.z. are the Russian initials for short circuit) the area of the short circuit and the whole first section are disconnected. Allowing t_z time from the occurence of the short circuit, the value of excitation voltage in the generator's excitation winding changes by a jump, in accordance with its "ceiling value" E_{ϕ}. The excitation boost is not switched

STAT

off.

This simplified calculation is intended to determine the influence of the principal parameters of the excitation boost system of the boost lag time t_z and of the duration of short circuit $t_{k.z}$. upon the transmission's dynamic stability. Conditions of this problem are similar to those set forth in the work (L.7), which cites the results of research work done on an electrodynamic model of a power transmission of MEI and on the mechanical integrator ENIN of AS USSR. Now we shall examine the specific features of the solution of a similar problem by an automatic digital computer.

With other major factors being similar, dynamic stability can be appraised by the expression of the largest value of angle θ_{om}, at which the generator still does not drop out of synchronism at a given fault. We can entrust the computer to find out two such boundary values of angle θ_o, which (differing from each other by less, say, than 5^o) satisfy the condition that at the lesser boundary value of the angle the system's stability is retained while at the larger boundary value of the angle it drops out of synchronism. It is convenient to characterize dynamic stability by a half-sum of the values of those angles.

The computer begins the calculation with some value of angle θ_o, that had existed prior to the occurence of fault in the regime. If at that angle value it is found that the system's stability is retained, the machine increased the angle by 5^o and checks the stability again and again, every time increasing the angle until it arrives at such angle value at which the system drops out of synchronism.

In determining dynamic stability at every value of angle θ_o, we have to solve a system of appropriate differential equations (by the Runge-Kutta method, for example), and determine the character of motions of the synchronous generator's rotor during the transition process, that follows the short circuit. A conclusion as to the system's dynamic stability, can be drawn on the basis of the system's

behavior in the first cycle of oscillations, applying the rules mentioned in connection with Fig. 7 - 7. A system is stable, when in the transition process the angle θ at first increases and then decreases, that is to say, when the inequation $\theta_i > \theta_{i+1}$ is valid for two successive time moments t_i and t_{i+1} $(t_i < t_{i+1})$.

When the rotor acceleration in a transition process changes its sign from minus to plus (Fig. 7 - 7b), the generator drops out of synchronism.

The computer is capable of verifying the implementation of the aforementioned conditions and of bringing the integration of differential equations of the transition period to a stop, as soon as one of those conditions is fulfilled. Thereupon, it begins to calculate a new value of angle θ_o and conducts a new set of calculations. To save time, there is no need in getting the results of the integration of differential equations printed: thus the machine prints only the sought value of angle θ_{om}.

Fig. 7 - 10 illustrates the logical scheme of an operator performing the following functions: 1) It verifies the fulfillment of conditions of stability and dropping out of synchronism; 2) it brings to a stop the process of integration of differential equations for the given value of initial angle θ_{ok}, once one of those conditions is met; 3) it calculates the next value of the initial angle $\theta_{ok+1} = \theta_{ok} \pm 5^\circ$, by selecting the sign dependent upon the system's stability at the value of angle θ_o from which the entire calculation was started; 4) calculates and prints the value θ_m. At the beginning of the program containing this operator, the commands that open up the route, I, II and III (Fig. 7 - 10 must be introduced.

The relative motions of a synchronous generator's rotor operating through long-distance transmission line on buses of infinite capacity, can be calculated by the application of the following differential equations (L.7):

$$1)\quad \frac{d\theta}{dt} = \omega;$$

$$2)\quad \frac{d\omega}{dt} = \frac{M_T - M_э}{M_0};$$

$$3)\quad \frac{d\Psi_в}{dt} = E_в - i_в r_в;$$

$$4)\quad \frac{dt}{dt} = 1;$$

$$5)\quad M_э = [E + i_d (x_d - x_{p.c})] i_q; \quad (7\text{-}13)$$

$$6)\quad i_q = \frac{x_{p.c}}{r^2 + x_q x_{p.c}} U \sin\theta +$$

$$+ \frac{r}{r^2 + x_q x_{p.c}} (E - U\cos\theta);$$

$$7)\quad i_в = \frac{\Psi_в - E}{x_{p.p}};$$

$$8)\quad i_d = \frac{x_q}{r^2 + x_q x_{p.c}} (E - U\cos\theta) -$$

$$- \frac{r}{r^2 + x_q x_{p.c}} U \sin\theta;$$

$$9)\quad E = F(i_в - i_d),$$

wherein θ — is the angle between vectors of equivalent voltage U of the receiving system and the e.m.f. of generator's lost motion ω - is the speed of the relative rotor motion; Ψ_μ — is the interlineage of rotor winding; M_t is the turbine rotation moment; $M_э$ — is the electromagnetic moment of the generator; M_0 — is permanent inertia, i_d and i_q — are the longitudinal and transverse stator current components i_μ — is the driving current; E_f — is the voltage of the generator's excitation winding; r — is the active resistance of the transmission line, transformer and generator, comprising the additional losses; $x_{пр}$ — is the resistance of rotor dissipation; $x_{p.c}$ — is the

STAT

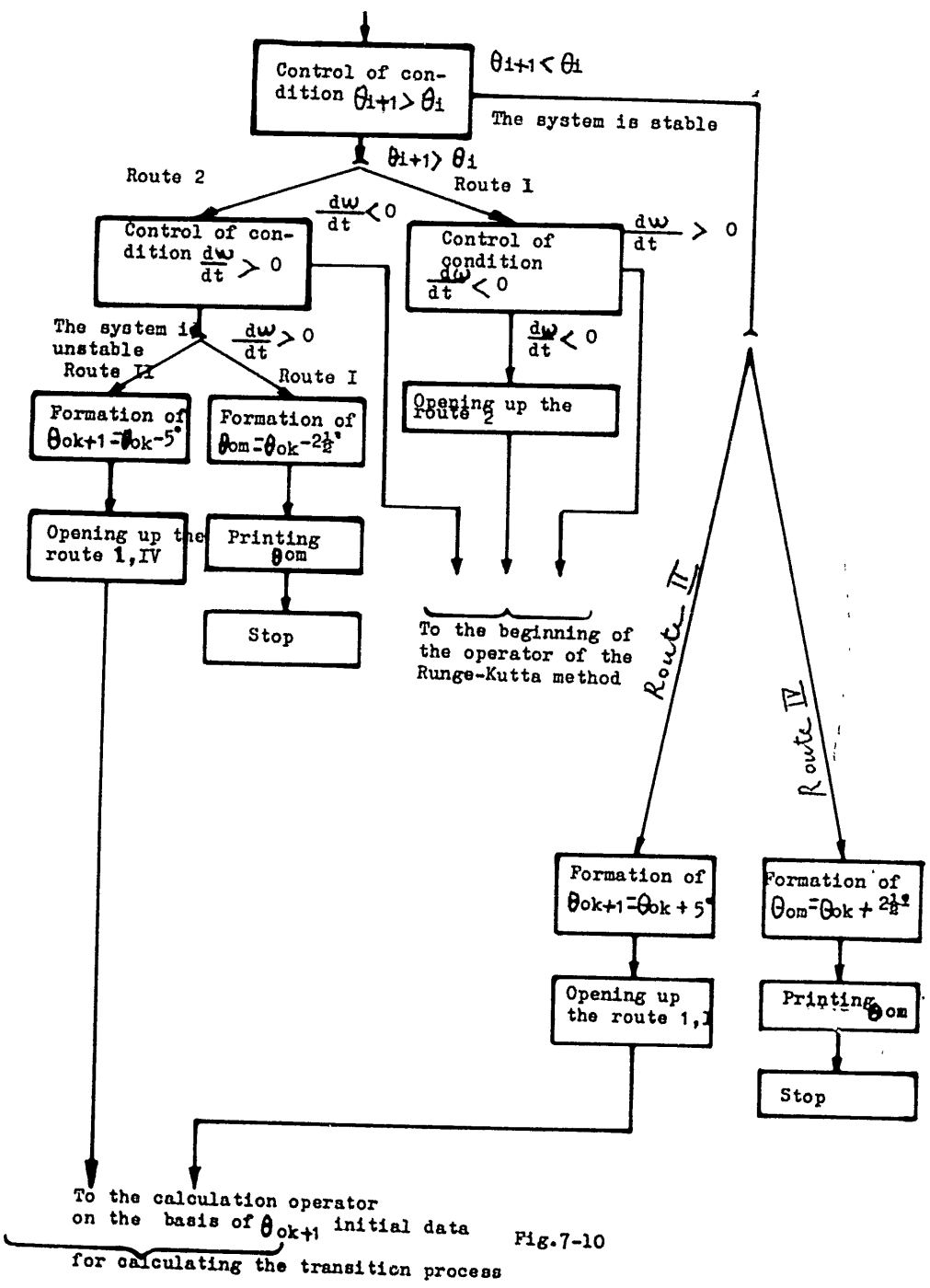


Fig.7-10

resistance of dissipation of the generator's stator, including the transmission's inductive reactance $x_{p.c} = x_{p.c\,T} + x_{\mathcal{L}}$; x_q — is the full generator inductive reactance of the transverse axle, including the transmission's inductive reactance $x_q = x_{q\,T} + x_{\mathcal{L}}$; E — is the longitudinal component of the e.m.f. corresponding to the longitudinal component of magnetic flux in the gap; $r_в$ — is the resistance of the excitation winding. A vector diagram for this system is shown on Fig. 7 - 11.

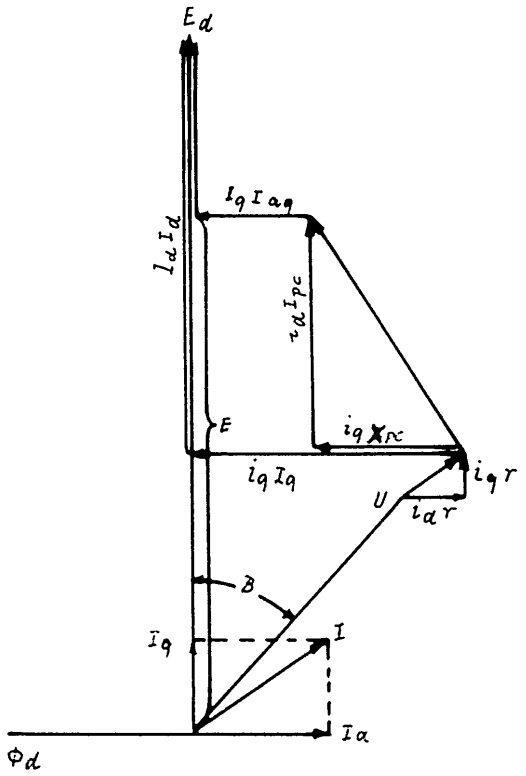


Fig. 7 - 11.

All values, including the time value, are expressed in equations (7-13) in relative unities (angle θ, time t and M_o are expressed in radians). Those (7-13) equations were drawn on the assumption that the transformer e.m.f., induced in the anchor winding along axles d and q during the transition processes, may be ignored. Likewise disregarded was the rotary moment of currents of the rotor's damper winding. Dependence $E = F(i_f - i_d)$, coinciding with the curve of generator's lost motion, takes into account the saturation of the magnetic circuit, in the assumption that it affects nothing else but the value of resistance of mutual inductance along the longitudinal axle.

For success in calculation of transition process $\theta = f(t)$, we must know the parameter values of the block generator-transformer and the parameters of transfer (all reduced to basic quantities), for three work regimes: a) the initial normal regime, b) the short circuit regime and c) the regime subsequent to unshorting the circuit and disconnection of the first section of one of the parallel circuits.

Examining dynamic stability of every initial normal regime ($\theta_o = \theta_{ok}$, $\omega_o = 0$), we must determine the initial condition for $\Psi_f = \Psi_{fo}$ and respective values M_{To} and E_{fo}. For the initial normal regime, these values can be determined by parameters $(r_1)_1$, $(x_q)_1$, $(x_{p.c.})_1$, $(U)_1$.

In this problem the calculation of the transition process consists of three stages:

1. Integration of the system of differential equations (7-13) at interval $0 \leq t \leq t_z$ at the initial conditions that $t=0$, $\omega=0$, $\theta = \theta_o$, $\Psi_f = \Psi_{fo}$ and at parameters $(r_1)_2$, $(x_q)_2$, $(x_{p.c.})_2$, $(U)_2 = 0$ for the short circuit regime.

2. At the time period $t = t_z$, the excitation voltage jumps to a value $E_f = E_{fp}$. The parameters remain unchanged and the system is integrated at interval $t_z \leq t \leq t_{kz}$ where the initial conditions

$t = t_3$, $(\theta)_{t=t_2}$, $(\omega)_{t=t_2}$, $(\Psi_3)_{t=t_2}$ are determined from the solution made at the foregoing section.

3. At the time period $t=t_{k.z.}$ the short circuit is eliminated. At this, a section of the parallel line is disconnected. Beginning from $t=t_{k.z.}$ the system is characterized by the new parameter values:

$$(r_1)_3, (x_q)_3, (x_{p.c})_3, (U)_3.$$

Excitation voltage remains as $E_f = E_{f\phi}$. Initial conditions for the calculation of developments in the system after eliminating the short circuit are derived from the solution of the foregoing section.

The integration of the system of equations (7-13) is conducted by the Runge-Kutta method with automatic choice of step and with the use of a standard program corresponding to the logical scheme depicted on Fig. 7 - 2. For this calculation, $n = 4$.

The functions of the right parts of differential equations are in this form:

$$f_1(y_1, y_2, y_3, y_4) = y_2;$$

$$f_2(y_1, y_2, y_3, y_4) = \frac{M_r - M_3}{M_o};$$

$$f_3(y_1, y_2, y_3, y_4) = E_f - i_f r_f;$$

$$f_4(y_1, y_2, y_3, y_4) = 1,$$

wherein $y_1 = \theta$, $y_2 = \omega$, $y_3 = \Psi_f$, $y_4 = t$, and i_f and M_3 are determined at every step of the integration by the equations (5-9) of system (7-13).

The logical scheme of the operator for the calculation of the right parts of the equations for the given problems is shown on Fig. 7 - 12. It needs no special explanations. We shall, however, make a stop to examine the way of calculation of the e.m.f. E and the currents i_f, i_d, which can be determined through the solution of the system of the three equations 7, 8, 9 from (7-13). (This method was suggested by T. M. Ter-Mikaelyan)

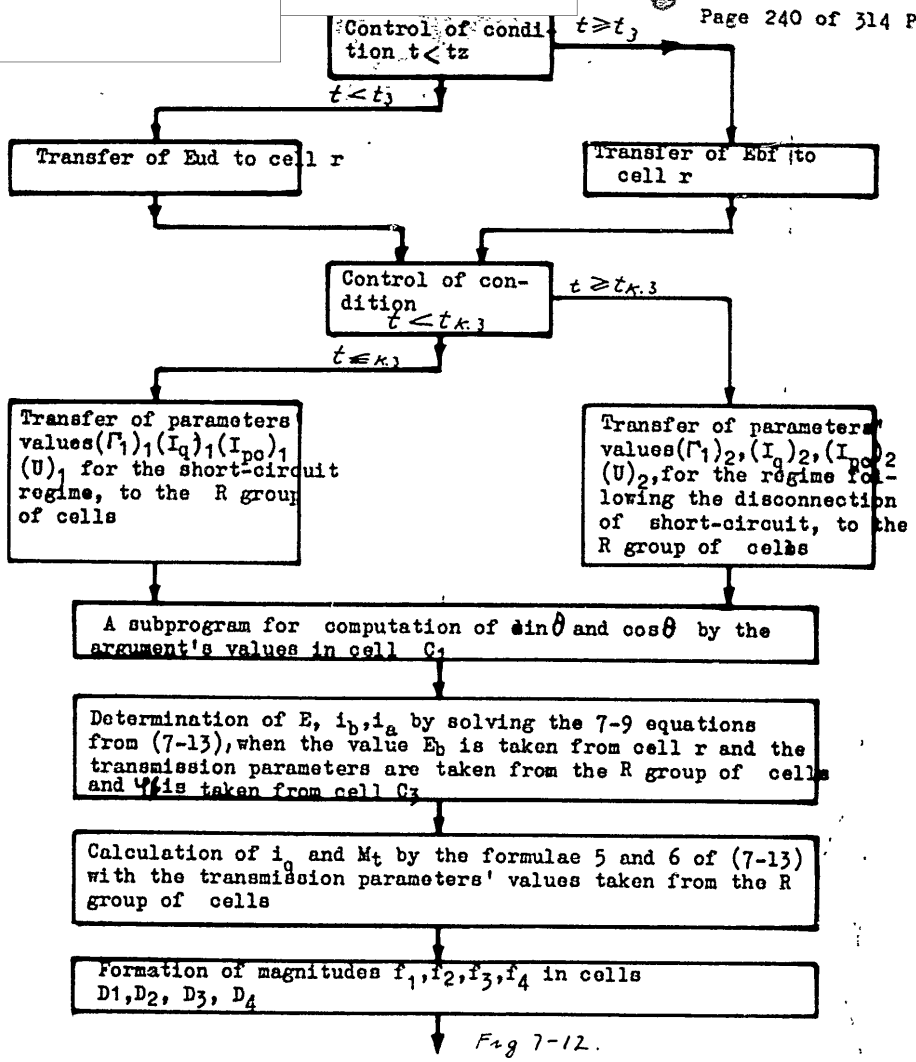


Fig 7-12.

Let us introduce a variable $z = i_b - i_d$. After subtraction of the eighth equation from the seventh equation in the (7-13) system, we have a system of equations

$$\left.\begin{aligned} E &= b - az; \\ E &= F(z) \end{aligned}\right\} \qquad (7\text{-}14)$$

where

$$a = \frac{1}{\frac{1}{x_{p.p}} + \frac{x_q}{r^2 + x_q x_{p.c}}};$$

$$b = \left[\frac{\Psi_b}{x_{p.p}} + \frac{U}{r^2 + x_q x_{p.c}} \times \right.$$

$$\left. \times (x_q \cos\theta + r \sin\theta)\right] \cdot a.$$

The first equation (7-14) is an equation of a straight line, and the second is given by the table of values of the curve of the generator's lost motion (Fig. 7 - 13).

N + 1 value of function F (z) for arguments

$$z_k = k \cdot \Delta z \quad (k = 0, 1 \ldots N)$$

is placed into the memory device, whereas function F (z) at interval z_k, z_{k+1} is replaced by a straight line, crossing the curve's points corresponding to abscissas z_k and z_{k+1}.

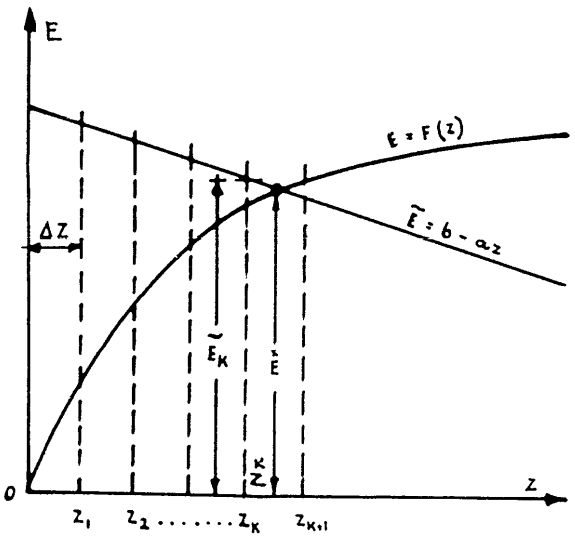


Fig. 7-13

It is easy to show, that in this case the solution E*, Z* of system (7-14), can be had from the expressions

$$E^* = \frac{\tilde{E}_{k+1}(E_k - E_{k+1}) - a \cdot \Delta z \cdot E_{k+1}}{(E_k - E_{k+1}) - a \cdot \Delta z};$$

$$z^* = \frac{b - E^*}{a}, \qquad (7\text{-}15)$$

where $\tilde{E}_{k+1}$, E_k, and E_{k+1} are the values of the coordinates of the straight and of the curve, for the respective values of z (Fig. 7 - 13).

In order to take advantage of expressions (7-15), we must find out the number k of the interval z_k, z_{k+1}, which contains the solution we are after. Performing a succession of examinations, the computer at last finds the minimal k, at which

STAT

$$\tilde{E}_{k+1} - E_{k+1} < 0, \quad (7\text{-}16)$$

then from the memory selects E_k, E_{k+1}, calculates

$$\tilde{E}_{k+1} = b - (k+1)\Delta z \cdot a$$

and determines E* and z*, by the formulas (7-15). After that are calculated:

$$i_\sigma = \frac{\Psi - E^*}{x_{p.p}} \quad \text{and} \quad i_d = i_\sigma + z.$$

CHAPTER VIII

APPLICATION OF DIGITAL COMPUTING DEVICES FOR CALCULATION OF ELECTRIC MACHINES

8-1. General Remarks

Application of small size electronic digital computers for the calculation of a series of electric machines, transformers and some other unique machines, constitutes an interesting field, especially when a great number of calculations of possible variants must be made for the determination of an optimum solution.

The computers can also be successfully applied to obtain data necessary in working out the specification of engineering methods and the calculation of some parameters and characteristics of electric machines (parameters of damper windings of synchronous machines, etc.).

The designing of new series of electric machines includes an extensive calculation work for the determination of optimum correlations of structural parameters and calculation of all new electric machines of a given series.

Until recently, this huge labor-consuming effort of calculation stood in the way of the designers of new series of electric machines and prevented them from performing complete calculatory determinations of optimum variants. The structural parameters used to be chosen after but a limited number of calculation operations on a limited number of variants, with the personal experience and intuition of the designers having a free play.

The appearance of automatic digital computers opened new possibilities.

Beginning with 1956, the NIIEP conducts the calculation of new series of electric machines with the aid of computer M-3. Performing the proof calculations of electric machines, the computers make use of but a part of their capabilities. The calculation of electric machines is fraught with the calculation of a great deal of initial and final data, so that the printing of the results by the computers takes more time than the calculation itself.

Full use of the computer capabilities necessitates such formation of the calculation problem, at which the computer's capabilities would be used not only to perform proving calculations, but also for the determination of the optimum geometrical and winding data of the given electric machines.

A proving calculation by the computer M-3 lasts about one minute. Therefore, the computer performing a great number of calculations envisaged in a given program for various variants of parameters, can automatically find out and "memorize" the optimum variant and then get the machine's calculation list printed.

In the application of computers it is imperative that the calculation lists of electric machines should include all the particular features of calculation performance by the mathematical machine. Magnetization curves must be preset in the form of polynomials. The work of automatic digital computers with fixed commas can be greatly facilitated by utilization of relative unities, which makes the choice of scale coefficients considerably easier.

8-2. Forming of Problem for Calculation

This paragraph with some simplifications examines the forming of a problem which was used for the calculation by the computer M-3 of asynchronous motors from 0.6 to 100 kw, in 1957. (This problem, as well as the list of calculation, was worked out by Professor, Doctor of Technical Sciences T. G. Soroker. The program performing the automatic search of the optimum calculation variant was worked out under the

author's direction, with the colaboration of Yu. V. Mordvinov, Ye. V. Plamod'yalo and V. T. Burmistrov). As the motor's optimum criterion was taken the minimum of the sum C of the cost of a motor's production and the expenditures on electric energy during its service time. At such optimum criterion, the calculation can result in the selection of a variant needing a great amount of copper. Therefore, for control and comparison a second optimum criterion C_b, the minimal cost of a motor's production was introduced, since a motor constructed with the application of this criterion would be most economic with respect to the expenditure of copper.

For every motor of the projected series, the computer must find and print (including the quantities C and C_b) the whole data of optimum variants obtained by the criterion C and the criterion C_b.

The projected motors must satisfy certain requirements with respect to such basic indices as power coefficient $\cos\varphi$, efficiency, excess of temperature in the stator's winding, and others. These requirements narrow down the freedom of choice of the motor's parameters.

At the execution of calculations of asynchronous motors the following factors were subjected to limitation: multiplicity of maximal moment k_m, excess of temperature of the stator's winding θ and power coefficient $\cos\varphi$.

The magnitude of efficiency was not subjected to limitations, for in a motor that has a minimal total of expenditures C, a high magnitude of efficiency is attained automatically, whereas in a motor chosen by the criterion C_b the magnitude of efficiency is limited by the admissible excess of temperature. Limitation of the magnitude of the power coefficient is required only in calculations by the criterion C_b, since the criterion of summary expenditures simultaneously secures a high value of $\cos\varphi$. Separate limitation of the expenditure of copper was not introduced, because of there being no substantiation for the selection of maximal values on this matter. The expenditure of copper was controlled by way of comparison

STAT

with a prescribed outer diameter, the calculations are made to determine the optimum inner diameter, the geometrical features of the grooves and the winding data.

The inner diameter and the geometrical features of the grooves, obtained as a result of such calculations, are taken for application to motors of primary lengths.

The calculation of motors of primary lengths consists in determining the optimum winding data for the inner and outer diameters and the groove geometry obtained by the preceding calculations.

8-3. Mathematical Interpretation of the Problem

Remarks on Linear and Non-Linear Programming

Let us examine a most general case of calculation, when all four quantities the outer stator's diameter D_a, the inner stator's diameter D_i, the inductance in the gap B_δ and the number of effective wires s —are unknown, and must be defined under minimization of the criterial quantities C and C_D while taking into consideration the limitations (8-1).

Mathematically, this problem can be formulated in the following manner. It is imperative to find the values of the variables D_a, D_i, B_δ and s, at which the function of these variables

$$C = F(D_a, D_i, B_\delta, s) \quad (8\text{-}2a)$$

or

$$C_D = F_D(D_a, D_i, B_\delta, s) \quad (8\text{-}2b)$$

assumes feasibly minimal values with the proviso that the limitations imposed upon the values of some functions of these variables are observed:

$$k_M = f_1(D_a, D_i, B_\delta, s) \geqslant k_M^{np}; \quad (8\text{-}3)$$

$$\delta = f_2(D_a, D_i, B_\delta, s) \leqslant \delta^{np}; \quad (8\text{-}4)$$

$$\cos\varphi = f_3(D_a, D_i, B_\delta, s) \geqslant \cos\varphi^{np}. \quad (8\text{-}5)$$

with a motor selected by the criterion C_D, which constitutes the limit of minimal expenditure of copper.

The computer must determine the motor parameters, at which the summary expenditure C (in another instance the cost of production C_D) has a feasibly minimal value, provided that the multiplicity of the maximal moment k_m and the power coefficient are not less and the excess of the stator's temperature is not higher than the prescribed maximal values, i.e., that the limiting conditions

$$\left.\begin{aligned} k_M &\geqslant k_M^{np}; \\ \theta &\leqslant \theta^{np}; \\ \cos\varphi &\geqslant (\cos\varphi)^{np}, \end{aligned}\right\} \qquad (8\text{-}1)$$

where k_M^{np}, θ^{np}, $(\cos\varphi)^{np}$ are the prescribed maximal values carried into effect.

For the calculation of asynchronous motors by the computer a specially worked out list of calculations (in relative unities) was used, which determined simply all the principal measurements (length of active part, sizes of grooves, etc), the winding data and the motor characteristics as functions of the four quantities of: 1) the outer stator diameter D_a; 2) the inner stator diameter D_i; 3) the inductance in the gap B_δ; 4) the number of effective wires in a groove s.

Thus, the search for the machine's optimum variant consists in determining the values of Da, D_i, B_δ and s.

The projecting of a series of machines includes the unification of diameters and stamps. Therefore, the calculation of optimum values of all four independent quantities Da, D_i, B_δ and s of every overall dimension of the machine is made only for such a machine, which heads the list of mass produced machines.

The outer diameter obtained by such calculation is taken as the standard diameter for all machines of a given size.

Then, for motors of other length and other revolution speeds

STAT

Considering, that from the physical standpoint the variables D_a, D_i, B_0, and S can assume only positive values, four more limitations must be added to the limitations (8-3), (8-4), and (8-5), to wit :

$$D_a \geqslant 0; \quad (8\text{-}6)$$

$$D_i \geqslant 0; \quad (8\text{-}7)$$

$$B_5 \geqslant 0; \quad (8\text{-}8)$$

$$s \geqslant 0; \quad (8\text{-}9)$$

In mathematical literature, the problem of minimization (or maximization) of function

$$z = F(x_1, x_2, \ldots, x_n) \quad (8\text{-}10)$$

under conditions that the variables are capable of assuming only positive values

$$x_i \geqslant 0 (i = 1,2,\ldots,n) \quad (8\text{-}11)$$

and the limitation of values of some functions of these variables

$$f_k(x_1, x_2, \ldots, x_n) \geq b_k \ (k = 1, \ldots m), \quad (8\text{-}12)$$

was named the problem of "programming". At the present time, this class of problems attracts great attention, particularly so in connection with the calculation by the computers of optimum solutions in the field of economics.

The inequalities (8-11) and (8-12) separate in the first quadrant of n-fold space, an area of admissible values of x_1, x_2,..., x_n.

When function F and the functions of limitation are linear, then the problem is called "linear programming". In such case the values of x_1, x_2,..., x_n must be defined at which the linear function

$$z = \sum_{i=1}^{n} a_i x_i \quad (8\text{-}13)$$

has a maximal (or a minimal) value, provided that

$$x_i \geqslant 0 (i = 1, \ldots, n) \quad (8\text{-}11)$$

and the linear (limiting) inequalities

$$\sum_{i=1}^{n} c_{ik}x_i \leq b_k \quad (k=1, 2,\ldots,m). \quad (8\text{-}14)$$

are observed.

The problem of linear programming is frequently encountered in research work.

In a problem of linear programming, in accordance with the linear inequalities (8-11) and (8-14), the boundaries of an area of admissible values of variables are formed by planes. In this case, the point corresponding to maximal (or minimal) possible value of a linear function, is always located on one of the vortexes of a polyhedron formed by the planes of limitation. (In some instances function z (8-13) can assume a maximal (or a minimal) value on an edge or on a facet of polyhedron).

The methods of solution of the linear programming are based on the cognizance of this fact.

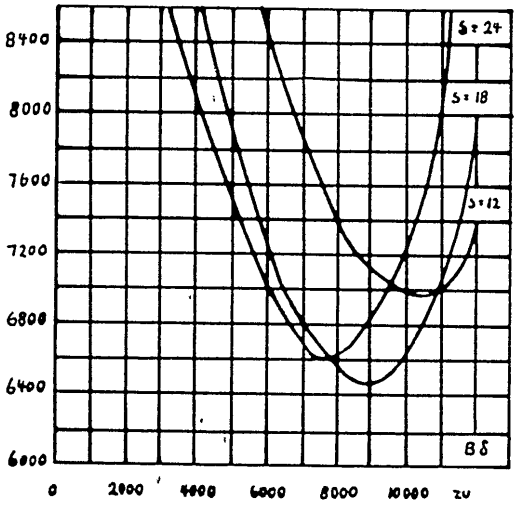


Fig. 8-1

In handling a problem of linear programming, one can say nothing in advance about the location of the point, that imparts to function z a possibly maximal (or minimal) value. It can be located at any place within the plane of the spherical polyhedron formed by the planes of limitation, or inside it.

At the present time, the general methods of solution of problems of the linear programming have not yet progressed beyond the study stage.

In the above-formulated problem of determination of optimum parameters of an asynchronous motor, the minimized functions and the functions of limitation are non-linear. Fig. 8 - 1 illustrates this by a set of curves of dependence of summary expenditures C for the motor A94-2 upon the inductance in the clearance of various numbers of effective wires in the groove. Thus, our problem constitutes a problem of the non-linear programming.

8-4. The Way of Search of an Optimum Variant of a Motor by an Automatic Digital Computer. The Logical Scheme of Programm

Taking into consideration, that this problem contains only four variables and that the calculation of a motor with prescribed values of D_a, D_i, B_δ and s by the machine M-3 lasts less than 1 minute, the determination of optimum parameters D_a, D_i, B_δ and s of a motor, can be performed in the following manner:

The first quadrant of a four dimensional space of variables D_a, D_i, B_δ, s, is dissected by parallel planes, D_a = const; D_i=const, with a due step by D_a and D_i.

Every thus obtained section constitutes plane of variables B_δ and s, with corresponding values of D_a = const and D_i= const. Upon these planes (within the first quadrant) a grid is plotted, with steps ΔB and Δs along the axles B_δ and S (Fig. 8-2).

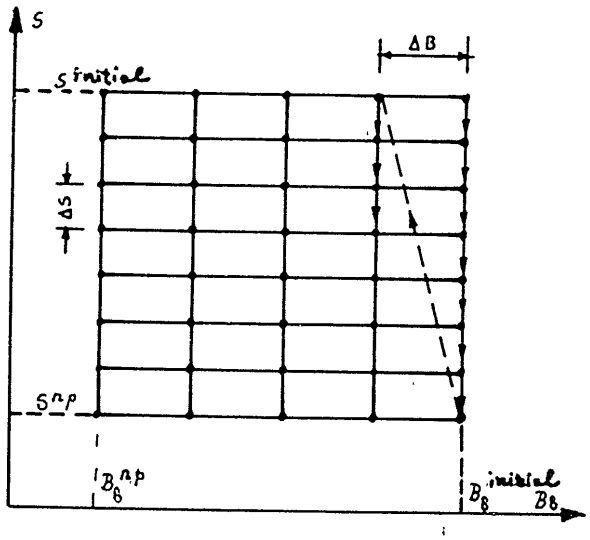


Fig. 8-2

The computer can be made to make the round of all main points of the grid of the plane B_δ , s for some values of D_a and D_i, and at each point perform the calculation of the motor, at the corresponding values of B_δ , s, D_a and D_i.

Doing that, the computer must discard all such variants for which the prescribed limitations (8-1) for the values k_m and $\cos\varphi$ are not observed, at the same time selecting and memorizing only the best variant obtained by the criterion C and the best variant obtained by the criterion C_D. Having completed the search of the best variants on one plane B_δ , s the computer begins the search on another plane B_δ, s, corresponding to the other values D_a and D

STAT

Such a search method can, within steps D_a, D_i, B_δ and s, determine the optimum values of parameters D_a, D_i, B_δ and s, at which the criterion C (or C_D) within the limitation (8-1), has the possibly minimal magnitude.

Thus, the search of an optimum solution of the problem consists of searches of the best variants in the separate planes of B_δ, s, corresponding to varied, but fixed values of D_a and D_i.

Choice of the plane B_δ, s, is convenient because in all the variants of the calculation of the motors, including the calculation of a prescribed outer diameter of a stator or with a prescribed D_a and D_i, the search for the best possible solution of plane B_δ, s remains.

At first it may seem, that the number of points the computor has to cover is inadmissibly large. However, searching for the optimum variant of the motor, we always can determine certain sound limits of changes of B_δ, s, and D_a, and D_i.

The search for the optimum variant on the plane B_δ, s, (Fig. 8 - 2) begins with the upper right point of the grid. Beginning with that point, the computer automatically examines all the points located on the same vertical line, moving from the top downwards. Once the number of effective wires in the groove (line load) has reached the utmost minimal value, the search goes over onto the next left vertical line.

For speeding-up the search, the calculation for a given point stops as soon as it turns out that even one of the conditions (8-1), is not fulfilled whereupon the computer proceeds to calculate another point.

For the same purpose, the search on a given vertical line stops and proceds onto the next vertical line: 1) if the condition $\cos\varphi \geq (\cos\varphi)^{np}$, was valid for some point of a given vertical line and then, at the downward movement, ceased to be valid. 2) if all the limitation conditions (8-1) for any point of a given vertical line were valid and then, upon crossing over to another point of the same

vertical line, it shows that at least one condition is limited.

In other words, from the standpoint of physical notions it has been presumed, that the area of plane B_δ, s, on which the condition (8-1) is valid, is simply connected and "convex". Obviously, the same is true with regard to the four-dimensional region (D_a, D_i, B_δ, s).

In the course of calculation, the computer collates the motors for which the limitations (8-1) are valid, by the summary cost C and per motor production cost C_D, and memorizes the parameters D_a, D_i, B_δ, s, for motors with a minimal cost C and with a minimal cost C_D.

Let us designate the summary cost and the cost of a motor at the current point of search, as C_i and C_{Di}, and the same for the preceding point of search as C_{i-1} and C_{Di-1}.

At a simultaneous implementation of inequalities

$$C_i > C_{i-1};$$

$$C_{Di} > C_{Di-1}$$

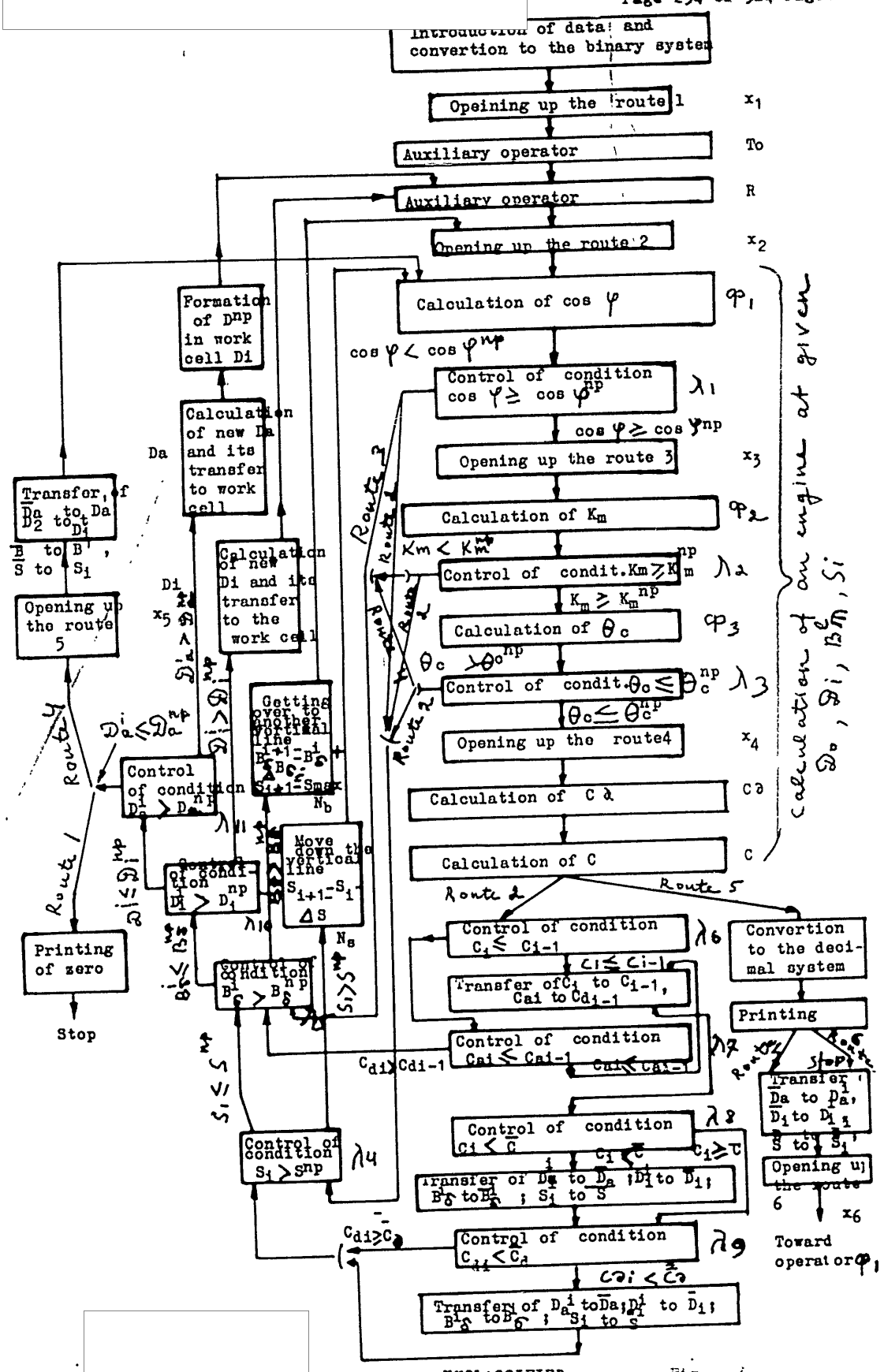
the search on a given vertical line of the grid stops and proceeds onto the next vertical line.

For the calculation of an asynchronous motor, the computer is charged with the following initial values: 1) capacity; 2) the number of poles; 3) the number of parallel branches; 4) the number of stator and rotor grooves; 5) the coefficient of the stator winding; 6) the size of the air gap; 7) the spacing of the stator winding; 8) the width of the slot of the stator's groove; 9) the table of the diameters of the insulated wires; 10) the magnetization curves in the form of polynomials for the separate sections of the magnetic circuit.

In addition, the maximal values of the principal indices of the machine, such as the power coefficient $(\cos\varphi)^{np}$, multiplicity of

maximal rotary moment k_M^{np} excess of stator temperature θ_{np}, as well as the initial and the final values of the line load, and the inductance in the gap and of the outer stator's diameter, are prescribed.

Fig. 8 - 3 shows a logical scheme of the search program of optimum parameters of an asynchronous motor.



UNCLASSIFIED

Fig. 4-3

In the course of search, the respective quantities of the engine's variant that were found to be of the least cost C, are placed into the memory cells $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_\delta$, $\bar{s}$, $\bar{C}$, whereas the quantities of another variant found to be of cost C_0 are placed into the memory cells $\bar{\bar{D}}_a$, $\bar{\bar{D}}_i$, $\bar{\bar{B}}_\delta$, $\bar{\bar{s}}$, $\bar{\bar{C}}_0$. Clearing the route 1, operator x_1, performs the preparation for the printing of zero, a conventional sign signifying that no point satisfying the requirements (8-1) is located within the search area.

For every calculated engine, the auxiliary operator T_0 determines the boundaries of the search area by calculating the initial value of the outer diameter D_a^{uox}, the nominal data of the engine, the initial value of the stator's inner diameter D_i^{uox}, and the maximal values of the diameters D_a^{np}, D_i^{np}, and then transfers the initial data of D_a^{uox} and D_i^{uox} into work cells for D_a and D_i. In addition to this, the same operator feeds a unity into cells $\bar{C}$ and $\bar{\bar{C}}_0$ (the initial state of the program).

The auxiliary operator R, using the given value of D_a and D_i, prepares a search on the B_δ, s plane, and calculates s^{uox}, s^{np} and the grid step Δ B and Δ s. Thus, for the number of effective wires in the groove and the step Δ s, the nearest larger even numbers are taken.

Meanwhile, operator R feeds a unity into cells $\bar{C}_{i-1}$ and $\bar{\bar{C}}_{i-1}$, where the cost data for the foregoing point of search are to be stored and transfers the coordinates B_δ^{uox} and s^{uox} of the first point of search on the given plane to the respective work cells for B_δ^1 and s_1.

Calculation formulas for an asynchronous engine (at the given D_a, D_i, B_δ and s) are broken down into several operators Φ_1, Φ_2, Φ_3, C, C_0. Each operator constitutes a completed calculation stage.

The first three operators are connected with one another by the logical operations in the form of comparison. In the long run, operator φ, determines the power coefficient $\cos\varphi$. Operator Φ_2 calculates torque values and determines the coefficient k_m.

STAT

Operator φ_3 carries out the thermal calculation and determines the excess of stator temperature θ.

Operators C_0 and C, containing in themselves the formulas of economic calculation, find out costs C_0 and C.

Operators λ_1, λ_2, λ_3, verify that the conditions of limitation (8-1) are complied with. If $\cos\varphi < \cos\varphi^{np}$, then, contingent upon the condition that the inequality $\cos\varphi \leq (\cos\varphi)^{np}$ has not yet been fullfilled during the movement along the given grid vertical line, the search is put to a stop and operator λ_1 transfers the control to operator λ_4. A search will then take place on a point with a lesser s.

Once, the inequality $\cos\varphi \leq (\cos\varphi)^{np}$ is fulfilled, the control will go over to operator x_3, which, while opening route 3 will stop the transfer of the control to λ_4 in case the value of s is on the decrease, thus the condition $\cos\varphi \geq (\cos\varphi)^{np}$ is not continued any more.

In this case, the control will be again turned over to operator λ_5 and the search will pass on to a new vertical line.

Operator φ_2 functions then, when the power coefficient is as high as is needed. Operator λ_2 checks the fulfillment of the condition $k_m \geq k_m^{np}$. At $k_m < k_m^{np}$, the control passes on to operator λ_4. When the condition is fulfilled, operator φ_3 starts working.

If the excess of temperature of the stator winding is less than prescribed, the control passes on to operator x_4, if it is greater than prescribed, the control passes on to operator λ_4.

Operator λ_4 compares s_i at the given point of the round with s^{np}. If number s_i has not reached its maximal value, the number of wires in the groove is reduced (downward movement on the grid's vertical line). Otherwise, the control passes on to operator λ_5 which sees to it that the maximal reading of induction B_δ^{np} is not yet reached. As soon as it is, operator λ_5 transfers the search onto a new vertical line of the grid, beginning with the uppermost point s = s maximal.

Transfer to operator x_4 means, that the current point is located within the area of admissible values of variables under the conditions of (8-1).

While cutting out the transfer of the control by route 7 from operator λ_{11}, operator x_4 prepares the printing of the data of the engine variant which was found by the search to be the best. As soon as the first point on the given vertical line, at which the conditions (8-1) are fulfilled is found by the computer, routes 4 are opened. Now, should at the further movement on the given vertical line toward a lesser s the area of admissible values B_{δ}, s (under conditions of (8-1)) be left out, then the search would go over to another vertical line, whereupon the computer proceeds to determine the costs C_D and C. The variants satisfying all limitation conditions are compared with one another, with respect to cost factors. Operators λ_6, λ_7 compare both costs C_{Di} and C_i at the given point with $C_{D.i-1}$ and C_{i-1} of the foregoing point on the same vertical line. If the comparison establishes that both costs at the given point are higher than at the foregoing one, then operator λ_7 transfers the control to operator λ_5, whereby the machine automatically relays the calculation process to another vertical line.

Operators λ_8, λ_9, search for points of the lowest cost. Operator λ_8 compares total cost C_i at the given point with the minimal cost arrived at the foregoing points. This value is found in cell $\bar{C}$. Conditionally upon the cost at the given point C_i being less than the value in cell $\bar{C}$, the coordinates of the current point (D_a, D_i, B , s) are relayed to cells $\bar{D}_a$, $\bar{D}_i$, $\bar{B}$ and $\bar{s}$, whereas the value C itself goes into cell $\bar{C}$.

Operator λ_9 compares the cost of engine $C_{D.i}$ at the given point, with the minimal cost of the engine (in cell $\bar{\bar{C}}_D$), which was arrived at in the course of search at other points.

$D_i < \bar{C}_D$, then the coordinates of the current point and values $C_{D.i}$ are relayed in cells $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_D$, $\bar{s}$, $\bar{C}_D$. If $C_i \geqslant \bar{C}$ and $C_{D.i} \geqslant \bar{C}_D$, then the exclusion of old coordinates does not take place.

Then the control passes on to operator λ_4, and, if it is possible, the movement goes down the vertical line, and the calculation of the engine with the other value of s ensues, etc.

Once the search on the given plane B_δ, s is over, i.e. the maximal reading of induction B_δ is reached, then operator λ_5 relays the control to an operator which, with a preset step, changes the value of the inner diameter D_i; whereupon the search on a new plane B_δ, s, at a changed value of D_i, commences. At this, operator λ_{10} sees to it that the value of the inner diameter D_i has not yet reqched a maximal value. Otherwise, the control goes over to operator D_a, with a preset step, changes the value of outer diameter, and to operator D, which in a respective work cell forming the initial value of diameter D_i. After this search is started in turn in the group of planes B_δ, s, with new values of D_a and D_i, changing with a prescribed step within limits from D_i^{ucx} to D_i^{np}.

The whole process of search for optimum variant of the engine is over as soon as operator λ_{11} discovers, that the outer diameter has reached the maximal value D_a^{np}. By the end of the search, memory cells $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_\delta$, $\bar{s}$ and $\bar{\bar{D}}_a$, $\bar{\bar{D}}_i$, $\bar{\bar{B}}_\delta$, $\bar{\bar{s}}$ contain the coordinates of the best variants of the engine, corresponding to the lowest costs of C and C_D.

Once the search is over, operator λ_{11} transfers the control to a group of operators performing the printing of the calculation data of the optimum variants. At first route 5 is cleared (preparation of circuits of operators performing the printing). The values of D_a, D_i, B_δ and s of the variant of a minimal cost C which were fed into respective work cells and the control passes on to operator Y_1. Then calculation of an engine with optimums D_a, D_i, B_δ and s (by criterion C) is performed again and prints the data of its

calculation list.

Then values of D_a, D_i, B_δ and s of the optimum variant found by criterion C_D are fed into the work cells, the computer performs the calculation of that engine, prints its calculation list and then comes to a stop.

The above-described program occupies 1,100 memory cells. 22 program constants undergo transformation at every calculation of a new engine.

A trial calculation of an engine with given D_a, D_i, B_δ and s, is being made by the computer M-3 in 45-55'. Ascertainment of 40 component data of the calculation list (conversion to decimal system and printing) takes 4 minutes. Search of the best variant of engine on plane B_δ, s, with fixed values of D_a and D_i, lasts, on an average, for about one hour. During that time, the computer performs 60 - 80 calculations of the engine's variants.

When the computer searched for the optimum values of D_a, D_i, B_δ and s, the diameters D_a, D_i assumed 6 various values with a 2.5% step. Thus, the entire calculation of the optimum variant of the engine consisted of searching for the best variant on 36 planes of B_δ, s, corresponding to the varied values of D_a and D_i. Such a search was performed by the computer M-3 in about 36 hours.

CHAPTER IX

SOME INFORMATION ON APPROXIMATE CALCULATIONS

9-1. Theory of Errors

In all practical calculation we deal with numbers which are obtained as a result of measuring various magnitudes such as, e.g. distances, time, weight, etc. Because of limited capabilities of measuring instruments, it is never possible to obtain absolutely exact values of prescribed magnitudes. Conversely, the result is always bound to contain some error, which can be brought to light during a second measuring, when we arrive at a number differing somewhat from the one we obtained before. Only in very rare cases

STAT

do we know the exact values of the magnitudes that enter a formula. Basically this is true with respect to mathematical constants, such as π, e and so on. But, even in such cases, these numbers can not be expressed precisely, for they contain an infinite number of decimals.

Even when we apply but precise formulas, some errors of initially taken magnitudes will go along through the whole calculation and at the close of the calculation we shall get but an approximate answer. Furthermore, some mathematical problems can be solved only after an endless process of trial. Integral calculus, the calculation of derivatives, etc. can serve as an example of such endless processes. Once we can not bring the endless process to an end, we have to stop at some final step within its course and then we are inescapably liable to have an answer containing an error.

At the same time, in the problems of engineering the unknown magnitudes are sought for practical application. Here again, inevitable errors arise in the production process, or in sizes of articles, as a result of inexactitude of initial measuring. Consequently, the attainment of genuine precision of an unknown magnitude has no practical significance and we can be satisfied with an approximate answer. At this, two problems arise; knowing the initial errors of magnitudes we have to determine the error in the answer, and, on the other hand, we have to determine what the initial errors may be in order that we may secure the attainment of the desired precision in the result. In order to be able to answer these questions, we have to define what the concept of exactness of a result actually is.

ABSOLUTE AND RELATIVE ERRORS. If some number a constitutes an approximate value of number A, then the modulus of the difference of

$$|A - a|$$

is called the real absolute error of the approximate number a.

In the majority of cases, the real absolute error is unknown,

STAT

for in order to know it, we should have to know the true value of A. But, as a rule, in conducting a calculation, one can warrant, that in the result of the calculation the value of the committed error will not exceed a certain limit. That certain limit is considered to be a measure of precision measurement and is called the absolute error of the approximate number a. Thus, the absolute error of approximate number a is possibly the smallest Δ number satisfying the inequality $|A-a| \leq \Delta$, i.e. the absolute error is more than, or equal to the true absolute error.

It is evident, that the absolute error does not define the measurement quality well. For example, the absolute error of $\frac{1}{2}$ k Γ at weighing railway cars and bricks indicate quite a different accuracy of measurement. Besides, the absolute error as a rule is a concrete quantity and its value is, therefore, changeable along with the change of unit of measurement. Therefore, for the determination the accuracy of approximate numbers, a special term has been introduced, namely, the "relative error".

Also, the relative error of an approximate number a, is called the relation of the absolute error Δ to number a:

$$\alpha = \frac{\Delta}{a}.$$

The relative error is a dimensionless magnitude, independont of units of measurement. Usually, the relative error is expressed either in per cents (1% = 0.01) or in thousandth ($1^{o}/_{00}$ = 0.001).

THE NUMBER OF CORRECT CIPHERS: There exists a practical and relatively simple method of determining the absolute and the relative errors by way of counting the number of correct ciphers. It is known, that in the decimal scale of notation every number is expressed as a sum of various powers of 10 multiplied by one of the digits 0,1,2,...,9. For instance, $1,023 = 1 \cdot 10^3 + 0 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$. Designating an index of the highest power of 10 included in number a as "p", we have the following expression:

STAT

$$a = a_p \cdot 10^p + a_{p-1} \cdot 10^{p-1} + \ldots +$$

$$+ a_{p-n+1} \cdot 10^{p-n+1} + a_{p-n} \cdot 10^{p-n}.$$

Other than zero, the first from the left cipher a_p is called the first significant or the senior cipher. For example, in the numbers 1,023 and 0.0023 the first significant cipher will respectively be 1 and 2, with "p" in these numbers being equal to 3 and -3 respectively. Every digit standing in a certain place in the decimal notation of number "a", has its special value. The first digit is equal to 10^p, the second 10^{p-1},..., the n-th 10^{p-n+1}. Rounding the number off, we replace all its ciphers, beginning with some cipher, with zeros. The admissible error does not exceed at this the value of the digit standing in place of the last untouched cipher.

Number a is an approximation of quantity A with n being the first correct cipher, provided that the absolute error of the number does not exceed the value of the digit standing in place of the last n-th cipher, i.e.

$$|A-a| < 10^{p-n+1}.$$

For example, number 2.718 expresses number e=2.718281... with four correct ciphers and number 2.7183 expresses e with five correct ciphers. This example illustrates, that the number of correct ciphers may be understood almost literally. Only the last of n "correct" ciphers can be different from the true value, and even then not more than by one. Only some exceptional numbers do not agree with this rule; for example, number 9.999 expresses number 10.000 with four correct ciphers.

The resulting rule therefrom is, if an approximate number a is

expressing number A with n correct ciphers, then the relative error a does not exceed

$$\frac{1}{a_p \cdot 10^{n-1}} .$$

Indeed, by definition the relative error is equal to:

$$a = \frac{\Delta}{a} \leqslant \frac{10^{p-n+1}}{a} < \frac{10^{p-n+1}}{a_p \cdot 10^p} =$$

$$= \frac{1}{a_p \cdot 10^{n-1}} .$$

Wherefrom it follows, that at any value a_p, three correct ciphers guarantee a relative error not exceeding 1%. A number that has n correct ciphers is usually expressed with n significant ciphers, even then, when the last ciphers are equal to zero. For example, if number 3.28 has five correct ciphers, it must be represented in the form 3.280.0.

ERRORS OF ARITHMETIC RULES. Let positive numbers $A_1, A_2, \ldots, A_m$ be expressed by approximate numbers $a_1, a_2, \ldots, a_m$ having absolute errors $\Delta_1, \Delta_2, \ldots, \Delta_m$ and relative errors $a_1, a_2, \ldots, a_m$? It is desired to find the error of the sum

$$a = a_1 + a_2 + \ldots + a_m,$$

which is an approximate value of number

$$A = A_1 + A_2 + \ldots + A_m.$$

Since

$$|A - a| = |(A_1 + A_2 + \ldots + A_m) - (a_1 + a_2 + \ldots + a_m)| \leq |A_1 - a_1| + |A_2 - a| + \ldots + |A_m - a_m|,$$

then the absolute error of the sum Δ does not exceed the sum of absolute errors.

From the correlation

$$\alpha = \frac{\Delta}{a} \leq \frac{\Delta_1 + \Delta_2 + \ldots \Delta_m}{a} = \frac{a_1\alpha_1 + a_2\alpha_2 + \ldots + a_m\alpha_m}{a} = \alpha_1 \frac{a_1}{a_1 + a_2 + \ldots + a_m} + \alpha_2 \frac{a_2}{a_1 + a_2 + \ldots + a_m} + \ldots + \alpha_m \frac{a_m}{a_1 + a_2 + \ldots + a_m}$$

it follows, that the relative error of the sum does not exceed the largest relative error of the items.

Indeed, if a^* is the largest of the numbers $a_1, a_2, \ldots, a_n$, from the forgoing inequality, it follows that

$$\alpha \leq \alpha^* \left(\frac{a_1}{a_1 + \ldots + a_m} + \ldots + \frac{a_m}{a_1 + \ldots + a_m} \right) = \alpha^*.$$

Inasmuch as quantity $\frac{a_j}{a_1 + a_2 + \ldots + a_m}$ is the multiplier of a_1, then the principal influence upon the magnitude of the relative error of the sum is exerted by the relative error of

STAT

the largest of the items. Therefore, there will be in the sum just as many correct ciphers as in the largest item. Adding up several numbers having equal numbers of upper ciphers, it is necessary at first to extract the largest item, and in other items to have all columns standing to the right of the last column of the largest item, discarded. For example, in the sum

$$7728.75 + 370.846 + 0.712813 = 8100.308813$$

only the senior six ciphers can be correct, in view whereof, the summation must be made this way:

$$\begin{array}{r} 7728.75 \\ 370.85 \\ \underline{\quad 0.71} \\ 8100.31 \end{array}$$

Evaluating the error in the difference of two numbers, we must differentiate between two instances. If these numbers have great disparity in value, we may repeat the above-adduced consideration and be convinced of the correctness of the statement, that is the error of the largest number, that is exerting the main influence upon the relative error of the difference. Consequently, the unnecessary columns must be discarded.

Let on the other hand, the minued and the subtrahend having equal numbers of correct ciphers, differ from each other only slightly. If so, then the same absolute error will fall upon the small difference, in view whereof its relative error will be greatly increased.

For example, the absolute error of numbers 150.46 and 150.35 does not exceed 10^{-4}, whereas the relative error of their difference, which is 0.11, can be equal to 10^{-1}, i.e. be increased almost 1000 times. Consequently, when performing calculations, we have to transform the formulas in such a way, as to be capable of finding out the difference of numbers with little disparity in value, without knowing those numbers themselves.

As regards the multiplication and the division of numbers, it

STAT

may be stated that: the relative error of the result of a successive series of multiplications and divisions does not exceed the sum of relative errors of each separate number.

We are not going to demonstrate the correctness of this rule, but want to note that when the number of operations is limited (around 10 operations), then from the above-formulated rule it follows, that the result is less by one-two correct ciphers than the smallest number of correct ciphers in the numbers participating in the operations. However, at a large number of operations, the result can be of a considerably lesser accuracy.

9-2. Solution of Algebraic and Transcendental Equations. Formulating a Problem. Let us have to solve the equation

$$f(x) = 0, \quad (9\text{-}1)$$

where f(x) is some transcendental, or algebraic function. In other words, we have to find out points $x_1, x_2, \ldots, x_n \ldots$, at which function f(x) turns into zero (so called zeroes of function f). Numbers x_i can be true as well as complex. There can be an infinite number of them, like, for example, an infinite number of roots of equation $\sin x = 0$. They may be located in immediate propinquity from one another, and may even have points of contiguity. At times there arises a problem of finding out the smallest positive root of equation (9-1), or of finding out the equation roots at an interval (a,b), and so on.

Our problem is to find out approximate values of roots of equation (9-1). If f(x) is a continuous function, then we shall call point x^* the approximate value of the root of equation (9-1) an absolute error ε, provided that at points $x^* - \frac{\varepsilon}{2}$ and $x^* + \frac{\varepsilon}{2}$, f(x) assumes different signs. A universal numerical method of solution of this problem for arbitrary function f (x) does not exist, so that we have to consider various particular

STAT

cases. The first and the most difficult step in searching for material roots of equation (9-1), is the task of separation of roots. We understand this as searching for two such points a and b, between which only one equation root is located. Once all the roots are disengaged, the remaining difficulty will be the great volume of calculation work.

Some information on distribution of true zeroes of function $f(x)$ can be shed by an examination of its derivative, since, according to Roll's theorem, between two zeroes of $f(x)$ there is situated an uneven number of zeroes of derivative of $f'(x)$. Sometimes, disengagement of roots can be managed by means of physical considerations.

For example, in the problem dealt with in chapter about the determination of frequencies of vibration in turbogenerator rotors, we knew beforehand that all roots of equation $D(\lambda)=0$ were positive and were spaced apart at a distance not less than a fixed number h. Under such conditions, the problem can be solved just the same way as it was solved in paragraph 5-1, that is to say, through finding out such an interval, at whose ends the function has different signs, with subsequent division of that interval in two. This method of division of the interval in two is relatively too laborious even for the high-speed computers.

Presuming that we know two numbers a and b between which there is situated only one equation (9-1) root, we are going to present more expeditions ways of definition of its values.

Iterational Method. If function $\psi(x)$ in some ambient area of root α of equation (9-1) does not turn into zero, and if

$$\varphi(x) = x - \psi(x)f(x),$$

then equation

$$x = \varphi(x) \qquad (9\text{-}2)$$

has in that ambient area only one solution $x=a$.

Assuming that we know a rough value x_1 of root α and want to have it defined more exactly, we name x_1 to be the first approximate and as the second approximate we take:

$$x_2 = \varphi(x_1);$$

as the third approximate

$$x_3 = \varphi(x_2)$$

and so on, until we get at the n-th approximate

$$x_n = \varphi(x_{n-1})$$

etc.

When the sequence x_n coincides, then its limit is nothing else but the root of equation (9-2). Actually, if we designate the limit of this sequence a^*, then

$$a^* = \lim_{n\to\infty} x_n = \lim_{n\to\infty} \varphi(x_{n-1}) =$$

$$= \varphi(\lim_{n\to\infty} x_{n-1}) = \varphi(a^*).$$

For sequence x_n it is sufficient, in order to tally, that in the ambient area of root α the condition

$$|\varphi'(x)| < 1,$$

was fullfilled, it being known that the tallying is the quicker, the lesser $|\varphi'(x)|$ is.

Calculation must be stopped as soon as the difference

$$|x_n - x_{n-1}|$$

becomes less than the admissible absolute error. In other words, it is assumed that the condition

$$|x_n - x_{n-1}|$$

STAT

engenders the inequality

$$|x_n - a| < \epsilon.$$

In the simpliest case it is assumed that $\psi(x)=1$ and, consequently, solve the equation $x-f(x)=x$. The iterational methods are particularly convenient for application in programming, on account of similarity of performance of operations.

Method of false posture, or method of proportional parts.

Let only one root a of equation (9-1) be located at interval (a,b). Then let us take b for the first approximation of x_1, and determine the iterative process by the formulas

$$x_2 = \frac{af(x_1) - x_1 f(a)}{f(x_1) - f(a)};$$

$$x_3 = \frac{af(x_2) - x_2 f(a)}{f(x_2) - f(a)};$$

. . . . . .

Geometrically x_{i+1} is a point of intersection of the axis of abscisses with the chord connecting points $[a, f(a)]$ and $[x_i, f(x_i)]$ of curve $y=f(x)$. It is easy to establish, that in our case function ψ has the form $\psi(x) = \frac{x-a}{f'(x) - f(a)}$.

Newton's Method. If function

$$\psi(x) = \frac{1}{f'(x)},$$

is taken for ψ, then function φ will have the form

$$\varphi(x) = x - \frac{f(x)}{f'(x)}$$

and the iterational process will be expressed by the formula:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}.$$

Geometrically x_{i+1} is a point of intersection of the axis of abscisses with a tangent of curve $y=f(x)$ drawn at point $[x_i, f(x_i)]$. It will be noted, that Newton's method secures a quicker tallying than the other above mentioned methods. In paragraph 3-7 we applied this method for the calculation of the value of the square root of x.

Algebraic equations. Let function f(x) in equation (9-1) be the polynomial of power n with material coefficients, i.e. that (9-1) has the form:

$$P_n(x) \equiv a_0 x^n + a_1 x^{n-1} + \ldots +$$

$$+ a_{n-1} x + a_n = 0. \quad (9-3)$$

According to the principal algrebraic theorem this equation has exactly n roots, with due regard for their multiplicity. With this, root a is called the k-multiple root of equation (9-3), provided that the conditions

$$P(a) = P'(a) = \ldots = P^{(k-1)}(a) = 0;$$

$$P^{(k)}(a) \neq 0.$$

are met.

These roots can include complex roots and then the latter are always conjugated in pairs. In other words, numbers $a+i\beta$ and $a-i\beta$ are, at the same time, the roots of the equation (9-3).

An approximate definition of the roots of equation (9-3) can be accomplished with the use of the above-described methods, but inasmuch as they assume an approximately known arrangement of roots, we are going to start off with the consideration of this factor. The simplest way would be the consultation of a polynomial graph. Yet, this method is not always within reach and we have to make use of other methods.

Let us, first of all, introduce the following theorem. If the maximum of the moduluses of all coefficients of polynomial P (x) is designated A, then the number

$$1+\frac{A}{|a_0|}$$

constitutes the upper boundary for moduluses of all its roots, be they true or complex.

Before we begin the examination of arrangement of real zeroes, let us point out the following. Let:

$$P_1(x)=x^nP\left(\frac{1}{x}\right);$$

$$P_2(x)=P(-x);$$

$$P_3(x)=x^nP\left(-\frac{1}{x}\right)$$

and let N_1, N_2 and N_3 be respectively the upper boundaries of their positive roots. Then $1/N_1$ is the lower boundary of positive roots of polynomial P(x), whereas numbers N_2 and $-1/N_3$ respectively represent the upper and the lower boundaries, of its negative roots. Thus, we may discuss in advance only the upper boundary of the positive roots of polynomials. Two theorems can thus be formulated onthis subject:

Langrange's theorem: if $a_0>0$, $a_k(k\geq 1)$ is the first negative coefficient and B is the greatest absolute value of negative coefficients, then number

$$1+\sqrt[k]{B/a_0}$$

constitutes the upper boundary of positive roots of a multinomial.

Newton's theorem: if at x=c the polynomial P (x) and all its consecutive derivatives $P'(x),P''(x),\ldots, P^{(n)}(x)$ are positive, then the upper boundary of the positive roots is number c.

Inasmuch as $P^{(n)}(x) = n!a_0 > 0$, then $P^{(n-1)}(x)$ is a growing function and, consequently, there exists such a c_1, where at $P^{(n-1)}(x) \geqslant 0$ at $x \geqslant c_1$. Consequently, at $x \geqslant c_1$, derivative $P^{(n-2)}(x)$ is a growing function and, therefore, there exists such a c_2 whereat $P^{(n-2)}(x) \geqslant 0$ at $x \geqslant c_2$. Continuing this way, we can find the unknown c.

Sturm's method provides an exhaustive answer to the problem of determining the real roots, but in view of its cumbersome character it is hardly usable and will not be the subject of our examination. We will now turn to Descarte's theorem, according to which, the number of positive roots of multinomial P(x) (taking into account their multiplicity) is equal to the number of changes of signs in the system of this multinomial, or is less than that number by an even number. For example, polynomial

$$x^8+5x^7+3x^5-2x^4-7x^3+9x-10=0$$

has either three, or only one positive root.

Lobachevskiy's method. For the solution of algebraic equations

$$x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n=0 \quad (9\text{-}4)$$

exists the Lobachevskiy method, which does not call for preliminary separation of roots. Let us for the beginning presume, that all roots of equation (9-4) are material and different. Aligning them in a sequence of decrease of moduluses

$$|a_1| > |a_2| > \cdots > |a_n|.$$

we can, on the basis of equation (9-4), make up the equation

$$z^n+a_1'z^{n-1}+\ldots+a_{n-1}'z+a_n'=0 \quad (9\text{-}4')$$

in accordance with the following rule:

coefficient a'_{n-k} at z^k is equal to the square of the coefficient at x^k minus the double product of the coefficients

STAT

STAT

at x^{k+1} and x^{k-1}, plus double the product of the coefficients at x^{k+2}, and so on, until the first or the last member of the initial equation.

In other words

$$a_1' = a_1^2 - 2a_2;$$

$$a_2' = a_2^2 - 2a_1 a_3 + 2a_4;$$

. . . . . .

$$a_k' = a_k^2 - 2a_{k+1}a_{k-1} + 2a_{k+2}a_{k-2} \cdots;$$

Proceding from equation (9-4'), another equation can be constructed, just the same way as equation (9-4') was made from equation (9-4). Continuing this way, at (k-1)-th and at k-th steps we get equations:

$$x^n + a_1^{(k-1)} x^{n-1} +$$

$$+ a_2^{(k-1)} x^{n-2} + \ldots + a_n^{(k-1)} = 0; \quad (9\text{-}5)$$

$$x^n + a_1^{(k)} x^{n-1} + a_2^{(k)} x^{n-2} + \ldots + a_n^{(k)} = 0,$$

for which the approximation equations

$$a_1^{(k)} \approx [a_1^{(k-1)}]^2;$$

$$a_2^{k} \approx [a_2^{(k-1)}]^2;$$

. . . . . .

$$a_n^{(k)} \approx [a_n^{(k-1)}]^2; \quad (9\text{-}6)$$

will be met, i.e. in the k-th transformed equation the coefficients will be equal to the square of corresponding coefficients in the preceding equation (with the accepted degree of calculation accuracy). Under these circumstances it may be stated that:

STAT

$$a_1^h = a_1^{(k)};$$

$$a_2^h = \frac{a_2^{(k)}}{a_1^{(k)}}$$

$$a_3^h = \frac{a_3^{(k)}}{a_2^{(k)}};$$

. . . . . .

$$a_n^h = \frac{a_n^{(k)}}{a_{(n-1)}^{(k)}},$$

where $h=2^k$ (if, for example, k=5, then $h=2^5=32$). From these equations, it is possible to determine the moduluses of roots a_1, $a_2,\ldots,a_n$; the sign of theroots is determined by way of direct substitution in equation (9-4).

Presuming now, that among the roots of equation (9-4) are several pairs of complex-conjugated roots

$$a_i = r_i(\cos\varphi_i + \sqrt{-1}\sin\varphi_i);$$

$$a_{i+1} = r_i(\cos\varphi_i - \sqrt{-1}\sin\varphi_i),$$

$$a_j = r_j(\cos\varphi_j + \sqrt{-1}\sin\varphi_j);$$

$$a_{j+1} = r_j(\cos\varphi_j - \sqrt{-1}\sin\varphi_j)$$

and having the roots numbered anew, so that

$$|a_1| > |a_2| > \ldots > |a_{i-1}| > r_i >$$
$$> |a_{i+2}| > \ldots > |a_{j-1}| > r_j >$$
$$> |a_{j+2}| > \ldots > |a_n|.$$

we shall again get equations (9-5), for which the equalities (9-6) will be met for coefficients on all powers of x, with the exception of powers i, j,... Conversely, coefficients $a_i^{(k)}$, $a_j^{(k)}$ will behave in a most unprediotable manner. They are even capable of changing their signs, which is the best manifest indication of the presence of complex roots. Under these conditions, in place of equalities (9-7) there appear equalities

$$a_1^h = a_1^{(k)}; \qquad a_{j-1}^h = \frac{a_{j-1}^{(k)}}{a_{j-2}^{(k)}};$$

. . . . .

$$a_{i-1}^h = \frac{a_{i-1}^{(k)}}{a_{i-2}^{(k)}}; \qquad r_j^h = \frac{a_{j+1}^{(k)}}{a_{j-1}^{(k)}};$$

$$r_i^h = \frac{a_{i-1}^{(k)}}{a_{i-1}^{(k)}}; \qquad a_{j+2}^h = \frac{a_{j+2}^{(k)}}{a_{(k)}};$$

$$a_{i+2}^h = \frac{a_{i+2}^{(k)}}{a_{i+1}^{(k)}}; \qquad \ldots\ldots\ldots$$

$$a_n^h = \frac{a_n^{(k)}}{a_{n-1}^{(k)}}.$$

With the use of these equalities we at first determine the material roots $a_1, a_2, \ldots, a_{i-1}, a_{i+2}, \ldots, a_j, a_{j+2}, \ldots, a_n$ and moduluses $r_i, r_j, \ldots$ of complex roots. If the complex roots are represented by one pair then their argument can be found from

$$-a_1 = a_1 + a_2 + \ldots + a_{i-1} +$$
$$+ 2r_i \cos \varphi_i + a_{i+2} + \ldots + a_n.$$

STAT

If, however, the complex roots are represented by two pairs, their arguments φ_i and φ_j can be found from the system of the two equations:

$$-a_1 = a_1 + \ldots + a_{i-1} + 2r_1 \cos\varphi_i +$$

$$+ a_{i+2} + \ldots + a_{j+1} + 2r_j \cos\varphi_j +$$

$$+ a_{j+2} + \ldots + a_n;$$

$$-\frac{a_{n-1}}{a_n} = \frac{1}{a_1} + \ldots + \frac{1}{a_{i-1}} + \frac{2}{r_j}\cos\varphi_i +$$

$$+ \frac{1}{a_{i+2}} + \ldots + \frac{1}{a_{j-1}} + \frac{2}{r_j}\cos\varphi_j +$$

$$+ \frac{1}{a_{j+2}} + \ldots + \frac{1}{a_n}.$$

The determination of arguments of complex roots when there are more than two pairs of complex roots, and determination of multiple roots or roots relative to them by the modulus, are not included in this work. Pertinent details thereto can be found in the book by A. N. Krylov (L.15).

Systems of linear equations.

A formally exact solution of a system of linear equations can be made by Kramer's formulas. However, in such cases when the number of a system's equations is sufficiently large, the calculation of determinants in the Kramer formulas becomes practically impossible even for the high-speed contemporaneous computers. Much more practicable is the Gauss method by which system

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1;$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2;$$

$$\ldots\ldots\ldots\ldots\ldots$$

$$a_{n1}x_1 + a_{n2}x_n + \ldots + a_{nn}x_n = b_n;$$

is reduced to the triangular form of

$$a'_{11}x_1 + a'_{12}x_2 + \ldots + a'_{1n}x_n = b'_1;$$

$$a'_{22}x_2 + \ldots + a'_{2n}x_n = b'_2;$$

$$\ldots\ldots\ldots\ldots$$

$$a'_{nn}x_n = b'_n,$$

as described in paragraph 4-3 dealing with the calculation of a determinant. The solution of the triangular system is not difficult at all.

There also exist some very economical iterative methods of approximate solution of systems of equations, but we are not going to mention them.

9-3. Interpolation of Functions.

Problem of interpolation.

Let us assume that in the n+1-th point

$$x_0, x_1, \ldots, x_n$$

are the prescribed values of

$$y_0 = f(x_0),\ y_1 = f(x_1), \ldots, y_n = (x_n)$$

of some function $y = f(x)$. The polynomial $P_n(x)$ of power n assuming at points $x_0, x_1, \ldots, x_n$ the values $y_0, y_1, \ldots, y_n$:

$$P_n(x_i) = y_i,\ i = 0, 1, \ldots, n,$$

STAT

is called an interpolation polynomial connected with function f (x) and the aggregate of points x_i (the existance of such polynomial has been proven in advanced algebra). The construction of this polynomial and the calculation of its values at points x which do not coincide with points x_i, i=0,1,...,n, are called "interpolation", and points x_i are called "basic points of interpolation". At times, the calculation of values of $P_n(x)$ at points x located outside of the interval (x_0,x_n) (in this case it is assumed that $x_0 > x_1 > \ldots > x_n$) is also called "extrapolation", whereas the name "interpolation" is retained for values of x located within this interval.

The necessity to have such a polynomial constructed, arises in various cases. Let, for example, the values $y_0, y_1,\ldots,y_n$ be regarded as obtained by way of experimentation and, consequently, the expression of function f(x) is not known. Then we take for the value of function $\dot{x}(f)$ at points x differing from x_i (i=0,1,...,n) the value of interpolation polynomial $P_n(x)$. Sure enough, in this case function f(x) is subject to imposition of some limitations (it is then regarded as sufficiently smooth). At times, the expressions of function f (x) is known, but it is so complex, that the calculation of its value is quite a labor-consuming process. In the interest of simplicity and labor-saving, the calculation of values of function f (x) in such cases is performed at a series of points $x_0,x_1,\ldots,x_n$, followed by the construction of an interpolation polynomial on those points, by taking for the values of function f(x) at points x, different from x_i, the values of $P_n(x)$.

Case of equidistant basic points. Now we shall examine a most frequently encountered case when values of some function y=f(x)

$$y_1,y_2,y_3,\ldots,y_m \quad (9\text{-}8)$$

are given at points $x_1,x_2, x_3,\ldots,x_n$ (9-9)

equidistant from one another by equal distances h, i.e.

STAT

$$x_i=x_1+h(i-1),\ i=1,2,\ldots,m.$$

In this case, the construction of interpolation polynomials is made with the aid of the so-called differences, which have the following definitions. The name "difference of the first order" is applied to values

$$\Delta y_1=y_2-y_1,$$

$$\Delta y_2=y_3-y_2,\ldots,\Delta y_{m-1}=y_m-y_{m-1}.$$

In turn, the differences of the first differences are called "difference of the second order". They are, consequently equal:

$$\Delta^2 y_1=\Delta y_2-\Delta y_1=y_3-2y_2+y_1;$$

$$\Delta^2 y_2=\Delta y_3-\Delta y_2=y_4-2y_3+y_2;$$

. . . . . .

$$\Delta^2 y_{m-2}=\Delta y_{m-1}-\Delta y_{m-2}=$$

$$=y_m-{}^2y_{m-1}+y_{m-2}.$$

In a similar way are determined the differences of the third order, and so on. Generally speaking, it offers no difficulty toshow that a difference of the k-th order has the form of:

$$\Delta^k y_i=\sum_{m=0}^{k}(-1)^m C_k^m y_{i+k-m},$$

wherein C_k^m are binominal coefficients

$$C_k^m = \frac{k(k-1)\cdots(k-m-1)}{1.2\cdots m}.$$

For convenience, these differences are gathered into a table of differences, which represents the behavior of differences of various orders. For example, a table of differences for function $f(x)=x^5$ is represented herein by table 9 - 1.

Table 9-1

Table of Differences of Functions $y-x^5$

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$	$\Delta^6 y$
0	0	1	30	150	240	120	0
1	1	31	180	390	360	120	0
2	32	211	570	750	480	120	0
3	243	781	1320	1230	600	120	—
4	1024	2101	2550	1830	720	-	—
5	3125	4651	4380	2550	—	—	—
6	7776	9031	6930	—	—	—	—
7	16807	15961	—	—	—	—	—
8	32768	—	—	—	—	—	—

In this table the differences of the fifth order are constant, those of the sixth order are equal to zero. Let now x be an arbitrary point and let u be

$$u = \frac{x-x_i}{n},$$

where x_i is one of the points (9-9). The polynomial power of n, assuming at n+1st point $x_i, x_{i+1}, \ldots, x_{i+n}$, the values $y_i, y_{i+1}, \ldots, y_n$, is defined by the Newton's interpolation formula.

$$P_n(x) = y_i + \frac{u}{1!}\Delta y_i + \frac{u(u-1)}{2!}\Delta^2 y_i +$$

$$+\frac{u(u-1)(u-2)}{3!}\Delta^3 y_i + \ldots$$

$$\ldots + \frac{u(u-1)(u-2)\ldots(u-n+1)}{n!}\Delta^n y_i.$$

Error $R_n(x) = |f(x) - P_n(x)|$, admitted at replacement of function f (x) by interpolation polynomial Pn (x), is expressed by the formula

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{n}(x-x_i)\times$$

$$\times(x-x_{i+1})\ldots(x-x_{i+n}), \quad (9\text{-}10)$$

wherein ξ is a point located between the largest and the smallest of points x, x_i, $x_{i+1},\ldots,x_{i+n}$. The magnitude can be found if the expression of function f(x) is known and the n+1th derivative $f^{(n+1)}(x)$ can be easily appraised. Otherwise, it will be necessary to employ the approximate equality

$$f^{(n+1)}(\xi) = \frac{\Delta^{n+1} f(x_i)}{h^{n+1}},$$

from which such a power n of the interpolation polynomial is taken which can be instrumental in rendering the n-th differences within the chosen accuracy to be positive (then the n+1th differences are equal to zero and R_n is equal to zero, too).

Considering formula (9-10) we can come to a conclusion with respect to the choise of the most expedient selection of points x_i (contained in the table) at a prescribed value x. Indeed, the formula for R_n includes polynomial

$$(x-x_i)(x-x_{i+1})\ldots(x-x_{i+n}),$$

whose roots are points x_i, $x_{i+1},\ldots x_{i+n}$, (at $x=x_i$ $f(x_i)=P(x_i)$

STAT

and $R(x_i)=0$). Examining this polynomial's graph, we shall see, that its maximums are close to the average root $x_i + \frac{n}{2}$ (considering n as an even number), and less than the maximums close to the end roots x_i and x_{i+n} (Fig.9-1). Therefore, x_i must be chosen so as to have point x as close as possible to the middle of the basic points of interpolation $x_i, x_{i+1}, \ldots, x_{i+n}$. Having made this choise, we may be sure, that the interpolation values have the same number of true signs, as there are in the table values of $y_1, y_2, \ldots, y_m$

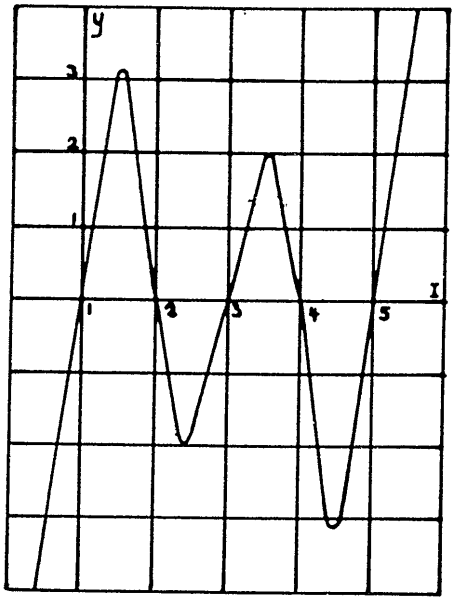


Fig. 9-1

Naturally, when x is located close to the table sides (to x_1 or x_m), then the above-mentioned choise of x_i is impossible and we have to take the best one from the possible points. Particularly, when $x_1 < x < x_2$, then it will be x_1 that provides an entry to the table; if however, $x_{m-1} < x < x_m$, then it will be x_{m-1}. Actually, performing interpolation, the entry of x_i in the table is chosen by the operator anew for every new value of x, whereas the power of n polynomial is taken always as positive.

Inequidistant basic points

In a case, when points (9-9) are disposed about arbitrarily, we may introduce the so called "divided differences", which can be determined in the following way: the divided differences of the first order are called the quantities

$$f(x_1, x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

$$f(x_2, x_1) = \frac{f(x_2) - f(x_1)}{x_2 - x_1}, \ldots$$

divided differences of the second order are called the quantities

$$f(x_2, x_1, x_0) = \frac{f(x_2, x_1) - f(x_1, x_0)}{x_2 - x_0},$$

$$f(x_3, x_2, x_1) = \frac{f(x_3, x_2) - f(x_2, x_1)}{x_3 - x_1}$$

and so on.

Divided differences find application in formulation of the Newton's formula for inequal intervals

$$P_n(x) = f(x_i) + (x - x_i) f(x_{i+1}, x_i) +$$

$$+ (x - x_i)(x - x_{i+1}) f(x_{i+2}, x_{i+1}, x_i) + \ldots +$$

$$\ldots + (x - x_i)(x - x_{i+1}) \ldots$$

$$\ldots (x - x_{i+n-1}) f(x_{i+n}, x_{i+n-1}, \ldots, x_i).$$

STAT

The Interpolation polynomial, that calls for no calculation of divided differences is defined by the Lagrange formula

$$P_n(x) = f(x_i)\cdot\frac{(x-x_{i+1})(x-x_{i+2})\dots(x-x_{i+n})}{(x_i-x_{i+1})(x_i-x_{i+2})\dots(x_i-x_{i+n})} +$$

$$+f(x_{i+1})\frac{(x-x_i)(x-x_{i+2})\dots}{(x_{i+1}-x_i)\,(x_{i+1}-x_{i+2})\dots}\rightarrow$$

$$\rightarrow\frac{\dots(x-x_{i+n})Z}{\dots(x_{i+1}-x_{i+n})}+\dots+f(x_{i+n})X$$

$$X\,\frac{(x-x_i)(x-x_{i+1})\dots(x-x_{i+n-1})}{(x_{i+n}-x_i)\,(x_{i+n}-x_i)\dots(x_{i+n}-x_{i+m-1})}$$

These are polynomials of the n-th power, which assume the prescribed values $f(x_i)$, $f(x_{i+1}),\dots,f(x_{i+n})$ at points $x_i, x_{i+1},\dots, x_{i+n}$. Consequently, both polynomials coincide, despite their difference in appearance resulting from the different way of grouping of their members. Let us note, that in this case the points are numbered quite arbitrarily; in more regular cases they are numbered in increasing order. The choice of power n and of entry of x_i into the table can be found by the formula (9-10). However, because points $x_1, x_2,\dots,x_m$ and asymmetric, the choise of entry x_i into the table is, in this case, much more complex.

Inverse interpolation. Let at points (9-9) be prescribed values (9-8) of some function $y=f(x)$. Finding out such argument x, at which function f(x) assumes a value y^*, that does not coincide with the table values (9-8), is called the problem of inverse interpolation.

If points (9-9) are disposed equidistantly, a problem of this type can be solved by Newton's formula. Assuming $P_n(x)=y^*$, STAT

STAT

we can express it as

$$u = \frac{y^* - y_i}{\Delta y_i} - \frac{u(u-1)}{2!}\frac{\Delta^2 y_i}{\Delta y_1} - \ldots -$$

$$- \frac{u(u-1)\ldots(u-n+1)}{n!}\frac{\Delta^n y_i}{\Delta y_i}.$$

In other words, Newton's formula assumes the form of equation

$$u = F(u),$$

which can be solved by way of interpolation, by taking as first approximation the quantity

$$u^{(1)} = \frac{y^* - y_i}{\Delta y_i}.$$

Having determined u, we can, on the grounds of equality

$$u = \frac{x - x_i}{h},$$

find the unknown x.

In case points (9-9) are disposed arbitrarily, the same problem is solved by the interpolation of the Lagrange formula which however, is expressed in such a way, that x is considered a dependent, and and y an independent variable:

$$x = x_i \frac{(y-y_{i+1})(y-y_{i+2})\ldots(y-y_{i+n})}{(y_i-y_{i+1})(y_i-y_{i+2})\ldots(y_i-y_{i+n})} +$$

$$+ x_{i+1}\frac{(y-y_i)(y-y_{i+2})\ldots(y-y_{i+n})}{(y_{i+1}-y_i)(y_{i+1}-y_{i+2})\ldots(y_{i+1}-y_{i+n})} +$$

$$+ \ldots + x_{i+n}\times$$

$$\frac{(y-y_i)(y-y_{i+1})\ldots(y-y_{i+n-1})}{(y_{i+n}-y_i)(y_{i+n}-y_{i+1})\ldots(y_{i+n}-y_{i+n-1})}.$$

assuming here that $y = y^*$, we can find out the unknown value of x.

STAT

9-4. Numerical Differentiation and Integration of Functions

Numerical differentiation. Differentiation of functions must be made then, when the values of functions are prescribed by a table, or when representation of derivatives by an analytical expression is too complex. In either case, for finding out the derivative functionf(x) at some point x, an interpolation polynomial must be made up, as follows:

$$P_n(x) = y_i + u\frac{\Delta y_i}{1!} + u(u-1)\frac{\Delta^2 y_i}{2!} +$$

$$+ u(u-1)(u-2)\frac{\Delta^3 y_i}{3!} + \ldots$$

$$\ldots + u(u-1)(u-2)\ldots(u-n+1)\frac{\Delta^n y_i}{n!};$$

where

$$u = \frac{x - x_i}{h}$$

and where values of derivatives of polynomial $P_n(x)$ are representing values of derivatives of $f(x)$.

Designating

$$V_m = u(u-1)(u-2)\ldots(u-m+1),$$

whence it follows

$$\frac{dV_m}{du} = 1!\left(\frac{1}{u} + \frac{1}{u-1} + \ldots + \frac{1}{u-m+1}\right)V_m;$$

$$\frac{d^2V_m}{du^2} = 2!\left(\frac{1}{u(u-1)} + \frac{1}{u(u-2)} + \ldots + \frac{1}{(u-m+2)(u-m+1)}\right)V_m;$$

$$\frac{d^3V_m}{du^3} = 3!\left(\frac{1}{u(u-1)(u-2)} + \frac{1}{u(u-2)(u-3)} + \ldots + \frac{1}{(u-m+3)(u-m+2)(u-m+1)}\right)V_m;$$

. . . . . .

$$\frac{d^mV_m}{du^m} = m!; \quad \frac{d^{m+1}V_m}{du^{m+1}} = 0.$$

STAT

With the use of these formulas and taking into consideration that

$$\frac{d^k P_n}{dx^k} = \frac{1}{h^k}\frac{d^k P_n}{du^k},$$

we can express the derivatives $P_n(x)$ in a form convenient for calculations

$$\frac{hP'_n}{1!} = \Delta y_i + \left[\frac{1}{u} + \frac{1}{u-1}\right]u(u-1)\frac{\Delta^2 y_i}{2!} + \left[\frac{1}{u} + \frac{1}{u-1} + \frac{1}{u-2}\right]u(u-1)(u-2)\frac{\Delta^3 y_i}{3!} +$$

$$+\ldots+\left[\frac{1}{u} + \frac{1}{u-1} + \ldots + \frac{1}{u-n+1}\right]u(u-1)(u-2)\ldots(u-n+1)\frac{\Delta^n y_i}{n!} +$$

$$\frac{h^2 P''_n}{2!} = \Delta^2 y_i + \left[\frac{1}{u(u-1)} + \frac{1}{u(u-2)} + \frac{1}{(u-1)(u-2)}\right]u(u-1)(u-2)\frac{\Delta^3 y_i}{3!} + \ldots$$

$$\ldots+\left[\frac{1}{u(u-1)} + \frac{u}{u(u-2)} + \ldots + \frac{1}{(u-n+2)(u-n+1)}\right]u(u-1)\ldots(u-n+1)\frac{\Delta^n y_i}{n!};$$

$$\frac{h^3 P'''_n}{3!} = \Delta^3 y_i + \ldots + \left[\frac{1}{u(u-1)(u-2)} + \frac{1}{u(u-2)(u-3)}\ldots\right.$$

$$\left.\ldots + \frac{1}{(u-n+3)(u-n+2)(u-n+1)}\right]u(u-1)\ldots(u-n+1)\frac{\Delta^n y_i}{n!}.$$

If we have to find out derivatives $f(x)$ in one basic point of interpolation x_{i+p} (i.e. at $u=p$), we have to exclude from every square bracket such items as

$$\frac{1}{(n-k)(u-1)\ldots(u-m)},$$

which do not contain the multiplier u-p and then replace in the thus obtained formulas u-p by a unity.

In this case it would be convenient to have beforehand calculated the magnitudes of the coefficients at $\Delta^k y_i$ (in the case under consideration $u=p$).

STAT

Numerical integration. Let function $f(x)$ be given at interval (a,b) and let us calculate integral $I=$

$$I=\int_a^b f(x)dx.$$

The process of approximate finding out quantity I is called "mechanical quadrature". It is made use of every time an integral is not taken in final form. A general scheme of approximate calculation of the integral consists of the following: function $f(x)$ is replaced by interpolation polynomial $P_n(x)$ and integral I_n is taken to represent the integral quantity

$$I_n=\int_a^b P_n(x)dx.$$

Let us examine a case when basic points of interpolation divide interval (a,b) by n equal parts and the lengths

$$h=\frac{b-a}{n};$$

$$x_o=a_1,\ x_1=a+h,$$

$$x_2=x_o+2h,\ldots,\ x_n=a+nh=b.$$

Assuming $A_n(x)=(x-x_o)(x-x_1)(x-x_2)\ldots(x-x_n)$,

we can represent the Lagrange interpolation formula in the following compact form:

$$P_n(x)=\sum_{i=o}^{n} f(x_i)\frac{A_n(x)}{(x-x_i)A'(x_i)}.$$

Integrating this expression we obtain

$$I_n = \sum_{i=0}^{n} f(x_i) \int_a^b \frac{A_n(x)}{(x-x_i)A'(x_i)}\,dx.$$

Behind the integral sign in this formula does not depend upon function f(x), but depends only upon the power of interpolation of the polynomial and the interval (a,b). Consequently, these integrals can be calculated for n=1,2,3,... and, for instance, for interval (0.1). Thereupon we express the preceding formula in theform of

$$I_n = (b-a)(C_o^{(n)} f(a) + C_1^{(n)} f(a+h) +$$

$$+ C_2^{(n)} f(a+2h) + \ldots + C_n^{(n)} f(b)), \qquad (9\text{-}11)$$

which is correct for any interval (a,b), where

$$C_i^{(n)} = \int_0^1 \frac{A_n(x)}{(x-x_i)A'_n(x_i)}\,dx.$$

Quantities $C_i^{(n)}$ are called 'Cotes' coefficients. Their numerical values for n=1,2,...,6 are given in table 9-2.

STAT

Cotes' Coefficients

Table 9-2.

n	$C_i^{(n)}$	Interpolation step $h=\frac{b-a}{n}$	Basic point of Interpolation
1	$\frac{1}{2}, \frac{1}{2}$	$b-a$	a, b
2	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$	$\frac{b-a}{2}$	a, a+h, b
3	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	$\frac{b-a}{3}$	a, a+h, a+2h, b
4	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	$\frac{b-a}{4}$	a, a+h, a+2h, a+3h, b
5	$\frac{19}{288}, \frac{75}{288}, \frac{50}{288}, \frac{50}{288}, \frac{75}{288}, \frac{19}{288}$	$\frac{b-a}{5}$	a, a+h, a+2h, a+3h, a+4h, b
6	$\frac{41}{840}, \frac{216}{840}, \frac{27}{840}, \frac{272}{840}, \frac{27}{840}, \frac{216}{840}, \frac{41}{840}$	$\frac{b-a}{6}$	a, a+h, a+2h, a+3h, a+4h, a+5h, b

If, for instance, the interpolation is made by four points ($n=3$), then

$$I_3=\frac{(b-a)}{8}\{f(a)+3f(a+h)+3f(a+2h)+f(b)\}.$$

In order to estimate error D_n, ommitted at numerical integration, we can examine formula (9-10), whence it follows that

$$|D_n|<\frac{M_{n+1}}{(n+1)!}\int_a^b|A_n(x)|\,dx,$$

where M_{n+1} is the maximum of modulus n+1 of the derivative $f^{(n+1)}(x)$ at interval (a,b). Since every multiplier contained in $A_n(x)$, by its modulus is not greater than the interval length $L=b-a$, then

$$|A_n(x)|<L^{n+1},$$

STAT

whence it follows that

$$|D_n| < \frac{M_{n+1}}{(n+1)!} L^{n+2}. \qquad (9\text{-}12)$$

Inasmuch as $\frac{L^{n+2}}{(n+1)!} \to 0$, at $n \to \infty$, then by increasing the power of n of the interpolation polynomials it is possible to make any error (presuming, as a matter of fact, that M_{n+1} do not grow very fast). But such an increase of n entails very cumbersome calculations. Therefore, in practice this is not done. Another way is this: Let us divide interval (a,b) by N equal lengths of intervals $l = \frac{L}{N}$. Let the points of division be:

$$a_0 = a, a_1, a_2, \ldots, a_N = b.$$

Formula (9-11) is applicable not to the whole intervals (a,b), but to each of the intervals (a_k, a_{k+1}) separately, (at which an own interpolation polynomial is constructed for every interval a_k, a_{k+1}). Then the integral I_n will be equal to the sum of partial integrals. Applying the formula (9-12) to every integral separately, we discover that the admissible error will be less than

$$N \frac{M_{n+1}}{(n,1)!} l^{n+2} = l^{n+1} \frac{M_{n+1}}{(n+1)!}.$$

Shortening the length of the partial interval, but having fixed the power of interpolation polynomial n, it is possible to attain any desired accuracy.

Formulas for mechanical quadratures for partial intervals can be summarized. In particular, at n=1, the so-called trapezoid formula is formed

STAT

$$I_1=\frac{b-a}{2N}\{f(a_0)+2f(a_1)+$$

$$+2f(a_2)+\ldots+2f(a_{N-1})+f(a_N)\},$$

whose error is less than

$$l2\frac{M_2(b-a)}{2}.$$

At n=2, Simpson's formula is formed

$$I_2=\frac{b-a}{6N}\left\{f(a_0)+4f\left(\frac{a_0+a_1}{2}\right)+\right.$$

$$+2f(a_1)+4f\left(\frac{a_1+a_2}{2}\right)+2f(a_2)+$$

$$\left.+\ldots+2f(a_{N-1})+4f\left(\frac{a_{N-1}+a_N}{2}\right)+f(a_N)\right\},$$

whose error is less than

$$l^3\frac{M_3(b-a)}{6}.$$

The geometrical meaning of these formulas is extremely simple: at every partial interval the curve $y=f(x)$ is, in one case, replaced by the chord running through the extreme points (that is, the interval is equal to area of the thus obtained trapezoid), in another case by the parabola that runs through the extreme points and thru the middle point.

9-5. Solution of Usual Differential Equations

Setting the problem. Let a solution of differential equation

$$y'=f(x,y), \qquad (9\text{-}13)$$

be sought, that would satisfy the conditions

$$x_0\leq x\leq X,\ y(x_0)=y_0. \qquad (9\text{-}14)$$

STAT

Assuming that a solution to this problem does exist and that it is the sole solution. If we can not express it in the form of formulas, or if formulas are impractical for calculation work, we shall have to resort to approximate solution (9-13); (9-14): viz; to find at some points $x_0 \leq x_1 < x_2 < \ldots < x_n = X$, the values of $y_1, y_2, \ldots, y_n$, which are approximately equal to the values of $y(x_1), y(x_2), \ldots, y(x_n)$ of the accurate solution $y(x)$, [y_i denotes an approximate value of function $y(x)$ at point x_i.]

Taylor's expansion series. Let us assume, that the solution $y(x)$ of equation (9-13) is capable of being expanded, in the vicinity of every point x_i, in Taylor's series

$$y(x) = y(x_i) + \frac{(x-x_i)}{1!} y'(x_i) +$$

$$+ \frac{(x-x_i)^2}{2!} y''(x_i) + \ldots \quad (9\text{-}15)$$

Differentiating the equation (9-13) a sufficient number of times (taking y for function x), we can find any derivative $(y^{(n)}(x_i)$, and, consequently can calculate the series (9-15) with any degree of accuracy.

Thus,

$$y'(x_i) = [f(x,y)]_{\substack{x=x_i \\ y=y(x_i)}} ;$$

$$y''(x_i) = [f'_x(x,y) + y' f'_y(x,y)]_{\substack{x=x_i \\ y=y(x_i)}} =$$

$$= [f'_x + f f'_y]_{\substack{x=x_i \\ y=y(x_i)}} ;$$

$$y'''(x_i) = [f''_{xx} + y' f''_{xy} + (f'_x + y' f'_y) f'_y +$$

$$+f\left(f''_{xy}+f''_{yy}y'\right)\Big]_{\substack{x=x_i\\y=y(x_i)}} =$$

$$-\left[f''_{xx}+2ff''_{xy}+f''_{x}f'_{y}+\right.$$

$$\left.+ff'^{2}_{y}+f^{3}f''_{yy}\right]_{\substack{x=x_i\\y=y(x_i)}}. \quad (9\text{-}16)$$

Inserting the obtained values $y^{(n)}(x_i)$ in the formula (9-15), and presuming that $x=x_{i+1}$, $x_{i+1}-x_i=h$, we finally get:

$$y(x_{i+1})=y(x_i)+\frac{h}{1!}f(x_i, y(x_i))+$$

$$+\frac{h^2}{2!}\left[f'_x(x_i y(x_i))+\ldots+\right.$$

$$\left.+f(x_i y(x_i))f'_y(x_i y(x_i))\right]+$$

$$+\frac{h^3}{3!}\left[f''_{xx}(x_i y(x_i))+\ldots\right]+\ldots$$

Retaining in this formula a sufficient number of terms and presuming that $i=0.1,\ldots,n$, we can, in consecutive order, find out the approximate values of $y_1, y_2, \ldots y_n$ with any degree of accuracy.

Let us, for example, take the equation

$$\frac{dy}{dx}=-ytg\,x=f(x,y),\ y(0)=1,$$

whose solution, as it is easy to prove, is the function $y=\cos x$. Differentiating this equation, we find:

STAT

STAT

$$f'_x = \frac{-y}{\cos^2 x}; \quad f'_y = -\operatorname{tg} x;$$

$$f''_{xx} = -\frac{2y \sin x}{\cos^3 x};$$

$$f''_{xy} = -\frac{1}{\cos^3 x}; \quad f''_{yy} = 0. \quad (9\text{-}17)$$

Thus, $y'(0)=0$; $y''(0)=-1$; $y'''(0)=0$, and, consequently, value $y(x)$ at point $x_1=h(x_0=0)$ is equal to:

$$y(x_1) = 1 - \frac{h^2}{2!} + y^{(IV)}(x_1)\frac{h^4}{4!} + \ldots$$

When h is sufficiently small, the remainder $y^{(IV)}(x_1)\frac{h^4}{4!} + \ldots$ is small too, and an approximate value y_1, of quantity $y(x_1)$ is equal to:

$$y_1 = 1 - \frac{h^2}{2!} \approx \cos h.$$

Substituting x_1 and y in equation (9-17), we can find an approximate value of y_2 at some point x_2, and so forth. Also, the above-named method is reduced to calculation of a segment of Taylor(s series (9-15) with the use of equalities (9-16). If the function f(x,y) is sufficiently complex in itself, the calculation work becomes quite labor-consuming. The value of a segment of Taylor's series can be determined with sufficient accuracy by other three methods, which do not call for application of the formulas (9-16). Those methods are:

Euler's method. Contending ourselves with only two members of Taylor's series, we have the Euler formula

$$y_{i+1} = y_i + hf(x_i y_i) \quad (9\text{-}18)$$

$i=0,1,\ldots, n-1.$

STAT

Assuming in this formula that i=0, we obtain the value y_1 (y_0 is contained in the text of the problem); assuming i=1 and using the heretofore found value y_1, we find y_2, and so on.

The improved Euler-Cauchy method. Using the Cauchy formula, we can find the values of the first three terms of the (9-15) series with an accuracy down to the smallest point of a higher order. The formula is

$$y_{i+1} = y_i + \frac{1}{2}(k_1 + k_2), \quad (9\text{-}19)$$

wherein

$$k_1 = hf(x_i, y_i), \quad k_2 = hf(x_{i+1}, y_i + k_1);$$

$$i = 0, 1, \ldots, n-1.$$

The Runge-Kutta method. At last, the value of the first five terms of the (9-15) series with an accuracy down to the smallest point of a higher order, can be found by the formula

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad (9\text{-}20)$$

where

$$k_1 = hf(x_i, y_i),$$

$$k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right),$$

$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right),$$

$$k_4 = hf(x_i + h, y_i + k_3);$$

$$i = 0, 1, \ldots, n-1.$$

Let us note, that for definition of the first five terms of series (9-15) by the (9-16) formula, it is necessary to have calculated the values of the function f and of all its compound derivatives, up to and including the third order, the total number of which (together with the function f) is 10. Meanwhile, using the Runge-Kutta formula, we have to perform the

STAT

calculation of value of the function f at four points only. Besides, the (9-16) formulas call for the calculation of the products of these derivatives, which is also quite a time-consuming operation.

Order of Accuracy of approximate formulas. If with the aid of some approximate methods we calculate a segment, containing an accuracy down to the smallest points of the higher order n + 1 term of series (9-15), it is said that that method has an accuracy of n-th order. This is so, because of therewith are calculated such terms, which contain h up to the n-th power inclusive. The Runge-Kutta method for example, is a method of the fourth order.

It can be shown, that the difference $y(x_i)-y_i$ is drawn towards zero just the same way as are drawn towards zero the h, h^2,h^3 in the respective Euler, Euler-Cauchy and Runge-Kutta methods. This assertion becomes selfevident, when in the process of calculation of y_i, instead of using the approximate value y_{i-1}, we use the exact value $y(x_{i-1})$. However, the error at the i-th step is made up not only of the error that arises from discarding the remainder of the Taylor's series, but also of an error that accumulates from step to step, incident to replacement of $y(x_k)$ by y_k, where k=1,2,...,i-1.

It is evident, that from the three above-named methods, the Runge-Kutta method is the most accurate. In turn, the Euler-Cauchy method produces an accuracy higher than that obtained by the Euler method.

The Runge-Kutta method can be easily applied to systems of differential equations. Appropriate formulas have been introduced in paragraph 7-2.

Programs of integration of differential equations by the Runge-Kutta method have been described in Par. 4-2, 7-2 and 7-3.

Determining the accuracy of approximate solution.

The quantity

$$S = \left|\frac{\partial f}{\partial f}\right| . h,$$

called "characteristic step", is applied for the selection of the magnitude of step h in relatively simple cases. When the quantity

$$\left|\frac{\partial f}{\partial y}\right|$$

undergoes a change, step h must be changed too, namely, in such a way as to have all the time the equality

$$S = \left|\frac{\partial f}{\partial y}\right| . h = \text{const.} \qquad (9\text{-}21)$$

unchanged.

It should be noted, that the right parts of the formulas (9-18) and (9-20) in a preceding point contain exclusively the values x_1 and y_1 and, therefore, it is possible at every stage to change the step arbitrarily. In order to secure a moderate accuracy, the choice of step must be made in such a way, as to have the constant in (9-21) equal to 0.1 - 0.05.

The Runge-Kutta method for evaluation of the step value, also considers the relation

$$\left|\frac{k_2 - k_3}{k_1 - k_2},\right|$$

which may not exceed a few hundredths.

The accuracy of a solution of a numerical integration of differential equations can be judged by comparing at every step, the results obtained by a full step and by a half step. At that, the step value must be changed so, as to have at every step the difference between the solutions arrived at by full step and by half step, within the prescribed limits (see par. 7-3).

The Adams extrapolation method. Another method of approximate solution of equation

$$\frac{dy}{dx} = f(x,y),\ y(x_0) = y_0,\ x_0 \leq x \leq X \qquad (9\text{-}13)$$

STAT

is based on the following:

As above, let's decompose the interval (x_0, X) by points $x_0 < x_1 < x_2 < \ldots < x_n = x_n = X$ and integrate the equation (9-13) within the limits from x_i to x_{i+1}. The result is:

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y(x))dx. \qquad (9\text{-}22)$$

Now, let us assume, that we have somehow succeeded in the calculation of the approximate value $R_{i,i+1}$ of the integral, standing in the right part of this equality. Having introduced the quantity $R_{i,i+1}$ into the expression (9-22), we shall obtain the formula of approximate integration of a differential equation (9-13), namely

$$y_{i+1} = y_i + R_{i,i+1}, \; i = 0, 1 \ldots, n-1$$

Quantity $R_{i,i+1}$ can be determined by various ways, depending on what the various approximate formulas are obtained from. Having, for example, replaced the value of the sub-integral function by its value at the left point x_i, we again get the Euler formula

$$y_{i+1} = y_i + (x_{i+1} - x_i) f(x_i, y_i).$$

Out of a great number of similar formulas, we shall now examine only the Adams extrapolation formula . Let us assume, that the approximate values $y_0, y_1, \ldots y_i$ are already determined. Now, we replace the sub-integral function in the expression (9-22) by Newton's (interpolation polynomial) $P_{i-r,i}(x)$, which at points $x_{i-r}, x_{i-r+1}, \ldots, x_{i-2}, x_{i-1}, x_i$, assumes the values $f(x_{i-r}, y_{i-r}), \ldots, f(x_{i-1}, y_{i-1}), f(x_i, y_i)$. In the interest of simplicity, let us express this polynomial in the form of:

STAT

$$P_{i-r,i} = f_i + \frac{u}{1!}\nabla f_i + \frac{u(u+1)}{2!}\nabla^2 f_i +$$

$$+ \frac{u(u+1)(u+2)}{3!}\nabla^3 f_i +$$

$$+ \ldots + \frac{u(u+1)\ldots(u+r-1)}{r!}\nabla^r f_i,$$

$$u = \frac{x - x_i}{h},$$

$$h = x_{k+1} - x_k,\ f_i = f(x_i, y_i)$$

Here, through ∇^k are denoted inverse differences, which can be determined like common differences (see par.9-3), but make use only of the values of the function going to the left of entry x_i. Thus,

$$\nabla f_i = f_i - f_{i-1};$$

$$\nabla^2 f_i = \nabla(\nabla f_i) = f_i - 2f_{i-1} + f_{i-2};$$

By having this polynomial integrated within the limits from x_i to x_{i+1}, we obtain the Adams formula,

$$y_{i+1} = y_i + h\left(f_i + \frac{1}{2}\nabla f_i + \frac{5}{12}\nabla^2 f_i + \right.$$

$$+ \frac{3}{8}\nabla^3 f_i + \frac{251}{720}\nabla^4 f_i + \frac{95}{288}\nabla^5 f_i +$$

$$\left. + \frac{19\,087}{60\,480}\nabla^6 f_i + \ldots\right). \quad (9\text{-}23)$$

The number of members to be retained in this formula depends upon the desired degree of accuracy.

In order to apply the Adams formula, we have to determine the first "r"s of values $y_1, y_2, \ldots, y_r$. This is usually accomplished through the use of one of the above-examined methods. These values have to be determined exceedingly accurately, in order to reduce the influence of the errors in the initial values $y_1, \ldots y_r$ upon the whole course of the solution of the problem.

The magnitude of step h is determined on the basis of the step characteristic mentioned before. It should be pointed out, however, that in this case, the change of step can not be made as easily as it was in the case of the foregoing formulas. Indeed, formula (9-23) contains the function values at the preceeding, equidistant points. Consequently, at some stage of the calculation, the execution of step reduction will necessitate an additional determination of the values of the derivatives at the intermediate points, which can be produced through the use of the interpolation formulas examined above.

****

STAT

PR 1-1-57 Supplement No. 1

LIST OF OPERATIONS OF THE COMPUTER M-3

Operational Code	Agreed Designation of operation	Description of Operation
00	+	To the content of the first address is added the content of the second address and the result is recorded in the memory device after the second address
10	+,	To the content of the first address is added the content of the second address, but without recording the result in the memory device.
20	↓ +	The content of the first address is added to the content of the preceding operation and the result is recorded in the memory device after the second address.
30	↓ +,	The content of the first address is added to the content of the preceding operation, but without recording the result in the memory device.
40	+ Π	To the content of the first address is added the content of the second address. The result is in the memory device after the second address and is printed recorded
50	|+|,	To the modulus of the content of the second address is added the modulus of the content of the first address, but the result is not recorded in the memory device.
60	↓ + Π	To the content of the preceding operation is added the content of the first address. The result is recorded in the memory device and is printed.
70	↓ |+|,	The modulus of the content of the first address is added to the modulus of the result of the preceding operation without recording in the memory device.
01	—	The content of the first address is subtracted from the content of the second address and the result is recorded in the memory device by the second address.
11	—,	The content of the first address is subtracted from the content of the second address without recording the result in the memory device.
21	↓ —	The content of the first address is subtracted from the result of preceding operation and the result is recorded in the memory device by the second address.
31	↓ —,	The content of the first address is subtracted from the result of the preceding operation without recording the result in the memory device.

STAT

41	—Π	The content of the first address is subtracted from the content of the second address. The result is recorded by the second address and is printed.
51	|—|,	The modulus of the content of the first address is subtracted from the modulus of the content of the second address without recording the result in the memory device.
61	↓—Π	The content of the first address is subtracted from the result of the preceding operation. The result is recorded by the second address and gets printed.
71	↓|—|,	The modulus of the content of the first address is subtracted from the modulus of the result of preceding operation, without recording of the result in the memory device.
02	:	The content of the second address is divided by the content of the first address. The result is recorded in the memory device by the second address.
12	:,	The content of the second address is divided by the content of the first address, without recording the result in the memory device.
22	↓:	The content of the preceding operation is divided by the content of the first address and the result is recorded in the memory device by the second address.
32	↓:,	The result of preceding operation is divided by the content of the first address, without recording the result in the memory device.
42	:Π	The content of the second address is divided by the content of the first address. The result is recorded in the memory device by the second address and gets printed.
52	|:|,	The modulus of the content of the second address is divided by modulus of the content of the first address, without recording the result in the memory device.
62	↓:Π	The result of preceding operation is divided by the content of the first address. The result is recorded in the memory device by the second address and gets printed.

72	↓|:|,	The modulus of the result of preceding operation is divided by modulus of the content of the first address, without recording in the memory device.
03	×	The content of the second address is multiplied by the content of the first address. The result is recorded in the memory device by the second address.
13	×,	The content of the second address is multiplied by the content of the first address, without recording the result in memory device.
23	↓×	The result of preceding operation is multiplied by the content of the first address and the result is recorded in the memory device by the second address.
33	↓×,	The result of preceding operation is multiplied by the content of the first address, without recording the result in the memory device.
43	×Π	The content of the second address is multiplied by the content of the first address. The result is recorded by the second address and gets printed.
53	|×|,	The modulus of the content of the second address is multiplied by modulus of the content of the first address, without recording the result in the memory device.
63	↓×Π	The result of preceding operation is multiplied by the content of the first address. The result is recorded in the memory device by the second address and gets printed.
73	↓|×|,	The modulus of the result of preceding operation is multiplied by modulus of the result of the first address, without recording the result in the memory device.
06	∧	The content of the second address is logically multiplied by the content of the first address. The result is recorded in the memory device by the second address.
16	∧,	The content of the second address is logically multiplied by the content of the first address, without recording the result in the memory device.
26	↓∧	The result of preceding operation is logically multiplied by the content of the first address. The result is recorded in the memory device by the second address.

STAT

36	↓Λ,	The result of preceding operation is logically multiplied by the content of the first address, without recording the result in the memory device.
46	Λ П	The content of the second address is logically multiplied by the content of the first address. The result is recorded in the memory device by the second address and gets printed.
56	|Λ| ,	The modulus of the content of the second address is logically multiplied by the modulus of the content of the first address, without recording the result in the memory device.
66	↓Λ П	The result of preceding operation is logically multiplied by the content of the first address. The result is recorded in the memory device by the second address and gets printed.
76	↓|Λ| ,	The modulus of the result of preceding operation is logically multiplied by the content of the first address, without recording the result in the memory device.
07; 27	Вв	Inlet of numbers. From the perforated tape the number is recorded by the second address, without retention on the register
05; 15	П Ч	The content of the first address is transferred by the second address and is retained in the arithmetical unit on the register of the second number.
45, 55	П Ч П	The content of the first address is transferred by the second address and gets printed, but is not retained in the arithmetical unit.
24	П У	Transfer of the control. The next command is taken by the first address. The result of preceding operation is recorded by the second address and is retained in the arithmetical unit.
64	П У П	Same as operation 24, supplemented by printing of the number.
74	П У 2	The next command is taken by the second address. The modulus of the result of preceding operation is retained in the arithmetical unit.

STAT

STAT

54	У П	Conditional transfer. The next command is taken by the second address when the result of preceding operation had a "+", and by the first address if it had a "—".
04, 14, 44, 54, 17, 37, 67, 77	Stop	"Stop" operations differ from one another by the content of registers of the arithmetical unit and by the contents of the selecting and the starting registers.

STAT

Supplement No. 2

LIST OF OPERATIONS OF THE COMPUTER "URAL"

Operational Code	Agreed Designation of operation	Condition of generation of signal $\omega = 1$	Description of operation	Remarks
01	C∧ 1a	$z < 0$	Algebraic addition of the number in the summator with the number in cell a.	—
02	C∧ 2a	$z < 0$	Quenching the summator and transfer to it of the content of cell a.	—
03	B∀ 1a	$z < 0$	Algebraic subtraction of the number in cell a from the number in the summator.	
04	B∀ 2a	$z < 0$	Algebraic subtraction of the modulus of the number in cell a from modulus of the number in the register.	
05	YM 1a	$z < 0$	Multiplication of the number in the register by the number in cell a, and the addition of the result to the number in the summator.	This operation permits the calculation of the sum of the products of type $a_1b_1 + a_2b_2 + \ldots$ without the transfer of intermediary sums to the memory device on the magnetic drum.
06	YM 2a	$z < 0$	Multiplication of the number in the summator by the number in cell a.	This operation permits the calculation of products of the a_1, a_2, $a_3 \ldots$ type, without transfer of the intermediary result to the memory device on the magnetic drum.
07	D∧a	$z < 0$	Division of the number in the summator by the number in cell a.	—
10	Φ 3a	$z = 0$	Assigning to the number in the summator the sign of the number in cell a.	—
11	C ∍	$z = 0$	Shift of the number in the register by a number of columns corresponding to the number in the summator located in columns 19 to 35	Upon execution of the C∍ operation, the overfil is blocked. If the number in the summator is positive, then the shift is made to the left, if negative — to the right. —
12	B∍ a	$z = 0$	Selection of a part of the number in the summator by the number in cell a, in accordance with the rule of logical multiplication by groups of columns	

STAT

UNCLASSIFIED

12	B ↄ a	z = 0	Selection of a part of the number in the summator by the number in cell a, in accordance with the rule of logical multiplication by groups of columns.	—
13	Φ pa	z = 0	Serial addition of the number in the summator with the number in cell a, by groups of columns	In every column the addition is made by the rules 0+0=0, 0+1=1+0=1, 1+1=1, without formation of transfer of unities.
14	Cpa	x ≠ y	Comparison of the content of the summator with the number in cell a.	In an incongruency of coincidence of the set of colums in the summator and in cell a, unities are recorded in respective columns of the result.
15	Hpa	z=0	Normalization of number in the summator with the record of the result in cell a.	A number corresponding to the number of shifts is fixed in the summator.
16	Пσa	z < 0	Transfer of number from the summator to cell a.	The number is retained in the summator
17	П pa	If the number fed into the register is = 0	Clearing of the register and transfer to it of the number from a.	—
20	ПCK	—	Clearing the summator and recording therein the number k located in the address portion of the command	—
26	Cma	x < 0	Algebraic addition of the number in the summator with the number in cell a.	Same as C∧la, but the arrester is blocked at overfill.
21	Ela	—	Conditional transfer of control (the first operation of the control transfer)	Depending upon the signal, worked out at the preceding cycle, the control is transferred: at ω = 0 - to the next command; at ω = 1 to the command in cell a.
22	E2a	—	Unconditional transfer of control (the second operation of the control transfer)	The control is transferred to command located in cell a, irrespective of value worked out the preceding cycle.
23	B3K	—	The third operation of transfer of control	At the command B3 k, depending on the position of the key K on the control panel, one kommand following the command B3 k is either executed, or left out. The numbr of the key K changes from 1 to 7.
		—		

STAT

24	Е4а	—	The fourth operation of transfer of control, securing repetition of the cycle the needed number of times	It is put at the end of the cycle. When the cycle has not yet repeated itself the needed number of times, the control is transferred to cell a. Otherwise it goes over to the next command.
25	Мцn	—	Preparatory operation for iteration of cycle the needed number of times	It is put ahead of the cycle. At every repetition of the cycle, in the negative commands the addresses of not full cells are increased by one (12th column of command —0), or by 2 (12th column of command —1). In the first case, the cycle is repeated n+1 times, in the second case 1+n/2 times.
30	Иза	—	Operation of transformation of commands	Upon execution of commands, the content of cell a is sent over to the command register and is summed up with the next command.
37	Оса	—	Bringing the computer to a stop	The machine comes to a stop and the number in cell a is transcribed to the summator in direct code.
31	Λna_H 01С $00a_k$	—	Transcribing the info from the perforated tape into the ZU on the magnetic drum	The contents zone C (S) of the perforated tape is transcribed on the magnetic drum, in the group of cells a_n-a_k of the memory device.
31	ΛMa_H 030 $00a_K$	—	Transcription of information from the magnetic tape, to the magnetic drum of the memory device.	The content of the magnetic tape's zone C(s) is transcribd into the group of cells a_H-a_k on the magnetic drum of the memory device.
31	ΛMa_H 03С $00a_K$	—	Transcription of information from the memory device on the magnetic drum to the magnetic tape	The contents of the group of cells a_H-a_k on the magnetic drum of the memory device, is transcribed to the zone C (s) of the magnetic tape.
32	П Ч	—	Presentation of the summator's content to printing	When the "perforation" key is switched, the content of the summator is being perforated.
34	Ин	—	Formation of an interval in the printing on the paper	

Remarks: 1. z — is the content of the summator after execution of an operation.

2. When the machine executes some operations, it works out some special control signals: the signal of conditional transfer of the control ("signal ω") and the signal of overfill ("signal Z"). The conditions of working out the signal $\omega = 1$ are shown in the table.

3. The signal $z = 1$ is worked out at the operations C.1a, B.1a, Y.1a, and D.a, provided that $|z| > 1$. At $z = 1$, when the "blocking z" key is not switched in, then, depending upon the position of the key "stop by z", the machine is brought to a stop, otherwise, the control is transferred to the command Nr. 0001.

STAT

SUPPLEMENT No. 3

Literature:

1. Bazilevskiy, Yu. Ya. The universal computer for Engineering Research. "Priborostroyeniye" (Instrument Construction), No. 4, 1956.

2. Bazilevskiy, Yu. Ya. The Universal Electronic computer "Strela". "Priborostroyeniye, No. 3, 1957.

3. Belynskiy, V. V., Dolkart, V. M. Kagan, B. M. Lopato, G. P. and Matyukhin, N. Ya. The Small-size Electronic Computer M-3. Published by FVINTI, 1957.

4. Bruk, I. S. The High-Speed Electronic computer M-2. "Elektrichestvo" (Electricity), No. 9, 1956.

5. Bruk, I. S. Matyukhin, N. Ya. Belynskiy, V. V., Iosif'yan, A. G., Kagan, B. M. Dolkart, V. M, and Lopato, G. P. The Universal Small-Size Electronic computer M-3. "Elektrichestvo", No. 1,1958.

6. High-Speed Calculators. Translation from English under the supervision of D. Yu. Panov, 1952.

7. Venikov, V. A., Ivanov-Smolenskiy, A. V. and Gorushkin, V. I. On the Problem of Effectiveness of Forcing Generator Exitation. "Elektrichestvo" No. 1, 1955.

8. Venikov, V. A. and Litkens, I. V. On the Influence of Regulation of Induction Upon the Traffic-Carrying Capacity of Electric Long-Distance Power Transmissions. "Elektrichestvo", No. 11, 1955.

9. Zhdanov, P. S. Stability of Electric Systems. "Gosenergoizdat", 1948.

10. Kagan, B. M., Calculation of Stability of Electric Systems of Automatic Regulation by Means of digital computers. "Vestnik Elektropromyshlennosti" (Herald of the Electric Industry), No. 10, 1957.

11. Kagan, B. M., Gurin, Ya. S. and Dolkart V, M. Application of Automatic Electronic Computers for the Calculation of Electrical Machine Series. "Raboty Ministerstva elektrotekh.STAT

nicheskoy promyshlennosti SSSR po avtomatizatsii i mekhanizatsii narodnogo khozyaystva" (The Works of the USSR Ministry of the Electrotechnical Industry on Automation and Mechanization of the National Economy). Volume 2, 1956.

12. Keldysh, M. V. Lyapunov, A. A. and Shura-Bura, P. M. Mathematical Problems of the Theory of Calculating Machines. "Vestnik Akademii Nauk SSSR" (Herald of the Academy of Sciences of the USSR), No. 11, 1956.

13. Kitov, A. I. Electronic Digital Computers. Published by "Sovetskoye Radio" (Soviet Radio), 1956.

14. Kóllatz, L. Numerical Methods of Solving Differential Equations. Publishing House of Foreign Literature, 1953.

15. Krylov, A.N. Lectures on Approximate Calculations, Published by "Gostekhizdat", 1958.

16. Lebedev, S. A. Electronic Computers. Published by the AS USSR, 1956.

17. Lebedev, S. A. Electronic Computers. A lecture at a session of the AS USSR, 1956.

18. Lebedev, S. A. Examination of Artificially-Created Stability. Collection "Ustoychivost' elektricheskikh sistem" (Stability of Electric Systems). Works of VEI, Issue No. 40, "Gosenergoizdat", 1940.

19. Lyapunov, A. M. General Problem of Stability of Motion. "Gostekhizdat", 1958.

20. Miln V. E. Numerical Solution of Differential Equations. Publishing House of Foreign Literature, 1955.

21. Sakharov, I. Ye. and Ter-Mikaelyan, T. M. Calculation of Critical Speeds of Turbogenerator Rotors by Electronic computer. "Vestnik Elektropromyshlennosti", No. 10, 1957.

22. Fadeeva, V. N. Calculation Methods of Linear Algebra. "Gostekhizdat", 1956.

23. Hausholder, A. S. Foundations of Numerical-Analysis. Publishing House of Foreign Literature, 1956.

24. Tsukernik, L. V. and Kochanova, N. A. Analysis of Static Stability of Complex Energetic Systems by means of Electronic Computers. "Elektrichestvo", No. 7, 1957.

25. Tsypkin, Ya. Z. and Bromberg, P. V. About the Degree of Stability. Published by AS USSR, OTN, No. 12, 1945.

26. Eterman, I. I., Gorchinskaya, G. D. and Karavashkina, G. I. Solution of Mathematical Problems on the Universal Digital Computer "Ural". "Priborostroyeniye", No. 5, 1956.

27. Johnson, D. L., Ward, J. B. The Solution of Power System Stability Problems by Means of Digital Computers. Power Apparatus and Systems, 1957, No. 28.

28. Cypser, R. J., Computer Search for Economical Operation of a Hydrothermal Electric System, Power Apparatus and Systems, 1954, No. 14.

29. Glimn A. F., Kirchmayer, L. K., Habermann, R. J., Thomas R. W., Automatic Digital Computer Applied to Generation Scheduling, Power Apparatus and Systems, 1954, No. 14.

30. Glimn, A. F., Habermann, R. J., Henderson, J. M., Kirchmayer, L. K., Digital Calculation of Network Impedances, Power Apparatus and Systems, 1955, No. 21.

31. Ward, J. B., Hale, A. W., Digital Computer Solution of Power-Flow Problems, Power Apparatus and Systems, 1956, No. 24.

32. Williams, S. B., Abetti, P. A., Magnuson, E. F., Application of Digital Computers to Transformers Design, Power Apparatus and Systems, 1956, No. 25.

33. Veinott, C. G., Induction Machinery Design Being Revolutionized by the Digital Computer, Power Apparatus and Systems, 1957, No. 28.

34. Hunt, P. M., The Electronic Digital Computer in Aircraft Structural Analysis, Aircraft Engineering, 1956, 28, Part I, No. 325, Part II, No. 326.

35. Mazelesky, B., O'Connel, R. F., The Intergrated Use of Analog and Digital Computing Machines for Aircraft Dynamic Load Problems, J. Aeronaut. Sci., 1956, 23, No. 8.

STAT

STAT

36. Black, G., Electronic Computers in Optical Design, Research, 1956, 9, No. 8.

37. Lavesley, R.K., The Application of Electronic Digital Computer to Some Problems of Structural Analysis Struct. Engineer, 1956, 34, No. 1.

38. Coob, J. R., McIntire, R. L., Natural Gasoliners Like Robot Calculators, Oil Gas Journal, 1956, 54, No. 50.

==========

STAT

STAT

Б. М. КАГАН и Т. М. ТЕР-МИКАЭЛЯН

РЕШЕНИЕ ИНЖЕНЕРНЫХ ЗАДАЧ НА АВТОМАТИЧЕСКИХ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

ГОСЭНЕРГОИЗДАТ

Б. М. КАГАН и Т. М. ТЕР-МИКАЭЛЯН

РЕШЕНИЕ ИНЖЕНЕРНЫХ ЗАДАЧ НА АВТОМАТИЧЕСКИХ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

ГОСУДАРСТВЕННОЕ ЭНЕРГЕТИЧЕСКОЕ ИЗДАТЕЛЬСТВО
МОСКВА 1958 ЛЕНИНГРАД

33-5-4

В книге рассматриваются вопросы использования автоматических цифровых вычислительных машин (АЦВМ) для инженерных расчетов и исследований.

Изложены принципы действия АЦВМ и основы подготовки и программирования математических задач для АЦВМ. Приведены краткие сведения о методах приближенных вычислений.

Рассмотрены примеры применения АЦВМ для решения конкретных инженерных задач (исследования переходных процессов в дальних электропередачах, расчет устойчивости систем автоматического регулирования, исследования критических скоростей роторов турбогенераторов, расчет серии электромашин по критерию минимальной стоимости). Хотя все примеры относятся к электротехническим устройствам, излагаемые задачи носят характер общеинженерных проблем.

Книга предназначается для научных работников, инженеров, аспирантов и студентов старших курсов вузов.

* * *

Авторы: *Борис Моисеевич Каган* и *Теодор Михайлович Тер-Микаэлян*

РЕШЕНИЕ ИНЖЕНЕРНЫХ ЗАДАЧ НА АВТОМАТИЧЕСКИХ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

* * *

Редактор *В. М. Курочкин* Технич. редактор *Г. Е. Ларионов*

Сдано в пр-во 24/IV 1958 г. Подписано к печати 22/VII 1958 г.

Бумага 70×108¹/₁₆ 15,07 п. л. + 1 вкл. Уч.-изд. л. 17

Т-07156 Тираж 10 000 Цена в переплете № 5—9 р. 50 к., в переплете № 7—10 руб. Зак. 1199

Типография Госэнергоиздата. Москва, Шлюзовая наб., 10.

ПРЕДИСЛОВИЕ

Теоретическое исследование большинства вопросов, встречающихся в различных отраслях техники, сводится к математическим задачам, в которых точное решение либо не может быть найдено, либо имеет настолько сложный характер, что им трудно пользоваться при расчетах. Для решения таких задач были предложены различные приближенные методы, позволяющие получить ответ в численном виде. Однако большое число арифметических действий, необходимых для реализации указанных методов, еще до недавнего времени часто делали их практически неприменимыми.

Появившиеся за последние 10—15 лет быстродействующие электронные вычислительные машины резко расширили круг решаемых задач и в настоящее время проникли в практику не только научных, но и инженерно-технических исследований. Учитывая трудности, возникающие у инженеров, обращающихся в своей работе к помощи вычислительных машин, авторы сделали попытку написать книгу, дающую основные сведения о принципах действия автоматических цифровых вычислительных машин и возможностях их использования для инженерных исследований и расчетов. Книга имеет целью дать читателю руководство, достаточное для самостоятельного программирования и постановки на машине инженерных задач, допускающих численный метод решения.

Первые четыре главы книги знакомят читателя с устройством автоматических цифровых вычислительных машин и с методами программирования математических задач. Большое влияние на содержание гл. 3 и 4 и особенно § 4-1 оказали лекции проф. А. А. Ляпунова, читанные им в 1954—1955 гг. в Московском университете, а также круг идей, развитых проф. А. А. Ляпуновым и руководимой им группой математиков на семинаре по машинной математике в МГУ. Эти главы излагают лишь основы метода и не должны рассматриваться, как полный курс теории и практики программирования.

В гл. 5—8 описываются логические схемы программ ряда конкретных задач, на примере которых читатель может познакомиться с методикой решения инженерных задач на автоматических цифровых вычислительных машинах. Хотя большинство из рассматриваемых здесь вопросов носит характер общих инженерных проблем, таких как исследование переходных процессов, расчет устойчивости динамических систем, исследование резонансных явлений в сложных конструкциях, однако изложению этих вопросов придан определенный электротехнический уклон.

Вполне понятно, что авторы не

могли даже в малой степени отразить все многообразие инженерных задач, для решения которых привлекаются в настоящее время цифровые вычислительные машины. Вместе с тем авторы надеются, что читатель, познакомившийся с методами программирования, изложенными в гл. 3 и 4, и с некоторыми из примеров, рассмотренных в гл. 5-8, сможет самостоятельно подготовить и решить на машине любую нужную ему задачу. При этом, конечно, предполагается, что этой задаче придана математическая формулировка и имеется численный метод ее решения.

В гл. 9, которая носит справочный характер, даны краткие сведения о приближенных методах решения некоторых математических задач. Основная библиография по этому вопросу приведена в конце книги.

В 1956 г. коллективом сотрудников ЛУМС АН СССР, НИИ электротехнической промышленности Госплана СССР и АН Армянской ССР под общим руководством чл.-корр. АН СССР И. С. Брука и академика АН Армянской ССР А. Г. Иосифьяна была построена вычислительная машина М-3.

В книге обобщен опыт, накопленный авторами, участвовавшими в работе коллектива сотрудников ЛУМС АН СССР, НИИ ЭП Госплана СССР и АН Армянской ССР по созданию машины М-3 и проведению на машине М-3 в НИИ ЭП Госплана СССР и машине БЭСМ в ИТМиВТ АН СССР ряда инженерных исследований. Являясь одной из первых работ такого рода, книга не может быть свободна от недостатков. Авторы будут благодарны за все замечания и пожелания, которые читатели найдут нужным сделать.

В заключение авторы пользуются случаем выразить свою глубокую благодарность проф. А. А. Ляпунову и доц. В. М. Курочкину за ряд ценных указаний, которые авторы учли при подготовке рукописи к печати.

Главы 2, 6, 7, 8 и § 1-1, 1-2, 1-4 написаны Б. М. Каганом, главы 3, 4, 5, 9 и § 1-6, 1-7, 1-8, 1-9 написаны Т. М. Тер-Микаэляном, § 1-3 и 1-5 написаны авторами совместно.

Авторы

СОДЕРЖАНИЕ

ГЛАВА ПЕРВАЯ

ОСНОВНЫЕ ПОНЯТИЯ

1-1. Введение

Развитие техники требует увеличения мощности отдельных агрегатов и машин, интенсификации технологических процессов, повышения скоростей, температур, давлений, нагрузок в конструктивных материалах машин и аппаратов, увеличения надежности, быстродействия и точности работы различного рода устройств. Решение этих задач невозможно без глубокого и всестороннего исследования процессов, происходящих в машинах, аппаратах и сложных схемах.

Во многих случаях математические зависимости, описывающие процессы в исследуемых устройствах, благодаря сложности схемы, наличию элементов с распределенными параметрами и из-за явления насыщения и других нелинейностей оказываются столь сложными, что расчетное исследование, проводимое обычным способом, становится практически невозможным.

Инженер-исследователь, встретившись с подобными трудностями, часто может прибегнуть к физическому моделированию изучаемого явления. В настоящее время, например, эффективно используются физические модели электропередач для исследования процессов, происходящих в этих сложных системах. Физическое моделирование имеет особое значение тогда, когда отсутствует достоверное математическое описание изучаемого явления. Однако на такой модели можно изучать лишь определенные физические явления, причем сколько-нибудь существенные изменения параметров оригинала могут потребовать изготовления новой модели.

Новые возможности открывают современные вычислительные машины, которые можно разделить на две большие группы: а) электронные моделирующие устройства-аналоги; б) электронные быстродействующие цифровые вычислительные машины.

В электронных моделирующих устройствах-аналогах при помощи электронных ламп, конденсаторов, сопротивлений и некоторых других элементов создаются схемы, в которых изменения токов и напряжений во времени описываются теми же дифференциальными уравнениями, что и изучаемое явление. Решение уравнений получается в виде осциллограмм соответствующих напряжений в цепях моделирующей установки. Эти напряжения-аналоги переменных величин решаемой задачи изменяются во времени непрерывно, если это соответствует исходным уравнениям. Электронные моделирующие устройства называют также вычислительными машинами непрерывного действия. Точность их работы составляет 5—10%. Электронные моделирующие машины не являются универсальными устройствами. Они удобны для решения таких задач, которые в математическом отношении сводятся к обыкновенным дифференциальным уравнениям с постоянными и переменными коэффициентами[1]. Электронные моде-

[1] Существуют машины-аналоги для решения уравнений в частных производных.

лирующие устройства получили применение в основном при исследовании систем автоматического регулирования. Этому способствует возможность сравнительно просто осуществить сочленение моделирующей установки с реальной аппаратурой.

В последние годы получили широкое развитие автоматические быстродействующие цифровые вычислительные машины (АЦВМ). Эти машины оперируют с числовыми величинами, представленными в цифровой форме. В цифровой вычислительной машине величины не могут изменяться непрерывно. Они изменяются прерывисто, принимая отдельные (дискретные) значения. Поэтому эти машины называются машинами дискретного действия. В них процесс решения задачи разбивается на отдельные элементарные арифметические операции (сложение, вычитание, умножение, деление чисел). Управление процессом вычислений производится автоматически по заранее составляемой программе. Основные преимущества цифровых машин состоят в их универсальности и точности работы.

Быстродействующие цифровые вычислительные машины позволяют решать весьма широкий круг задач. Необходимо только, чтобы задача имела численный способ решения. Точность работы таких машин велика, так как вычисления производятся обычно с 9—10 десятичными знаками. Точность вычислений на цифровых вычислительных машинах ничем не ограничена и зависит лишь от количества разрядов в числах, с которыми машина оперирует.

Цифровые вычислительные машины работают с огромной скоростью, делая тысячи и десятки тысяч операций в секунду. Например, машина БЭСМ, построенная под руководством акад. С. А. Лебедева, выполняет 8—10 тыс операций в секунду.

Быстродействующие цифровые вычислительные машины могут выполнять не только арифметические, но и некоторые логические операции. Благодаря этому создается возможность автоматизации процесса счета, осуществления автоматического выбора (в зависимости от выполнения определенных условий) одного из нескольких заданных вариантов хода вычислений. Хотя вычисления на этих машинах требуют выполнения предварительной довольно трудоемкой работы по составлению программы расчета (так называемое «программирование задачи»), но затраченный труд вполне окупается, если решается сложная задача или производится расчет многих вариантов, например с целью нахождения оптимального решения.

Выполнение автоматической цифровой вычислительной машиной некоторых логических операций открывает новые возможности в построении систем автоматического управления и регулирования. Эти возможности реализуются путем включения вычислительной машины в состав системы автоматического управления.

Быстродействующие вычислительные машины являются мощным средством для исследований и расчетов при решении сложных инженерных задач в различных областях техники. Они создают возможность математического (численного) моделирования рабочих процессов в различного рода устройствах.

При создании новой техники много сил и средств тратится на построение и исследование различных физических моделей, изготовление промежуточных опытных образцов. Между тем, если известны дифференциальные уравнения, описывающие процессы в проектируемом устройстве, то на АЦВМ возможно в короткий срок рассчитать рабочие процессы для большого числа конструктивных вариантов и выбрать наилучший. При помощи АЦВМ можно расчетным путем определять наивыгоднейшие режимы работы сложного оборудования. Такой путь позволяет во многих случаях уменьшить объем экспериментальных исследований, работ по физическому моделированию и по испытанию промежуточных опытных образцов.

При исследовании рабочих процессов весьма важно отделить факторы, существенно влияющие на процесс, от второстепенных, которые могут быть опущены. Цифровые вычислительные машины могут быть использованы для расчета рабочих процессов при различных допущениях. Сопоставление полученных результатов позволяет определить разумные пределы для упрощающих предположений. После этого в ряде случаев подробное исследование вопроса может быть перенесено на более простые устройства — электронные машины непрерывного действия, расчетные столы и т. д.

Расчеты и исследования на АЦВМ в ближайшее время получат широкое применение при проектировании электротехнических устройств. В связи со строительством и вводом в эксплуатацию крупнейших электрических станций, работающих на дальние линии передачи, важное значение приобретают расчеты статической и динамической устойчивости систем, исследования влияния на их работу параметров генераторов и схем возбуждения синхронных генераторов. Решение этих задач, связанное с большими вычислительными трудностями, сравнительно просто производится на вычислительных машинах. Помимо проблем устойчивости, цифровые вычислительные машины в настоящее время используются для анализа ряда других сложных вопросов работы энергосистем.

На цифровых машинах выполняют исследования потерь в сетях, потокораспределения, расчеты экономического распределения нагрузок.

Большую перспективу имеет использование вычислительных машин для определения оптимальных по экономичности режимов работы энергосистем. Эти расчеты должны определять графики экономичного распределения нагрузок между отдельными тепловыми и гидравлическими станциями, входящими в систему, с учетом графиков общей нагрузки системы, потерь в линиях, стоимости топлива, уровней воды в водохранилищах, естественного стока, к. п. д. отдельных агрегатов. Возможность определения на АЦВМ экономичного распределения нагрузок создает перспективу автоматизации диспетчерского управления энергосистемами.

В области энергетического оборудования АЦВМ используются для расчетов при проектировании атомных реакторов, для расчетов теплового баланса турбин и выбора оптимального режима работы энергоустановок, для исследований в области прочности и вибраций конструкций.

Известно, какое важное значение для турбогенераторов большей мощности имеет определение критических скоростей роторов, выяснение связи величин критических скоростей с конструктивными параметрами ротора и опор.

Из-за трудоемкости расчетов практически невозможно обычными методами провести сколько-нибудь полное исследование этого вопроса. Вычислительные машины позволяют решить эту задачу. Подобным же образом вычислительные машины могут быть применены для расчетов резонансных частот щитов турбогенераторов, режимов работы подпятников гидрогенераторов, для исследования температурных полей, скоростей воздуха в каналах и других вопросов нагрева и вентиляции крупных электрических машин.

Цифровые вычислительные машины находят применение для расчета и исследования устойчивости и различных режимов работы сложных систем автоматического управления и регулирования, для исследования динамических процессов в сложных системах электропривода, в схемах регулирования и управления, содержащих различные нелинейные устройства — дроссели насыщения, ртутные выпрямители и др. Они могут быть использованы для расчета оптимальных процессов в системах автоматического управления при наличии определенных ограничений (ограничение скорости, момента и т. п.), для расчетов по синтезу систем автоматического управления с заданными характеристиками.

Большой интерес представляет расчетное исследование на АЦВМ динамики систем автоматического управления, находящихся под воздействием непрерывно изменяющихся случайных сигналов. Это связано с возможностью использования АЦВМ для выработки случайных величин с различными законами распределения.

Достигнутый прогресс в области конструирования вычислительных машин выдвинул как одну из важных задач современного этапа развития техники — использование АВЦМ в качестве элемента систем автоматическо-

го управления («управляющие вычислительные машины») и создание на этой основе систем комплексной автоматизации технологических процессов.

В связи с этим большое практическое значение имеет применение универсальных АЦВМ, предназначенных для инженерных и научных расчетов, для расчетных исследований, или, иначе говоря, для численного моделирования процессов управления в системах автоматизации с управляющими вычислительными машинами. Универсальные АЦВМ при наличии специальных преобразователей непрерывных величин в дискретную (цифровую) форму и обратно могут в ряде случаев соединяться с реальными объектами и использоваться при экспериментальных исследованиях в качестве макетов управляющих вычислительных машин.

Вычислительные машины целесообразно применять при расчетах серий электрических машин и трансформаторов. При этом оказывается возможным, исходя из заданных номинальных данных (мощность, к п д и др.) и определенных критериев (минимальный вес, минимальная стоимость и др.), отыскивать оптимальные размеры и обмоточные данные электрических машин и трансформаторов.

Мы перечислили лишь немногие инженерные задачи из области электротехники и некоторых смежных областей, для решения которых привлекаются цифровые вычислительные машины.

Приведем несколько примеров применения АЦВМ для инженерных расчетов в других областях техники. В области авиационной техники АЦВМ используются для расчетов прочности, для изучения проблем вибрации и флаттера в самолетостроении, для расчетов наилучшей формы крыла самолета, сопла реактивного двигателя, для исследования вопросов, связанных с взлетом, посадкой, катапультированием самолетов, определения скорости на взлете и траектории и для решения других проблем.

При помощи АЦВМ рассчитываются баллистические таблицы.

В строительном деле АЦВМ могут применяться для расчетов сложных ферм, мостов, зданий, плотин и т. д. Например, на машине БЭСМ были определены формы контуров наиболее крутых, неосыпающихся откосов каналов.

При помощи АЦВМ можно выполнять расчеты давлений в сложных гидравлических системах и в газовых сетях.

В нефтяной промышленности АЦВМ применяются для решения таких задач, как определение конфигурации нефтяных пластов, определение наиболее эффективных режимов перегонки нефти в зависимости от свойств первичного сырья и др.

Большую перспективу имеет применение АЦВМ для экономических расчетов и исследований в области планирования производства, анализа хода производства, расчетов себестоимости, цен, заработной платы и т. д.

Возможность применения универсальных АЦВМ для решения инженерных задач нисколько не уменьшает значения использования при некоторых расчетах других вычислительных устройств, таких как модели электрических систем, электронные машины непрерывного действия. Например, расчеты потокораспределения в сложных энергосистемах удобнее производить на моделях электрических сетей. Для многих задач в области автоматического управления и регулирования следует отдать предпочтение машинам непрерывного действия — аналогам, не требующим составления программы решения задачи и допускающим непосредственное сочленение вычислительной машины с реальной аппаратурой. В настоящее время развиваются методы расчета, использующие в различных комбинациях АЦВМ и модели электрических сетей, АЦВМ и машины-аналоги.

1-2. О численных методах решения математических задач

Исследование научно-технических проблем в математическом отношении обычно сводится к отысканию и анализу решений дифференциальных уравнений. Хотя очень многие процессы в технике можно достаточно точно описать дифференциальными уравнениями, однако лишь в очень редких случаях решение этих уравнений удается получить аналитическим путем (в «замкнутой форме»). Существуют численные (приближенные) методы решения математических задач, не требующие отыскания аналитических решений. При использовании этих методов решение сложных математических задач сводится к некоторой последовательности арифметических операций, выполняемых в определенном порядке. Численные методы решения математических задач излагаются в гл. 9. Здесь мы ограничимся одним простым примером.

Рассмотрим хорошо известное инженерам-электрикам нелинейное дифференциальное уравнение[1]

$$M_0 \frac{d^2\theta}{dt^2} = P_0 - P_m \sin\theta, \quad (1\text{-}1)$$

где M_0, P_m, P_0 — постоянные.

Решение уравнения (1-1) не может быть выражено в конечном виде через элементарные функции. Для решения этого уравнения можно воспользоваться численными методами, например методом конечных приращений Эйлера (метод последовательных интервалов*).

Пусть угловые относительная скорость и ускорение обозначены

$$\Omega = \frac{d\theta}{dt}, \quad \alpha = \frac{d^2\theta}{dt^2}$$

и заданы начальные условия

$$t = 0, \ (\theta)_0 = \theta_0, \ \left(\frac{d\theta}{dt}\right)_0 = (\Omega)_0 = 0.$$

Необходимо решить уравнение (1-1), т. е. определить $\theta = f(t)$ при $0 \leqslant t \leqslant T$.

Заданный диапазон значений t разбивают на достаточно малые интервалы времени — шаги Δt — и считают, что на каждом интервале ускорение постоянно и равно его значению в начале интервала. Индексами l и $l+1$ можно обозначить значения величин соответственно в начале и в конце $l+1$-го интервала. Величина ускорения α_l, соответствующая началу $l+1$-го интервала, определяется выражением

$$\alpha_l = \frac{P_0 - P_m \sin\theta_l}{M_0}. \quad (1\text{-}2)$$

Тогда приращения скорости $\Delta\Omega_{l+1}$ и угла $\Delta\theta_{l+1}$ в течение $l+1$-го интервала будут:

$$\left.\begin{aligned} \Delta\Omega_{l+1} &= \alpha_l \Delta t; \\ \Delta\theta_{l+1} &= \Omega_l \Delta t + \alpha_l \frac{\Delta t^2}{2}. \end{aligned}\right\} \quad (1\text{-}3)$$

В конце $l+1$-го интервала относительная скорость и угол примут следующие значения:

$$\left.\begin{aligned} \Omega_{l+1} &= \Omega_l + \Delta\Omega_{l+1} = \Omega_l + \alpha_l \Delta t; \\ \theta_{l+1} &= \theta_l + \Delta\theta_{l+1} = \theta_l + \Omega_l \Delta t + \alpha_l \frac{\Delta t^2}{2}. \end{aligned}\right\} \quad (1\text{-}4)$$

Наконец, имеем еще одно соотношение

$$t_{l+1} = t_l + \Delta t. \quad (1\text{-}5)$$

При ручном счете интегрирование уравнения (1-1) производится следующим образом. Зная θ_l и Ω_l в начале $l+1$-го интервала (для первого интервала $\theta_l = \theta_0$, $\Omega_l = 0$), определяют по таблицам $\sin\theta_l$ и находят ускорение α_l по формуле (1-2). Затем по формулам (1-3) — (1-5), действуя в определенной последовательности, находят Ω_{l+1}, θ_{l+1}, t_{l+1}. После окончания вычисления для одного интервала переходят к следующему. При определении значения ускорения в начале следующего интервала подставляют в (1-2) значение θ_{l+1}, полученное для конца предыдущего интервала, и т. д. Перед переходом к вычислениям для следующего интервала надо сравнить величину t_{l+1} с заданным пределом интегрирования T, и если $t_{l+1} < T$, то необходимо выполнить следующий шаг интегрирования, в противном случае вычисления прекращаются, так как расчет уже закончен.

Таким образом, процесс интегрирования уравнения (1-1) может быть сведен к выполнению определенной последовательности арифметических операций и одной простой логической — сравнению величин t_{l+1} и T. В зависимости от результата этого сравнения расчет либо продолжается, либо прекращается.

Последовательность арифметических и логических операций, которые надо выполнить над исходными данными и

[1] Уравнение (1-1) описывает движение ротора синхронной машины. θ — угол между вектором э. д. с. холостого хода генератора и напряжением приемной системы.

над результатами промежуточных вычислений, чтобы получить ответ, называется алгоритмом решения математической задачи [Л. 12].

1-3. Блок-схема АЦВМ

При решении большинства математических задач численными методами необходимо произвести огромное число арифметических операций. Еще недавно выполнение подобных расчетов требовало таких больших затрат труда, что решение многих задач было практически неосуществимым. За последние полтора десятилетия созданы электронные автоматические цифровые вычислительные машины (АЦВМ), работающие с огромной скоростью. В АЦВМ посредством программного управления процесс вычислений полностью автоматизирован.

На рис. 1-1 представлена упрощенная блок-схема АЦВМ. Машина состоит из следующих основных узлов: арифметического узла, запоминающего устройства, устройства ввода данных в машину и устройства вывода данных из машины, управляющего устройства.

Арифметический узел АУ производит операции над поступающими в него числами. Скорость работы арифметических узлов современных вычислительных машин составляет тысячи арифметических операций в секунду. Можно сказать, что арифметический узел подобен арифмометру, работающему с громадной скоростью.

При решении задач на арифмометре или настольной вычислительной машине оператор от руки набирает на них нужные числа, включает машину, а затем выписывает на бумагу полученный результат. Применение подобного метода в случае электронных машин сделало бы совершенно бессмысленной высокую скорость работы их арифметического узла. Человек не может с нужной скоростью вводить числа в арифметический узел и считывать результат операций. Поэтому эти процессы автоматизируются при помощи так называемого «запоминающего устройства» ЗУ или, как кратко говорят, «памяти».

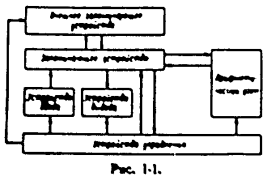


Рис. 1-1.

Запоминающее устройство состоит из ряда отдельных ячеек, в каждой из которых хранится одно или несколько чисел. Ячейкам присвоены номера, позволяющие отличать их друг от друга. Эти номера называются «адресами» ячеек ЗУ.

Из запоминающего устройства числа передаются в арифметический узел машины («чтение числа»), а полученный в арифметическом узле результат операции помещается в запоминающее устройство («запись числа»). Время, необходимое на передачу числа из ЗУ в АУ или обратно, называется временем обращения к запоминающему устройству. Это время должно быть соразмерно со скоростью работы арифметического узла для эффективного использования возможностей последнего.

Схемы запоминающих устройств выполняются таким образом, чтобы после чтения числа из ячейки запоминающего устройства содержание ее не изменялось. Если это число понадобится нам в дальнейшем, его можно вновь взять из той же ячейки. В то же время при передаче нового числа в ячейку памяти ранее хранившееся в ней число стирается и на его место записывается новое число.

Чтобы полностью автоматизировать весь вычислительный процесс и совершенно исключить участие человека в работе машины, АЦВМ снабжаются устройством управления УУ, которое управляет всем вычислительным процессом и, в частности, передает числа из ЗУ в АУ, включает АУ на выполнение требуемой операции и помещает полученный результат в ЗУ.

Как было отмечено в предыдущем параграфе, всякий численный метод сводит решение математической задачи к ряду последовательных арифметических действий и логических операций над числами как заданными в условии задачи, так и получающимися в процессе счета. Так как управляющее устройство цифровой вычислительной машины само управляет всем процессом вычислений, то предварительно должно быть составлено точное описание того, какие действия, в каком порядке и над какими числами должны быть выполнены. Такое описание всего процесса счета называют программой решения данной задачи на автоматической цифровой вычислительной машине. Автоматическое программное управление является основной принципиальной особенностью быстродействующих цифровых вычислительных машин.

Программа состоит из отдельных «команд» (говорят также «приказов» или «инструкций»), которые указывают на то, какое конкретное действие и над какими числами должна выполнить машина на данном этапе счета. Эти команды последовательно охватывают все операции, которые надо выполнить для решения на машине данной задачи выбранным численным методом. Совокупность команд, необходимых для решения задачи, записанных в определенной последовательности, образует программу.

Сказанное можно пояснить на элементарном примере. Пусть, например, необходимо вычислить определитель второго порядка

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Для этого нужно умножить число a на d, затем умножить число b на c и вычесть второе произведение из первого. Иными словами, выполняются следующие действия:

1	×	a	d	ad
2	×	b	c	bc
3	−	ad	bc	$ad - bc$

Здесь в первом столбце стоят символ операции, которую нужно выполнить, во втором и третьем столбцах стоят числа, над которыми эти действия выполняются, и, наконец, в четвертом столбце выписаны результаты действия.

Таким образом, программа вычисления определителя второго порядка состоит из трех отдельных команд. Однако непосредственно в таком виде эта программа не может быть реализована на машине. Для этого необходимо программу закодировать в такой форме, чтобы машина могла ее «прочесть» и выполнить.

В следующих параграфах мы рассмотрим те операции, которые должна выполнять автоматическая цифровая вычислительная машина, чтобы с их помощью можно было составить программу любого вычислительного процесса, и ознакомимся со способом кодирования и хранения чисел и команд в машине.

Программа и числа помещаются в запоминающее устройство машины при помощи «устройства ввода». Наконец, для печатания полученных результатов машина снабжена «устройством вывода *B*».

В современных вычислительных машинах большая скорость вычислений (тысячи операций в секунду) достигается путем выполнения запоминающего устройства из двух узлов (рис. 1-1): а) внутреннего быстродействующего запоминающего устройства ЗУ на сравнительно небольшое число ячеек (обычно от одной до четырех тысяч); б) внешнего запоминающего устройства ВЗУ, сравнительно медленно работающего, но способного хранить большое количество чисел (несколько десятков или даже сотен тысяч).

В этом случае во внешнее запоминающее устройство вводится весь материал, необходимый для решения данной задачи. В ходе вычислений части программы и константы, соответствующие отдельным этапам решения задачи, переписываются во внутреннюю память. Таким образом, собственно вычислительный процесс происходит без обращений к внешней памяти.

В следующей главе будут рассказано о работе элементов и узлов автоматической цифровой вычислительной машины.

Хотя каждая вычислительная машина обладает рядом специфических

(и часто весьма существенных) особенностей, тем не менее принципиальные свойства всех цифровых вычислительных машин с программным управлением в основном одни и те же. Поэтому в настоящей главе все рассуждения будут проведены для некоторой условной машины.

1-4. Системы счисления

В повседневной жизни применяется десятичная система счисления, в которой для изображения чисел употребляются десять знаков (цифр): 0; 1, 2, 3, 4, 5, 6, 7, 8, 9, и в каждом разряде числа может стоять любая из этих цифр. Иными словами, в десятичной системе всякое число представляют в виде суммы степеней числа десять, причем коэффициенты при них равны числу единиц в соответствующих разрядах десятичного числа. Например, число $37406{,}15 = 3 \cdot 10^4 + 7 \cdot 10^3 + 4 \cdot 10^2 + 0 \cdot 10^1 + 6 \cdot 10^0 + 1 \cdot 10^{-1} + 5 \cdot 10^{-2}$.

В быстродействующих цифровых вычислительных машинах для представления чисел и команд часто применяется двоичная система счисления. В этой системе в каждом разряде двоичного числа может стоять лишь цифра 0 или 1. В двоичной системе каждое число представляется в виде суммы степеней числа 2, причем коэффициентами при степенях числа 2 могут быть либо 0, либо 1. Например, десятичное число $21{,}5 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1}$ и в двоичной системе счисления имеет вид 10101,1. В табл. 1-1 приводятся первые 17 десятичных и двоичных чисел.

Для изображения двоичных чисел необходимо располагать бо́льшим числом разрядов, чем для тех же чисел в десятичной системе. Тем не менее

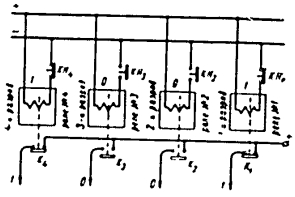


Рис. 1-2

применение двоичной системы позволяет уменьшить общее количество аппаратуры и создать большие удобства для проектирования цифровых вычислительных машин, так как для представления в машине разряда двоичного числа может быть использован любой элемент, имеющий всего два устойчивых состояния. Такими элементами, например, являются реле, триггерные схемы и т. д.

Можно принять, что замкнутое состояние реле соответствует единице, а разомкнутое — нулю. В цепи контакта реле единице может соответствовать наличие, а нулю — отсутствие напряжения. Например, имея четыре реле (рис. 1-2) и принимая, что состояние реле № 1 изображает цифру первого двоичного разряда, реле № 2 — второго двоичного разряда и т. д., можно при помощи кнопок *КН* представить при помощи этих реле в двоичной форме любое целое число от 0 до 15. На рис. 1-2 включены реле № 1 и № 4, что соответствует двоичному числу 1001 (десятичному числу 9).

Сложение двух чисел в двоичной системе, как и в десятичной системе, можно производить столбиком. При этом в каждом разряде при сложении руководствуются правилами: $0+0=0$; $0+1=1+0=1$; $1+1=0+$ единица переноса в старший разряд. Например, операция сложения для двух чисел $23+25=48$, представленных в двоичной системе, производится следующим образом:

$$\begin{array}{r} 10111 \\ +\ 11001 \\ \hline 110000 \end{array}$$

Важное достоинство двоичной системы состоит в исключительной простоте ее таблицы умножения:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Например, операция умножения $6 \times 5 = 30$ в двоичной системе записи выглядит следующим образом:

$$\begin{array}{r} 110 \\ \times\ 101 \\ \hline 110 \\ 000 \\ 110 \\ \hline 11110 \end{array}$$

Таким образом, операция умножения сводится к операции сдвига и сложения. При этом частные произведения получаются путем сдвига множимого влево на число разрядов, соответствующее номерам значащих разрядов множителя.

Помимо двоичной и десятичной систем счисления используются также восьмеричная и шестнадцатиричная системы счисления, имеющие основанием соответственно числа 8 и 16. В восьмеричной системе в каждом разряде может стоять любая цифра от 0 до 7. Десятичное число 3011 в восьмеричной записи имеет вид: $5703 = 5 \cdot 8^3 + 7 \cdot 8^2 + 0 \cdot 8^1 + 3 \cdot 8^0$.

В шестнадцатиричной системе для изображения чисел употребляются 15 цифр, при этом приходится вводить новые обозначения для цифр, бо́льших девяти. Например, можно обозначить десять 0, одиннадцать 1, двенадцать 2, тринадцать 3, четырнадцать 4, пятнадцать 5. В шестнадцатиричной системе десятичное число 3011 запишется следующим образом: $\overline{123} = 1 \cdot 16^2 + 2 \cdot 16^1 + 3 \cdot 16^0$.

Основание любой системы счисления, записанное в этой же системе, имеет вид 10 (число два в двоичной системе есть 10, число восемь в восьмеричной системе есть 10 и т. д.).

Восьмеричные и шестнадцатиричные числа легко преобразуются в двоичные и, наоборот, двоичные числа просто преобразовать в восьмеричные или шестнадцатиричные. Это связано с тем, что основания восьмеричной и шестнадцатиричной систем есть целые степени числа 2: $8 = 2^3$, $16 = 2^4$. Для перевода восьмеричного числа в двоичную форму достаточно заменить каждую цифру восьмеричного числа соответствующим трехразрядным двоичным числом. Таким же образом для перехода от шестнадцатиричной системы к двоичной каждая цифра шестнадцатиричного числа заменяется соответствующим четырехразрядным двоичным числом. Например, восьмеричное число 5703 в двоичной форме записи имеет вид:

$$\underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{000}_{0}\ \underbrace{011}_{3}$$

а шестнадцатиричное число $\overline{123}$ в двоичной системе запишется следующим образом:

$$\underbrace{1011}_{\overline{1}}\ \underbrace{1100}_{\overline{2}}\ \underbrace{0011}_{3}$$

При переходе от двоичной к восьмеричной (или шестнадцатиричной) системе, последовательно начиная с младших разрядов для целой части числа и со старших — для дробной части, заменяем группы по три (или четыре) двоичных разряда соответствующими цифрами восьмеричного (или шестнадцатиричного) числа.

Рассмотренные числа 101 111 000 011 и 1011 1100 0011 можно считать соответственно восьмеричным и шестнадцатиричным числами, но в которых цифра каждого разряда записана в двоичной системе. Такие формы записи чисел носят названия *двоично-восьмеричной* и *двоично-шестнадцатиричной систем*. Они называются также *двоично-кодированными системами*.

В вычислительных машинах применяется также *двоично-десятичная система изображения чисел*. В этой системе каждая цифра десятичного разряда записывается в виде соответствующего четырехразрядного двоич-

Таблица 1-1

Десятичные числа	0	1	2	3	4	5	6	7	8	9	10	11
Двоичные числа	0	1	10	11	100	101	110	111	1000	1001	1010	1011
Десятичные числа	12		13		14		15		16		17	
Двоичные числа	1100		1101		1110		1111		10000		10001	

ного числа, например число 952 в этой системе имеет вид 1001 0101 0010

В настоящее время для большинства вычислительных машин основной системой счисления является двоичная. Двоичная система используется в машинах для представления и хранения чисел и команд и при выполнении арифметических операций. Восьмеричная и шестнадцатиричная системы используются при составлении программ вычислений для более короткой и удобной записи двоичных чисел, так как эти системы не требуют специальных операций для перевода в двоичную систему.

Постоянные величины («константы») — начальные условия, коэффициенты и т. п., необходимые для решения задачи, вводятся в машину в восьмеричной (шестнадцатиричной) или двоично-десятичной системах. В последнем случае перевод двоично-десятичных чисел в двоичные выполняется машиной по специальной программе. Переход от десятичной к двоично-десятичной системе производится вне машины на устройствах для пробивки программы на ленте (перфорация и др.).

Результаты расчета выводятся из машины в восьмеричной (шестнадцатиричной) или десятичной системах, причем в качестве промежуточных систем, используемых внутри узла вывода данных применяются двоично-кодированные формы указанных систем. Перевод таких из двоичной системы в двоично-десятичную производится машиной по специальной программе.

1-3. Машины с плавающей и фиксированной запятой

Каждый разряд двоичного числа изображается в автоматической цифровой вычислительной машине при помощи какого-нибудь технического устройства, например реле. Машина содержит лишь конечное число таких устройств, и поэтому она может оперировать только с числами определенной длины, т. е. с числами, содержащими определенное число разрядов. Выбор числа разрядов определяется, с одной стороны, требованиями к точности решения задач, а с другой — конструктивными соображениями.

Принципиально цифровая вычислительная машина может обеспечить любую точность вычислений за счет соответствующего увеличения числа разрядов, используемых для представления чисел. Но чрезмерное увеличение числа разрядов приводит к увеличению объема аппаратуры и большим конструктивным трудностям. В цифровых вычислительных машинах для представления чисел обычно используются 30—40 двоичных разрядов.

Диапазон чисел, который может быть представлен в машине при данном числе разрядов, существенно зависит от принятой формы представления чисел. В машинах используются две формы представления чисел: а) с «фиксированной запятой»; б) с «плавающей запятой». В зависимости от используемой формы различают машины с «фиксированной запятой» и машины с «плавающей запятой».

В машинах с фиксированной запятой положение запятой, отделяющей целую часть числа от дробной, фиксируется между определенными разрядами числа и сохраняется неизменным при всех вычислениях. Обычно запятая находится перед крайним слева, т. е. перед старшим, разрядом. В этом случае в машину могут быть помещены и машина может оперировать лишь с числами, по модулю меньшими единицы.

Для изображения знака числа выделяется специальный разряд, например, стоящий слева от запятой. При этом знаку плюс в знаковом разряде обычно соответствует нуль, а знаку минус — единица.

Пусть, например, количество двоичных разрядов в машине равно 34, причем один разряд используется для записи знака числа. Тогда разрядная сетка числа имеет вид, изображенный на рис. 1-3. В рассмотренном случае в машине могут быть представлены числа в диапазоне: от $-(1-2^{-33})$ до -2^{-33}, от $+2^{-33}$ до $+(1-2^{-33})$ и число нуль. Число, модуль которого меньше 2^{-33}, но больше нуля, не может быть представлено в машине и воспринимается как нуль (выход числа из разрядной сетки машины вправо). Число, по модулю большее, чем $(1-2^{-33})$, также нельзя представить в машине из-за выхода числа из разрядной сетки машины влево (как говорят, происходит переполнение разрядов). Если в процессе вычислений появится такое число, то его старшие разряды (разряды слева от запятой) потеряются и результат вычислений окажется неверным. Во избежание этого обычно предусматривается автоматическая остановка машины при появлении числа, большего или равного единице.

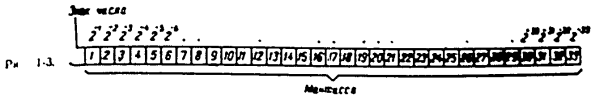


Рис. 1-3.

При программировании задач для вычислительных машин с фиксированной запятой приходится вводить специальные масштабные коэффициенты, чтобы все исходные, промежуточные и конечные величины были бы меньше единицы.

Ошибка в результате вычислений для операций сложения и вычитания зависит от абсолютной, а для умножения и деления — от относительной точности представления чисел в машине. Абсолютная точность представления числа определяется количеством используемых разрядов. При фиксированной запятой относительная точность представления числа тем ниже, чем меньше число. Поэтому масштабные коэффициенты нельзя выбирать слишком большими, так как в этом случае снижается точность вычислений.

В машинах с плавающей запятой все числа изображаются в виде:

$$X = 2^p \cdot A, \quad |A| < 1, \qquad (1\text{-}6)$$

где p — целое число (положительное, отрицательное или нуль), называемое порядком числа X;

A — мантисса числа X.

Величина порядка определяет положение запятой в числе. Если число A удовлетворяет неравенству

$$|A| \geq \frac{1}{2}, \qquad (1\text{-}7)$$

то говорят, что X — «нормализованное число». Последнее неравенство означает, что у нормализованного числа первый разряд после запятой всегда единица. В машинах с плавающей запятой изображаются отдельно порядок p и отдельно мантисса A. При этом разрядную сетку машины делят на две части: одну часть разрядов отводят для изображения порядка p, а остальную часть для изображения мантиссы A. Не нужно забывать, что у нормализованных чисел не только мантисса A, но и порядок p могут быть обоих знаков. Например, число $\frac{1}{8}$ в нормализованном виде изображается, как $2^{-2} \cdot \frac{1}{2}$ и т. п. Поэтому необходимо отвести по одному разряду для представления знака порядка и знака мантиссы.

В дальнейшем будет рассмотрена некоторая условная машина с плавающей запятой и с разрядной сеткой, содержащей 42 разряда. Пусть разрядная сетка чисел в этой машине имеет вид, представленный на рис. 1-4. Левые шесть разрядов отводятся под изображение порядка (из них один — под его знак), а остальные 36 — под мантиссу (из них один — под ее знак).

Пятью разрядами можно записать числа от 0 до $2^5 - 1 = 31$, поэтому порядок p будет принимать целые значения от -31 до $+31$. В нормализованном виде мантисса может принимать значения от $0{,}\overbrace{100\ldots0}^{36}$ до $0{,}\overbrace{111\ldots1}^{36}$, т. е. от 2^{-1} до $2^{-1}+2^{-2}+\ldots+2^{-36} = 1-2^{-36}$.

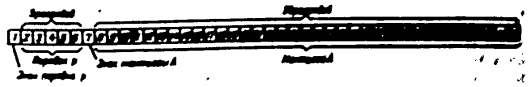


Рис. 1-4.

Наименьшее по модулю нормализованное число, представляемое в нашей машине, будет $2^{-k}\cdot\frac{1}{2}=2^{-k-1}$. Числа, по модулю меньшие 2^{-k-1}, уже не могут быть изображены и приравниваются равными нулю. Наибольшее по модулю нормализованное число будет $2^{2^k-1}(1-2^{-2^k})\approx 2^{2^k-1}$. Из сказанного следует, что в рассматриваемой машине могут быть представлены числа, лежащие в диапазоне от $-2^{2^k-1}(1-2^{-2^k})$ до -2^{-k-1} и от $+2^{-k-1}$ до $+2^{2^k-1}(1-2^{-2^k})$, и кроме того, число — нуль. Так как 2^{-k-1} и 2^{2^k-1} приблизительно равны 10^{-n} и 10^{n}, то рабочим диапазоном чисел машины будут интервалы от -10^{n} до -10^{-n} и от $+10^{-n}$ до $+10^{n}$, иначе говоря, числа представляются с точностью до n десятичных знаков. Для большинства встречающихся на практике задач такой диапазон чисел оказывается достаточным.

Диапазон чисел, с которыми может оперировать машина с плавающей запятой, определяется числом разрядов, отведенных под порядок, а точность — числом разрядов, отведенных под мантиссу. Фактически при одном и том же числе разрядов мантиссы точность вычислений на машине с плавающей запятой больше, чем на машине с фиксированной запятой. Это объясняется тем, что при фиксированной запятой относительная точность представления чисел уменьшается с уменьшением числа, а при плавающей запятой благодаря нормализации относительная точность сохраняется постоянной для всех чисел в рабочем диапазоне машины.

Таким образом, система представления чисел с плавающей запятой по сравнению с системой с фиксированной запятой позволяет получить больший рабочий диапазон чисел, большую точность вычислений (при одинаковом числе разрядов мантиссы) и кроме того, упрощает программирование.

Однако осуществление в машине системы представления чисел с плавающей запятой требует добавления аппаратуры для представления порядков и для операций с порядками. Поэтому обычно плавающая запятая используется в больших вычислительных машинах, а малые вычислительные машины выполняют с фиксированной запятой.

Если в процессе вычислений образуются ненормализованные числа, то машина (с плавающей запятой) автоматически нормализует их[1]. Если старшие k разрядов мантиссы равны нулю, то операция нормализации состоит в сдвиге мантиссы влево на k разрядов [чтобы выполнялось условие (1.7)] и соответствующем уменьшении порядка числа на k единиц. Например, чтобы нормализовать двоичное число $2^{10}\cdot 0{,}0001011$, надо сдвинуть мантиссу влево на три разряда и на три единицы уменьшить порядок числа. В результате получается то же число, но в нормализованном виде $2^{7}\cdot 0{,}1011000$.

При умножении чисел в машинах с плавающей запятой порядки чисел складываются, а мантиссы перемножаются. Затем полученное произведение нормализуется. Аналогичным образом при делении порядок делителя вычитается из порядка делимого, мантисса делимого делится на мантиссу делителя и полученный результат (частное) нормализуется. При сложении и вычитании чисел в машинах с плавающей запятой предварительно выравниваются порядки чисел. Порядок меньшего числа делается равным порядку большего числа, а мантисса этого числа соответственно сдвигается вправо на число разрядов, равное разности порядков чисел, после чего мантиссы складываются (или вычитаются). Порядок суммы (или разности) принимается равным порядку большего числа. Все эти операции автоматически выполняются в арифметическом узле машины.

В запоминающем устройстве машины могут храниться как нормализованные, так и ненормализованные числа. К последним, в частности, относятся, как мы увидим из дальнейшего, команды.

1-6. Кодирование команд

Как уже не раз отмечалось, цифровые вычислительные машины предназначены для выполнения арифметических действий над числами. При этом для эффективного использования скорости, с которой арифметический узел выполняет операции, нужно, чтобы эти числа хранились в самой машине. Следовательно, в запоминающем устройстве машины должны храниться не только исходные числовые данные задачи, но и результаты промежуточных вычислений, которые используются при дальнейшем ходе решения. Если, например, для решения дифференциального уравнения

$$y'=f(x,\ y),$$
$$x_0\leqslant x\leqslant X,\ y(x_0)=y_0$$

используется формула Эйлера

$$y_{i+1}=y_i+hf(x_i,\ y_i),$$

то в машине должны храниться числа x_0, y_0, X, шаг h и, кроме того, значения x_i и y_i, полученные на предыдущем шаге интегрирования.

В примере, приведенном в § 1.3, нужно было сохранять в машине не только элементы определителя a, b, c и d но и произведения ad и bc, чтобы иметь возможность затем вычесть второе произведение из первого. Таким образом, каждая команда должна содержать информацию о том, какое действие должно быть выполнено, над какими числами и куда должен быть помещен ответ. Программа решения задачи, т. е. последовательность команд, полностью описывающая весь вычислительный процесс, также должна храниться в машине. Следовательно, командам должен быть придан такой вид, при котором они легко могут быть помещены в машину.

Пусть, например, нужно вычесть число b из числа a. Числа a и b хранятся в определенных ячейках памяти, пусть это будут соответственно ячейки α и β. Кроме того, нужно в какую-то ячейку памяти поместить разность $a-b$, например в ячейку γ. Нужно особо подчеркнуть, что α, β, γ — это номера ячеек запоминающего устройства, т. е. числа, которыми перенумерованы все ячейки памяти.

Рассматриваемой операции будет соответствовать команда, содержащая следующие данные: «вычесть из числа, хранящегося в ячейке α, число, хранящееся в ячейке β, и поместить результат в ячейку γ». Схематически эта команда может быть записана в следующем виде:

—	α	β	γ
вычесть	a	b	$a-b$

Здесь в первой клетке стоит символ операции, которую машина должна выполнить, в двух последующих — номера ячеек, в которых хранятся числа, и, наконец, в последней клетке — номер той ячейки, в которую помещается результат. Если бы было нужно разделить число a на b, то схематическое изображение такой команды приняло бы вид

	α	β	γ
разделить	a	b	$a:b$

Мы уже отмечали, что α, β, γ — это обычные числа. Пусть, например, α есть третья ячейка, т. е. ячейка, двоичный номер которой есть 11, β — пятая ячейка, ее двоичный номер есть 101, и, наконец, γ — десятая ячейка, ее двоичный номер есть 1010. Тогда последняя команда примет вид:

	11	101	1010

Нам остается придумать для операции деления код, заменяющий символ «:» и удобный для хранения в машине. Таким кодом может быть какое-нибудь двоичное число, например 10. Теперь уже рассматриваемая команда имеет вид

10	11	101	1010

(1-8)

и изобразится в виде следующего набора нулей и единиц:

10111011010.

Таким образом, если каждой арифметической операции вместо привычного для нас символа $+$, $-$, $\times$, $:$ придать некоторый числовой «код операции» (конечно, каждой операции свой определенный), то всякая команда примет вид обычного двоичного числа.

Если бы α, β и γ были соответственно семнадцатой, тридцать второй и тридцать третьей ячейкой, т. е. если бы их двоичными номерами были числа

[1] Если операция нормализации не нужна, то она блокируется.

10001, 100000 и 1101, то предыдущая команда в цифровой записи приняла бы вид

$$10103011000031101 \qquad (1\cdot9)$$

С изменением номеров ячеек число разрядов в цифровой записи команды изменилось. Это обстоятельство приводит к значительным неудобствам при расшифровке такой записи. Например, в первом случае ячейку α изображают третий и четвертый разряды, а во втором — разряды с третьего по восьмой. Следовательно, для каждой команды нужно заново указывать, что именно изображают разряды ее цифровой записи. Поэтому для изображения команд выбирают одинаковое число разрядов, так чтобы все они имели постоянную длину. При этом следует руководствоваться двумя соображениями.

Во-первых, как было указано выше, устройство управления автоматически управляет всем вычислительным процессом, а для этого программа решения задачи должна храниться в машине. Так как каждой команде уже придан вид некоторого двоичного числа, то для хранения в машине программы достаточно было бы снабдить устройство управления специальным запоминающим устройством. В ячейках этого запоминающего устройства могли бы храниться все команды программы. Однако конструктивно выгоднее помещать программу в запоминающее устройство, уже имеющееся в машине для хранения чисел. Поэтому при составлении цифрового кода команды мы должны учитывать длину ячеек памяти нашей условной машины, равную 42 разрядам.

Во-вторых, в качестве α, β и γ могут быть взяты любые ячейки памяти, следовательно, под каждую из этих ячеек должно быть отведено число разрядов, достаточное для изображения наибольшего номера ячейки. Если, например, запоминающее устройство нашей условной машины содержит тысячу ячеек, то под изображение каждого из чисел α, β и γ должно быть отведено не менее десяти разрядов.

Действительно, это число двоичных разрядов необходимо для записи всех чисел от 0 до 1000, а α, β и γ могут, конечно, принимать все эти значения. Точно так же под код операции мы должны отвести количество разрядов, допускающее изображение достаточного числа различных операций, выполняемых машиной.

В силу этих соображений каждая команда программы будет храниться в отдельной ячейке и, следовательно, команда будет изображаться при помощи 42 разрядов. При этом под код операции отводятся те шесть разрядов, которые служат для изображения порядка числа (удобство такого выбора станет ясным в гл. 3). Оставшиеся в ячейке 36 разрядов разбиваются на три группы по 12 разрядов каждая, изображающие соответственно номера ячеек α, β и γ. Эти три группы разрядов называются соответственно первым, вторым и третьим адресами команды.

Таким образом, можно сказать, что команда состоит из кода операции, которую должна выполнить машина, адресов чисел, над которыми эта операция выполняется, и адреса, по которому записывается результат. Схематически распределение разрядов в команде изобразится табличкой

Код операции 6 разрядов	IA 12 разрядов	IIA 12 разрядов	IIIA 12 разрядов

где, например, IA читается, как „первый адрес".

Если номер какой-нибудь ячейки не использует всех 12 разрядов, то свободные в данном адресе разряды заполняются нулями. Полученное двоичное число называется кодом команды. Например, кодами приведенных выше команд (1·8) и (1·9) будут соответственно числа

000010	000000000011	000000000101	000001001010
000010	000000010001	000001000100	000000001101
6 разрядов	12 разрядов	12 разрядов	12 разрядов

Теперь вся программа может быть изображена в виде последовательности двоичных чисел и помещена в запоминающее устройство машины. Это является одним из самых существенных свойств вычислительных машин с программным управлением. В § 1·8 будет показано, каким образом устройство управления машины «отличает» ячейки, в которых хранятся команды программы, от ячеек, в которых хранятся обычные числа.

Под изображение модуля порядка числа отведены 5 разрядов и 1 разряд отведен под знак порядка. Так как различным знакам соответствует наличие нуля или единицы в знаковом разряде, то фактически порядки чисел изображаются 6-разрядными двоичными числами. В 6 двоичных разрядах можно записать $2^6 = 64$ различных числа 0,1,2, ,63.

Эти же 6 разрядов используются для изображения кода операции. Поэтому машина может выполнять не более 64 различных операций (каждая операция должна иметь свой отдельный код). Обычно современные машины выполняют меньшее число операций, так как с увеличением числа выполняемых операций растут конструктивные трудности при осуществлении устройства управления. Поэтому не всякое двоичное число, даже если оно имеет установленное число разрядов (в нашем примере 42), может рассматриваться как код какой-то команды.

Из рассмотренных выше примеров видно, что коды команды, рассматриваемые как числа, могут быть ненормализованными. Следовательно, в запоминающем устройстве будут храниться как нормализованные, так и ненормализованные числа.

Как указывалось выше, каждый адрес является номером той ячейки, в которой хранится нужное для выполнения данной команды число. Следовательно, должна быть обеспечена возможность указания в каждом из трех адресов номера любой ячейки запоминающего устройства. Иными словами, в рассматриваемой машине оперативное запоминающее устройство не может иметь больше чем $2^{12} = 4\,096$ ячеек, так как для изображения каждого адреса отведены 12 разрядов.

Обычно, помимо оперативного запоминающего устройства, вычислительная машина снабжается внешним запоминающим устройством (ВЗУ на рис. 1·1), содержащим несколько десятков тысяч ячеек. Номера этих ячеек мы не можем помещать в адреса рассмотренных выше команд, но при помощи специальных операций содержимое этих ячеек может быть переписано в оперативное запоминающее устройство. Таким образом достигается увеличение числа ячеек памяти без увеличения длины кода команды.

Машины, команды которых имеют описанный выше вид, называются машинами с трехадресной системой управления. Существуют также одно-, двух- и четырехадресные машины.

При одноадресной системе каждая команда содержит лишь код операции и один адрес. При этом для изображения арифметической операции над двумя числами с записью результата в запоминающее устройство требуется в общем случае три команды. Например, операция сложения чисел a и b, расположенных соответственно в ячейках α и β, с передачей суммы в ячейку γ запишется следующим образом:

M	α
N	β
R	γ

Здесь M, N, R — коды операций: M — передача содержимого ячейки α на регистр арифметического узла; N — сложение числа, расположенного на регистре, с числом, хранящимся в ячейке β; R — передача числа из регистра в ячейку γ запоминающего устройства. Одноадресная система программирования используется, например, в вычислительной машине «Урал» (см. § 2·12).

При двухадресной системе каждая команда содержит код операции и адреса первого и второго чисел. Результат операции записывается по одному из адресов, например по второму. Такая система используется в машине М-3 (см. § 2·11).

Применяется также другой вариант двухадресной системы, в которой первый адрес используется, как в одноадресных машинах, а второй адрес указывает номер ячейки, в которой хранится следующая команда.

Наконец, при четырехадресной системе команд первые три адреса ис-

пользуются так же, как в трехадресных командах, а четвертый адрес указывает номер ячейки, в которой хранится следующая команда. Подробнее этот вопрос здесь не будет рассматриваться.

1-7. Некоторые операции, выполняемые цифровыми машинами

Основными операциями, которые выполняют цифровые вычислительные машины, являются четыре арифметических действия: сложение, вычитание, деление и умножение чисел. Но помимо этих операций, автоматические цифровые вычислительные машины выполняют еще ряд других операций, с которыми мы познакомимся в настоящем и следующем параграфах.

Весьма удобной для программирования является операция логического умножения (обозначается символом $\wedge$), которая определяется следующим образом. В одном разряде логическое умножение производится по тем же правилам, что и обычное умножение, т. е.

$$1 \wedge 1 = 1, \quad 1 \wedge 0 = 0 \wedge 1 = 0 \wedge 0 = 0.$$

Результатом логического умножения двух n-разрядных двоичных чисел $\alpha_n \ldots \alpha_1$ и $\beta_n \ldots \beta_1$ есть n-разрядное число $\gamma_n \ldots \gamma_1$, в котором каждый разряд γ_i есть результат логического умножения цифр α_i и β_i, стоящих в том же разряде сомножителей. Таким образом, логическое умножение есть поразрядная операция. Это значит, что результат действия в каждом разряде не зависит от значения прочих разрядов сомножителей. Например,

$$\begin{array}{r} 1010 \\ \wedge\ 1101 \\ \hline 1000 \end{array}$$

Если почему-либо нужно выделить некоторые k-тый и l-тый разряды двоичного числа $\alpha_n \alpha_{n-1} \ldots \alpha_2 \alpha_1$, то достаточно это число умножить логически на n-значное число, у которого в разрядах k и l стоят единицы, а в остальных разрядах нули:

$$\begin{array}{r} \alpha_n \alpha_{n-1} \ldots \alpha_{k+1} \alpha_k \alpha_{k-1} \ldots \alpha_{l+1} \alpha_l \alpha_{l-1} \ldots \alpha_1 \\ \wedge\ 0\ 0\ \ldots\ 0\ 1\ 0\ \ldots\ 0\ 1\ 0\ \ldots\ 0 \\ \hline 0\ 0\ \ldots\ 0\ \alpha_k\ 0\ \ldots\ 0\ \alpha_l\ 0\ \ldots\ 0 \end{array}$$

Часто приходится пользоваться операцией сдвига числа вправо или влево на некоторое число разрядов. Если n-значное число $\alpha_n \ldots \alpha_1$ сдвинуть вправо на k разрядов, то в первые k разрядов поступят нули и получится число

$$\underbrace{0\ 0\ \ldots\ 0}_{k}\ \alpha_n \alpha_{n-1} \ldots \alpha_{k+1}$$

При сдвиге того же числа на k разрядов влево образуется число

$$\alpha_{n-k} \alpha_{n-k-1} \ldots \alpha_1\ \underbrace{0\ 0\ \ldots\ 0}_{k}$$

Часто бывает нужно число, находящееся в одной ячейке, поместить в другую ячейку, для чего используется операция передачи числа из одной ячейки в другую. Иногда из одной ячейки в другую передается лишь модуль этого числа или же передается число с обратным знаком. Можно передавать число из какой-нибудь ячейки запоминающего устройства в нее же. Этим, например, пользуются, когда хотят изменить знак числа достаточно выполнить передачу числа с обратным знаком из ячейки, в которой оно хранится, в ту же ячейку.

Для вывода чисел из запоминающего устройства машины применяется команда печати, по которой машина при помощи выводного устройства отпечатывает число, хранящееся в определенной ячейке памяти.

Рассмотрим, наконец, операцию взятия целой и дробной частей некоторого числа a, находящегося в ячейке a. Целой частью не отрицательного числа a называется такое целое число $[a]$ (читается «антье» a), для которого разность

$$a - [a]$$

есть не отрицательное число, меньшее единицы. Например,

$$[3,7] = 3, \quad [6] = 6, \quad [3/4] = 0.$$

Разность $a - [a]$ называют дробной частью числа a и обозначают $\{a\}$. Например,

$$\{3,7\} = 0,7, \quad \{6\} = 0, \quad \{3/4\} = 3/4.$$

Если же число a отрицательное, то под целой частью понимается такое целое отрицательное число $[a]$, что разность

$$a - [a] = \{a\}$$

есть положительное число, меньшее 1. Число $\{a\}$ называется дробной частью отрицательного числа. Например

$$[-5] = -5, \quad [-3,7] = -4,$$

$$\{-3,7\} = 0,3,$$

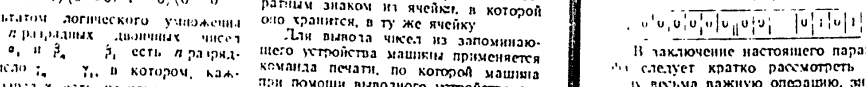
$$[-0,1] = -1; \quad \{-0,1\} = 0,9$$

При этом дробная часть $\{a\}$ записывается в какую-нибудь ячейку β как ненормализованное число $2^r \{a\}$, т. е. в разряды порядка ячейки β записывается нуль, а в разряды мантиссы — число $\{a\}$. Целая же часть числа записывается в последние разряды какой-нибудь другой ячейки γ. Таким образом, команда эта имеет вид

$$\boxed{1\ 1\ \|\ \alpha\ |\ \beta\ |\ \gamma}$$

Если, например, в ячейке α находится число $a = 5\frac{1}{4}$ или в двоичной записи $a = 101{,}01$, то после этой операции в ячейках β и γ будут храниться числа

Знак +



В заключение настоящего параграфа следует кратко рассмотреть еще одну весьма важную операцию, значение которой станет ясным в гл. 3. Как мы указывали в § 1-5, при сложении двух нормализованных чисел $2^p \cdot A$ и $2^q B$ машина предварительно приравнивает их порядки, а потом уже складывает мантиссы, т. е. считая, что $p > q$, выполняет следующие действия:

$$2^p A + 2^q B = 2^p A + 2^{q-p} B =$$

$$= 2^p A + 2^p \frac{B}{2^{p-q}}.$$

Если в результате получается число ненормализованное (числа A и B могут быть разных знаков), то машина автоматически нормализует получаемую сумму.

В машинах применяется также операция условного сложения, при которой складываются отдельно порядки и отдельно мантиссы чисел. Эта операция носит название сложения команд и обозначается символом СК.

Таким образом,

$$(2^p A) \cdot \text{СК} \cdot (2^q B) = 2^{p+q} (A + B).$$

1-8. Операции управления

Наибольшее распространение в АЦВМ получили такие системы управления, в которых после выполнения команды, расположенной в ячейке с номером k, запоминающего устройства, машина автоматически переходит к выполнению команды, записанной в ячейке $k+1$ и т. д. Изменение этого нормального порядка чередования выполняемых машиной команд производится посредством операций управления.

Если программа решения какой-то задачи содержит n команд, то их можно поместить в n ячеек памяти с последовательными номерами, например в ячейки $c+1$, $c+2$, ..., $c+n$. При этом первая команда программы помещается в ячейку с номером $c+1$, вторая — в ячейку $c+2$ и т. д.

Устройство управления[1] имеет специальный адресный регистр (счетчик). В нем образуется адрес очередной команды, которая должна быть прочтена из запоминающего устройства и выполнена машиной[1].

Оператор с пульта управления набирает в адресном регистре номер $c+1$ (номер той ячейки, в которой хранится первая команда программы) и затем включает машину. Устройство управления согласно номеру ячейки, установленному на адресном регистре, выбирает команду из ячейки $c+1$, выполняет ее и добавляет в адресный регистр единицу, где таким образом, образуется номер $c+2$. Затем машина переходит к выполнению команды, находящейся в ячейке $c+2$ и т. д. Таким образом, будут последовательно выполнены все команды программы.

Однако часто бывает необходимо, чтобы, выполнив первые k команд программы, т. е. выполнив команды, хранящиеся в ячейках $c+1 - c+k$, машина перешла к команде, хранящейся в ячейке с некоторым номером p. Иными словами, последовательный порядок выполнения команд прерывается, например, нужно возвратиться к уже выполненной ранее команде. Для этого используется так называемая операция «безусловного перехода»

[1] Об устройстве управления см. § 2-6.

ции «безусловной передачи управления». Эта операция записывается в виде

$c+k$	ПУ	p	—	—
$c+k+1$				

и читается «передать управление команде, расположенной в ячейке p».

Выполнение операции безусловного перехода состоит в том, что в адресный регистр устройства управления передается число p, после чего обычного прибавления единицы не происходит.

Следовательно, после команды безусловной передачи управления (команды, хранящейся в ячейке $c+k$) мы переходим не к команде, которая хранится в последующей ячейке $c+k+1$ как обычно, а к команде, хранящейся в ячейке p, номер которой указан по первому адресу команды безусловной передачи управления. Таким образом осуществляется безусловное переключение вычислительного процесса от одного места программы к другому каждый раз, когда мы доходим до команды, хранящейся в ячейке $c+k$.

В отличие от рассмотренной безусловной передачи управления существует так называемая условная передача управления, для реализации которой используется операция сравнения двух чисел, имеющая следующий вид (предполагаем, что эта команда хранится в ячейке с номером m)

m	$\leqslant$	α	β	γ
$m+1$				

Смысл этой операции сводится к следующему: если число, хранящееся в ячейке α (по первому адресу), меньше числа, хранящегося в ячейке β (по второму адресу), то следующей выполняется команда, хранящаяся не в ячейке $m+1$, а в ячейке γ, указанной по третьему адресу операции сравнения. Если же число, хранящееся в ячейке α, больше или равно числу, хранящемуся в ячейке β, то следующей выполняется команда из ячейки $m+1$. Как говорят, операция сравнения «отсылает» по третьему адресу в случае, если первое число меньше второго, и «пропускает» — в противном случае. Здесь переключение счета от одного места программы к другому происходит в зависимости от выполнения определенного условия (соотношения величин двух чисел). Поэтому эта операция называется условной передачей управления. Как мы увидим в гл. 3, операция условной передачи управления играет принципиальную роль в автоматизации процесса вычислений на цифровых вычислительных машинах.

Практически операция условной передачи управления осуществляется на основе определения знака разности чисел, хранящихся в ячейках α (по первому адресу) и β (по второму адресу). Если знак этой разности отрицательный, в адресный регистр управляющего устройства передается число γ, в противном случае в этот регистр, как обычно, прибавляется единица.

Такая же условная передача управления может осуществляться при помощи сравнения двух чисел по модулю, которая «отсылает», если модуль первого числа меньше модуля второго, и «пропускает» — в противном случае. Введем также операцию условной передачи управления, которая будет «пропускать» в случае равенства двух чисел и «отсылать» в противном.

Наконец, к операциям управления относится операция, останавливающая машину, когда все вычисления окончены и программа исчерпана.

После сказанного в настоящем параграфе становится ясно, каким образом устройство управления машины отличает ячейки, в которых хранятся числа (константы), от ячеек, в которых хранятся команды, хотя те и другие являются обычными числами и как таковые ничем друг от друга не отличаются.

Действительно, номер ячейки, в которой хранится первая команда, мы сами от руки набираем на пульте управления. Кроме того, вся программа либо расположена в последовательных ячейках памяти, либо сама переключает управление к тем ячейкам, в которых хранятся команды, и так до тех пор, пока не будет пройдена вся программа, последняя команда которой останавливает машину.

1-9. Код команд условной машины

В предыдущих параграфах были рассмотрены основные операции, выполняемые вычислительными машинами. В зависимости от конструктивных особенностей различные машины могут иметь различный состав выполняемых операций.

Ниже приводится список операций для некоторой условной трехадресной машины.

Как видно из приведенной таблицы, некоторые операции, как, например, операция передачи чисел, не используют всех трех адресов. В коде такой команды в разрядах отсутствующего адреса пишут нули. Если, например, операции передачи числа присвоен код 010001, то код команды, передающей число из ячейки 0.....01 в ячейку 10.....01, будет иметь вид:

$$\underbrace{010001}_{6}\ \underbrace{0\ \ldots 01}_{12}\ \underbrace{00\ldots\ldots 0}_{12}\ \underbrace{10\ldots\ldots 01}_{12}$$

Код команд условной машины

№	Символ операции	ГА	ПА	ША	Содержание операции
1	+	α	β	γ	Число из ячейки α прибавляется к числу из ячейки β и сумма записывается в ячейку γ
2	-	α	β	γ	Число, хранящееся в ячейке β, вычитается из числа, хранящегося в ячейке α, и разность записывается в ячейку γ
3	×	α	β	γ	Число из ячейки α умножается на число из ячейки β и произведение записывается в ячейку γ
4	:	α	β	γ	Число из ячейки α делится на число из ячейки β и частное записывается в ячейку γ
5	СК	α	β	γ	Сложение команд. В ячейке γ образуется число, порядок которого равен сумме порядков чисел, хранящихся в ячейках α и β, а мантисса—сумме их мантисс
6	[ ]	α	β	γ	Взятие целой и дробной части. Дробная часть числа из ячейки α в ненормализованном виде записывается в ячейку β, а целая—в последние разряды ячейки γ
7	$\wedge$	α	β	γ	Число из ячейки α логически умножается на число из ячейки β и результат записывается в ячейку γ
8	←	α	k	γ	Сдвиг влево. Число из ячейки α сдвигается влево на k разрядов и результат записывается в ячейку γ. Правые k разрядов ячейки γ заполняются нулями
9	→	α	k	γ	Сдвиг вправо. Число из ячейки α сдвигается на k разрядов вправо, и результат записывается в ячейку γ. Левые k разрядов ячейки γ заполняются нулями
10	ПЧ	α	—	γ	Передача числа. Число из ячейки α передается в ячейку γ
11	|ПЧ|	α	—	γ	Передача по модулю. Модуль числа из ячейки α передается в ячейку γ
12	−ПЧ	α	—	γ	Передача с обратным знаком. Число из ячейки α передается в ячейку γ с обратным знаком
13	Печать	α	—	—	Число, хранящееся в ячейке α, печатается
14	ПУ	α	—	—	Передача управления. Следующей выполняется команда, хранящаяся в ячейке α
15	<	α	β	γ	Сравнение чисел. Если число, хранящееся в ячейке α, меньше числа, хранящегося в ячейке β, то следующей выполняется команда, хранящаяся в ячейке γ. В противном случае следующей выполняется команда, номер которой на единицу больше номера данной команды
16	|<|	α	β	γ	Сравнение по модулю. Если модуль числа, хранящегося в ячейке α, меньше модуля числа, хранящегося в ячейке β, то следующей выполняется команда, хранящаяся в ячейке γ. В противном случае следующей выполняется команда, номер которой на единицу больше номера данной команды
17	≠	α	β	γ	Сравнение на равенство. Если число, хранящееся в ячейке α, не равно числу, хранящемуся в ячейке β, то следующей выполняется команда, хранящаяся в ячейке γ. Если же эти числа равны, то следующей выполняется команда, номер которой на единицу больше номера данной команды
18	Стоп	—	—	—	Счет прекращается и машина останавливается

ГЛАВА ВТОРАЯ

ПРИНЦИПЫ ДЕЙСТВИЯ АВТОМАТИЧЕСКИХ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

2-1. Понятие о последовательном и параллельном кодах

В большинстве цифровых вычислительных машин в настоящее время применяется двоичная система счисления. Как уже отмечалось, в двоичной системе употребляются лишь две цифры 0 и 1, и поэтому для представления двоичного числа в машине могут быть использованы различные элементы (механические, электрические, магнитные и т. п.), имеющие два устойчивых состояния. Например, таким элементом может служить электромагнитное реле. Включенному состоянию реле можно приписать значение *1*, а выключенному — значение *0*, или наоборот. Точно таким же образом запертому состоянию электронной лампы, когда ток через нее не проходит, можно приписать, например, значение *1*, а открытому состоянию лампы — значение *0*.

В электрическом отношении *0* и *1* представляются («кодируются») в машине либо высоким и низким уровнем напряжения в соответствующей точке схемы (потенциальный код), либо положительным и отрицательным электрическими импульсами или наличием и отсутствием электрического импульса на соответствующем зажиме схемы (импульсный код). В качестве примера на рис. 2-1 число 1011001 представлено в потенциальном (*а*) и импульсном (*б*) кодах.

Приведем некоторые понятия, характеризующие сигналы в виде импульсов и в виде уровней напряжения.

Импульс можно характеризовать амплитудой импульса U_A, шириной (продолжительностью) импульса по основанию $t_и$, амплитудой выброса $U_в$ (рис. 2-2,*а*). Передним и задним фронтами импульса называются соответственно времена нарастания и спадания импульса $t_ф$ и $t_з$.

Аналогичные понятия могут быть применены к потенциальному сигналу (рис. 2-2,*б*). Потенциальный сигнал характеризуется, кроме того, разностью U_C верхнего и нижнего уровней напряжений. Понятия переднего и заднего фронта у потенциального сигнала всегда связаны с процессом перехода соответственно от нижнего к верхнему и от верхнего к нижнему уровням напряжений (сравнить рис. 2-2,*б* и *в*).

В вычислительных машинах применяются два способа кодирования чисел (и команд). Рассмотрим, например, код отрицательного десятиразрядного двоичного числа 1,1011010011. Здесь двоичный разряд перед запятой служит для указания знака числа (знак плюс изображается нулем, а минус — единицей). Двоичный код рассматриваемого числа может быть представлен в виде некоторой временной последовательности импульсов, проходящей по одной цепи. Пусть единице соответствует наличие, а нулю отсутствие импульса. Тогда в определенные моменты времени, задаваемые вспомогательными синхронизирующими импульсами, по цепи будут последовательно один за другим проходить сигналы, соответствующие значениям цифр в разрядах числа (рис. 2-3). Такой способ кодирования чисел, когда код числа развертывается во времени, называется «последовательным кодом» или «временным кодом».

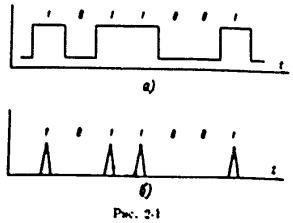


Рис. 2-1

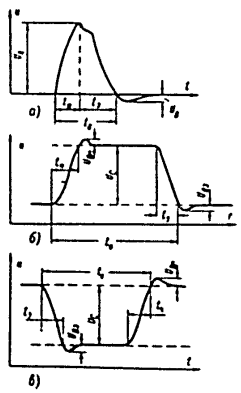


Рис. 2-2

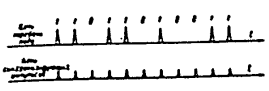


Рис. 2-3.

При другом способе кодирования — при «параллельном коде» (или «пространственном коде») код числа развертывается не во времени, а одновременно в нескольких электрических цепях (т. е. в пространстве). Количество цепей равно числу разрядов числа. В один и тот же момент во всех цепях возникают сигналы, указывающие на значения цифр в соответствующих разрядах двоичного числа (рис. 2-4).

В зависимости от применяемого кода вычислительные машины называются машинами последовательного или параллельного действия. Существуют вычислительные машины смешанного действия, в которых в одних узлах применяется параллельный код (например, в арифметическом узле), а в других — последовательный (например, в запоминающем устройстве).

При последовательном коде операции с числами, в том числе и передача чисел из одного устройства в другое, производятся поочередно для каждого разряда числа, и поэтому машины последовательного действия работают медленнее, чем машины параллельного действия. Однако машины параллельного действия требуют большего объема аппаратуры, так как при параллельном коде надо иметь столько цепей передачи сигналов, воспринимающих и решающих элементов, из скольких разрядов (с учетом разряда знака) состоят числа, с которыми оперирует машина.

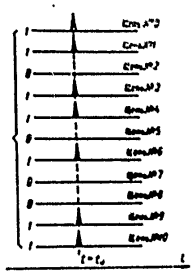


Рис. 2-4.

2-2. Основные электронные элементы АЦВМ

Триггер или электронное реле является одним из основных элементов электронных цифровых вычислительных машин. Схема триггера (рис. 2-5) состоит из двух ламп, аноды и сетки которых соединены между собой цепями обратной связи.

При уменьшении анодного тока, проходящего через электронную лампу, уменьшается падение напряжения на сопротивлении в ее анодной цепи, следовательно, увеличивается напряжение на аноде. Наоборот, увеличение анодного тока сопровождается уменьшением напряжения на аноде лампы.

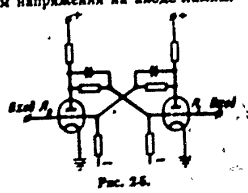


Рис. 2-5.

В триггерной схеме благодаря наличию цепей обратной связи при уменьшении анодного тока одной лампы повышается потенциал на сетке второй лампы, и ее анодный ток увеличивается. В свою очередь, увеличение тока второй лампы приводит к уменьшению потенциала сетки первой лампы, что вызывает дальнейшее уменьшение тока первой лампы и т. д. Этот процесс происходит лавинообразно до тех пор, пока первая лампа не закроется полностью, а вторая лампа — не откроется. При этом сетка первой лампы получает отрицательный, а сетка второй лампы — положительный потенциал. Если отрицательный импульс поступит на сетку второй лампы, имевшей до этого момента положительный потенциал, ток второй лампы начнет уменьшаться и произойдет «опрокидывание» схемы: первая лампа откроется, а вторая закроется.

Таким образом, триггерная схема имеет два устойчивых состояния, причем в одном из них анодное напряжение лампы $Л_0$ высокое, а лампы $Л_1$ — низкое, а в другом состоянии, наоборот. Если одному устойчивому состоянию триггера присвоить значение нуля, а другому — единицы, то каждый разряд двоичного числа может быть представлен в машине соответствующим триггером. Примем, например, что состояние триггера соответствует нулю, когда лампа $Л_0$ имеет высокое анодное напряжение, и единице, когда высокое анодное напряжение имеет лампа $Л_1$.

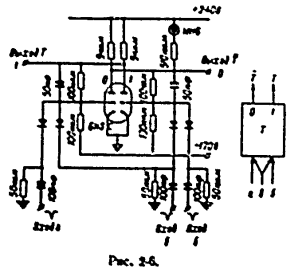


Рис. 2-6.

Управление триггерными схемами осуществляется подачей отрицательных импульсов на сетку. Для того чтобы поставить триггер в положение *0*, достаточно подать отрицательный импульс на сетку лампы $Л_0$. Если до подачи импульса триггер был в состоянии *1*, то отрицательный импульс перебрасывает его в состояние *0*. Точно так же для сообщения триггеру состояния *1* достаточно подать отрицательный импульс на сетку лампы $Л_1$.

Для изображения двоичного числа надо иметь столько триггеров, сколько разрядов в числе, и поставить каждый триггер в состояние *0* или *1* в зависимости от значения соответствующего разряда. Такая схема из нескольких триггеров, используемая для хранения двоичного числа, называется триггерным регистром.

Если соединить через вентили сетки обеих ламп триггера и подавать на общую точку отрицательные импульсы, то каждый импульс будет перебрасывать схему из одного устойчивого состояния в другое, а состояние схемы будет указывать, является ли общее число импульсов четным или нечетным. Легко заметить, что в рассматриваемом случае триггер работает как двоичный счетчик.

Свойства триггеров, позволяющие использовать их для хранения двоичных чисел и для построения счетчиков, обусловливают широкое применение триггерных схем в цифровых вычислительных машинах. На рис. 2-6 показана одна из используемых на практике схем триггеров[1]. Справа дано условное изображение триггера, употребляемое в функциональных схемах узлов вычислительной машины. Напряжение на аноде закрытой лампы триггера на рис. 2-6 составляет около 200 *в*, а на аноде открытой — около 100 *в*. Триггер имеет два выхода и три входа. Высокий уровень напряжения на выходе T имеет место, когда триггер находится в состоянии *1*. В состоянии *0* на выходе схемы $\overline{T}$ будет высокий уровень напряжения. Параллельно выходу $\overline{T}$ присоединена сигнальная неоновая лампа. Лампа загорается, когда левая лампа триггера открыта, т. е. триггер находится в состоянии *1*. При подаче на вход *а* отрицательного импульса триггер примет состояние *0*. Точно таким же образом после подачи отрицательного импульса на вход *б* триггер будет находиться в состоянии *1*. Если нужно изменить состояние триггера на противоположное, используется вход *в*, являющийся общим для обеих ламп триггера. Вход *в* называется «счетным входом» триггера, так как этот вход используется при работе триггеров в схемах счетчиков, когда каждый следующий импульс, приходящий на данный триггер, должен перебрасывать триггерный счетчик данного разряда в противоположное состояние. Для управления триггером, изображенным на рис. 2-6, используются треугольные остроконечные импульсы амплитудой 40—60 *в* и шириной у основания 3—4 *мксек*. Вентили во входных цепях триггера разделяют цепи сеток триггера от цепей источника импульсов. Временная диаграмма напряжения для триггерной схемы показана на рис. 2-7.

[1] На рис. 2-6 показана схема триггера вычислительной машины М-3 [Л. 3].

В приведенной выше триггерной схеме состояния схемы *0* или *1* определяются уровнями напряжения на анодах ламп. В последнее время в электронных вычислительных машинах нашли применение так называемые «динамические» триггеры, в которых состояния схемы (*0* или *1*) определяются отсутствием или наличием колебаний (импульсов) на выходе схемы. На рис. 2-8 приведена схема динамического триггера [Л. 17].

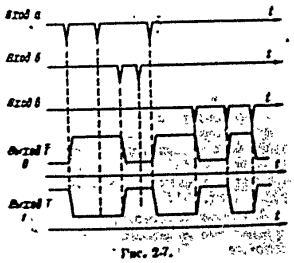


Рис. 2-7.

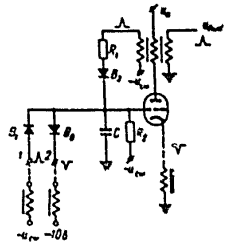


Рис. 2-8.

На катод лампы непрерывно поступают синхронизирующие отрицательные импульсы, понижающие напряжение катода. Триггер устанавливается в состояние *1* подачей положительного импульса на зажим *1*. Через диод B_1 импульс заряжает конденсатор C, чем резко повышается потенциал сетки лампы. В этом случае величина отрицательных синхронизирующих импульсов на катоде оказывается достаточной для открытия лампы. Во вторичных обмотках выходного трансформатора при этом генерируются положительные импульсы. С одной вторичной обмотки положительные импульсы поступают в схему вычислительного устройства, а с другой через сопротивление R_1 и диод B_2 осуществляют подзарядку конденсатора C и тем самым поддерживается рассматриваемый режим работы схемы.

Триггер перебрасывается в состояние *0* подачей отрицательного импульса на зажим *2*. При этом конденсатор C разряжается через диод B_3. На сетке лампы резко возрастает запирающее напряжение, и синхронизирующие импульсы уже не могут пройти через лампу.

В рассматриваемой схеме конденсатор C как бы является «запоминающей» ячейкой. Уровень напряжения конденсатора меняется при изменении состояния схемы, и это напряжение может быть использовано также как анодное напряжение обычных (потенциальных) триггеров для управления

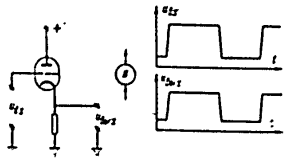


Рис. 2-9.

другими элементами вычислительного устройства.

При использовании для управления внешней цепью уровня напряжения конденсатора C быстродействие триггера, изображенного на рис. 2-8, определяется не частотой синхронизирующих импульсов, а продолжительностью входных импульсов нуля и единицы.

Схема динамического триггера на рис. 2-8 содержит всего одну лампу. Достоинством таких схем является высокое входное и низкое выходное сопротивления, сравнительно невысокие требования к стабильности параметров лампы и элементов схемы, возможность получения сигналов с большой крутизной фронтов в схемах с маломощными и экономичными лампами.

Управление различными операциями в электронных вычислительных машинах основано на воздействии на электронные цепи определенных комбинаций или электрических импульсов, или уровней электрических напряжений (потенциалов), или, наконец, одновременно электрических импульсов и уровней напряжений. Поэтому в схемах вычислительных машин широко используются различные элементы для формирования и преобразования электрических импульсов (импульсные элементы) и уровней напряжения (потенциальные элементы), а также устройства для осуществления определенных законов управления или точнее логических законов[1]. Например, рассмотренный выше триггер (рис. 2-6) преобразует поступающие на его вход короткие отрицательные импульсы управления в определенные уровни напряжения на выходных зажимах схемы.

Во многих случаях уровни напряжений анодов триггеров подаются во внешнюю цепь через промежуточные элементы—катодные повторители. Катодный повторитель (рис. 2-9) представляет собой усилитель постоянного тока с нагрузкой, включенной в цепь катода. При подаче на сетку катодного повторителя положительного напряжения высокого уровня лампа повторителя открывается, ток через лампу увеличивается и увеличивается падение напряжения на сопротивлении в цепи катода лампы. Если на сетку подан низкий уровень напряжения, лампа запирается напряжением постоянного смещения, ток через нагрузочное сопротивление уменьшается и, следовательно, уменьшается падение напряжения на нем. Таким образом, высокому уровню входного напряжения соответствует высокий уровень выходного напряжения, и наоборот.

Катодный повторитель имеет высокое входное и низкое выходное сопротивления и поэтому используется как усилитель мощности для управляющих сигналов. Присоединение катодных повторителей к выходу триггера делает независимой работу последнего от параметров цепи нагрузки. Схема блока триггер — катодный повторитель, используемая в машине М-2, изображена на рис. 2-10 [Л. 4].

Усиление по мощности управляющих сигналов с одновременным изменением уровня напряжения сигнала достигается применением усилителя постоянного тока с анодной нагрузкой, называемого также инвертором (рис. 2-11). В этой схеме высокому уровню входного напряжения соответствует низкий уровень выходного напряжения, и наоборот.

В ряде случаев в схемах управления возникает необходимость в получении прямоугольных импульсов. Для этой цели используется схема одновибратора. Иногда эта схема называется «кипреле» (рис. 2-12).

На сетку правого триода через сопротивление R подано положительное напряжение. Поэтому при отсутствии сигналов на входе правая лампа все время открыта, а левая закрыта; на зажиме *Выход 1* — низкий уровень, а на зажиме *Выход 2* — высокий уровень напряжения.

При подаче на вход отрицательного импульса правая лампа закрывается, а левая открывается. Затем через время, определяющееся постоянной времени $\tau = RC$, потенциал на сетке правой лампы снова увеличится до такой величины, что правая лампа откроется, а левая закроется. С зажимов *Выход 1* и *Выход 2* снимаются прямоугольные импульсы, продолжительность которых может регулироваться изменением величин R и C.

Кипп-реле используются в цепях формирования импульсов и для осуществления цепей с задержкой импульса по времени.

Импульсы в цепях управления могут быть получены дифференцированием сигналов (выходных напряжений) триггеров и кипп-реле при помощи дифференцирующих цепочек RC. Переднему фронту сигнала будет соответствовать положительный импульс, а заднему—отрицательный (рис. 2-13). Дифференцирующая цепочка часто употребляется во входных цепях триггеров, кипп-реле и других устройств для улучшения формы импульса на входе (рис. 2-6).

Во всех ответственных случаях для формирования импульсов в цепях управления используются специальные формирующие электронные схемы. Форма и амплитуда импульсов на выходе формирующей схемы мало зависят от параметров входного импульса. Это очень важно для надежной работы цепей управления. Формирующая схема (рис. 2-14) состоит из дифференцирующей цепочки и из электронного генератора ударного возбуждения

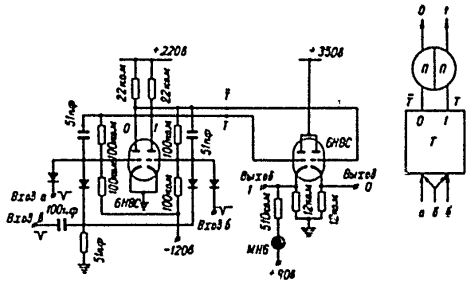


Рис. 2-10.

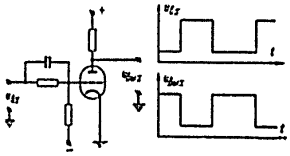


Рис. 2-11.

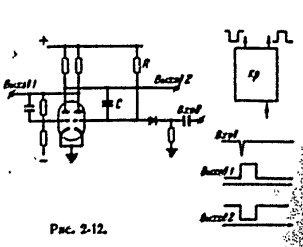


Рис. 2-12.

[1] Эти устройства рассматриваются в следующем параграфе.

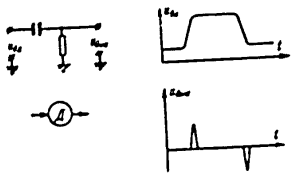


Рис 2 13

(блокинг-генератора) В анодную цепь лампы включен импульсный трансформатор, вторичная обмотка которого шунтирована диодом. При подаче на сетку положительного импульса лампа резко открывается и в колебательном контуре, образованном индуктивностью и распределенной емкостью импульсного трансформатора, возникают мощные колебания, которые определяются параметрами этого контура и мало зависят от параметров возбуждающего импульса Благодаря шунтирующему диоду во второй полупериод колебаний в контур вносится большое затухание и колебания в контуре срываются.

Отрицательный и положительный импульсы снимаются с зажимов соответственно первичной и вторичной обмоток трансформатора. Справа на рис 2-14 показано условное изображение формирующей схемы.

Повышение надежности работы вычислительных машин, уменьшение их размеров и потребляемой мощности достигаются применением в их схемах кристаллических триодов вместо электронных ламп.

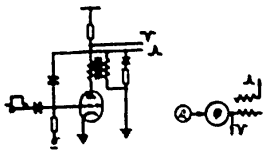


Рис. 2-14.

2-3. Схемы для выполнения элементарных логических операций

Цепи управления различными операциями вычислительной машины основаны на использовании в разнообразных сочетаниях небольшого числа типов схем, осуществляющих элементарные логические операции. В основном используются три типа схем а) разделения, б) совпадения, в) отрицания. Эти схемы выполняются на полупроводниковых вентилях и электронных лампах. Элементарные логические схемы на полупроводниковых вентилях (схемы совпадения и разделения) называются также дешифраторами. Логические схемы создают определенные управляющие сигналы (потенциальные или импульсные) при выполнении некоторых условий в отношении сигналов на входе этих схем.

Схемы разделения или смесительные схемы имеют один выход и несколько входов. На рис 2-15 показана схема разделения на полупроводниковых вентилях с двумя входами Примем, что единице на входе и выходе схемы соответствует высокий уровень, а нулю — низкий уровень напряжения. Если на всех входах низкий уровень напряжения, то на выходе схемы также низкий уровень напряжения. Если на любой из входов схемы или на несколько входов одновременно подан входной сигнал в виде высокого уровня напряжения, то возникнут ток и падение напряжения в сопротивлении и на выходе схемы появится высокий уровень напряжения

Схемы разделения работают по логическому закону *ИЛИ*. Сигнал, соответствующий единице, появляется на выходе, если на вход *а* или на вход *б* (или другие, если их больше двух) подается сигнал единицы. Вентили в рассматриваемой схеме разделяют цепи входных сигналов и препятствуют попаданию сигналов одной цепи в источник сигналов другой цепи.

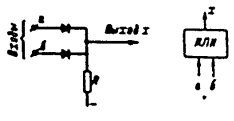


Рис. 2-15

Схемы разделения работают как с потенциальными, так и с импульсными входными сигналами.

Схемы отрицания имеют один вход и выход и дают на выходе сигнал единицы, если на вход поступает сигнал нуля, и наоборот. В качестве схемы отрицания может использоваться схема инвертора (рис. 2-11). При подаче на вход напряжения низкого уровня лампа заперта и ее анодное напряжение (выход схемы) имеет высокий уровень. Наоборот, при высоком уровне входного сеточного напряжения лампа открыта и анодное напряжение падает до своего низкого уровня.

Схемы отрицания работают по логическому закону *нет*. Если на вход подан сигнал *единица*, то на выходе имеем сигнал *не единица*, т. е. нуль. Если на вход подан сигнал *нуль*, то на выходе имеем *не нуль*, т. е. *единицу*.

Схемы отрицания изображаются на функциональных схемах прямоугольником с надписью *НЕТ*.

Схемы совпадения имеют один выход и несколько входов. На рис. 2-16 представлен один из вариантов схемы совпадения на полупроводниковых вентилях. Если на все входы подан низкий уровень напряжения $u_н$, то через сопротивление R и вентили будет протекать ток и напряжение на выходе будет равно напряжению $u_с$ за вычетом падения напряжения на сопротивлении R, что будет соответствовать низкому уровню напряжения. Если на некоторые, но не на все входы подать высокий уровень напряжения $u_в$, то соответствующие вентили окажутся запертыми и через них ток протекать не будет. Однако ток по-прежнему будет протекать через вентили, на которые поданы со стороны входов низкие уровни напряжений, и если сопротивления в цепях вентилей много меньше R, то по-прежнему из-за падения напряжения на сопротивлении R на выходе будет низкий уровень напряжения. Если на все входы подать напряжение высокого уровня $u_в$, то все вентили окажутся запертыми и напряжение на выходе возрастет до значения, соответствующего высокому уровню. Схема совпадения работает по логическому закону *И*. На выходе этой схемы имеем сигнал *единица* только в случае, если на все входы *а, б*, ... подан сигнал *единица*.

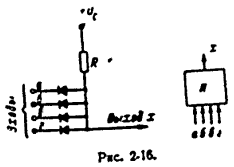


Рис. 2-16.

Схемы совпадения на полупроводниковых вентилях обычно используются в цепях с потенциальными входными сигналами.

При необходимости осуществления схемы совпадения, когда один входной сигнал задается уровнем напряжения, а другой — импульсом или когда оба входных сигнала — импульсы, часто употребляется электронная схема, называемая клапаном импульсов. Одна из употребляемых на практике схем клапана импульсов на пентоде изображена на рис. 2-17. Для управления лампой используются первая и третья сетки. На первую сетку подается уровень напряжения («управляющее напряжение») обычно с выхода логической схемы на полупроводниковых вентилях или с триггерной схемы, а на третью сетку — импульс («селекторный импульс»).

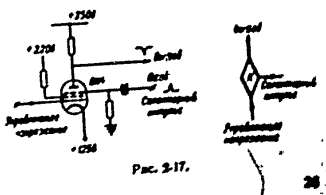


Рис. 2-17.

Параметры схемы и входных сигналов выбираются таким образом, чтобы лампа открывалась при положительном импульсе на третьей сетке лишь в том случае, если в это время к первой сетке приложено напряжение

высокого уровня. При этом с анода лампы снимается мощный отрицательный импульс. Подача высокого уровня напряжения на первую сетку в отсутствие импульса на третьей сетке, равно как и подача импульса на третью сетку при низких уровнях напряжения на первой сетке, не открывают лампу.

Клапан может работать как схема совпадения для положительных импульсов на первой и третьей сетках. Он употребляется так же, как схема формирования импульсов.

При необходимости получить на выходе положительный импульс применяется клапан с импульсным трансформатором в анодной цепи с шунтированием вторичной обмотки полупроводниковым вентилем аналогично схеме на рис. 2-14.

Схема несоответствия («дешифратор несоответствия») широко используется в некоторых вычислительных машинах (например, М-2 и М-3) в качестве типовой элементарной логической схемы

По существу схема несоответствия состоит из двух своеобразно соединенных между собой схем совпадения с тремя входами каждая. В каждой схеме совпадения на два входа поданы уровни напряжений с противоположных выходов триггеров, а на третий — общий для этих обеих схем сигнал управляющего напряжения (рис. 2-18).

Если напряжение высокого уровня подано на зажимы схемы, соединенные с вентилями B_1, B_2 и B_7 *, то эти вентили, а также B_3 оказываются запертыми. На выход подается напряжение высокого уровня. Если хотя бы на один из указанных вентилей подано напряжение низкого уровня, то через сопротивление R_1 будет протекать ток и из-за падения напряжения на R_1 снизится уровень выходного напряжения. Точно так же работает схема, образованная вентилями B_3, B_4, B_6, B_7 и сопротивлением R_2. Таким образом, на выходе схемы несоответствия (рис. 2-18) будет высокий уровень напряжения только в случае одновременного выполнения следующих условий: а) оба триггера, выходы которых присоединены к входам схемы несоответствия, находятся в различных состояниях; б) на дополнительный вход схемы подан высокий уровень управляющего напряжения.

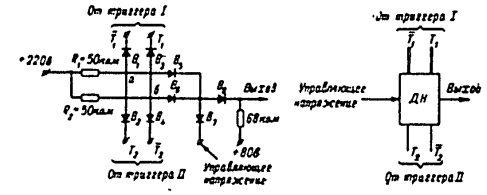


Рис. 2-18.

2-4. Выполнение некоторых операций при помощи логических схем

В настоящем параграфе рассмотрим выполнение при помощи простых логических схем следующих операций: перевод чисел из одной системы в другую, передача кода из одного триггерного регистра в другой, сдвиг кода.

На рис. 2-19 представлена диодная сетка для преобразования чисел из двоичной системы в восьмеричную. Схема состоит из восьми схем совпадения с тремя входами каждая. Входы этих схем подсоединены надлежащим образом к выходным зажимам трех триггеров, соответствующих трем разрядам двоичного числа. При помощи трех триггеров можно изобразить в двоичной форме любое число от нуля до семи. Напомним, что высокий уровень напряжения триггера будет на зажиме $0(\bar{T})$ или $1(T)$, если триггер находится соответственно в состоянии *0* или *1*. Легко убедиться, что в схеме на рис. 2-19 высокий уровень напряжения будет на шине, номер которой соответствует двоичному числу, установленному на триггерном регистре T_0—T_1—T_2.

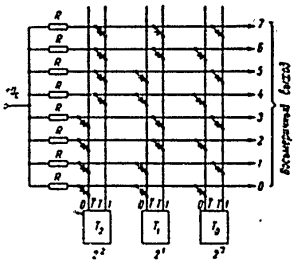


Рис. 2-19.

Схемы, аналогичные рис. 2-19, могут использоваться для выбора цепи в соответствии с кодом на входе схемы.

Рассмотрим схему для передачи кодов чисел из одного триггерного регистра в другой. На рис. 2-20,*а* представлена такая схема, используемая в машине М-3 (показана схема для одного разряда). Эта схема состоит из триггеров T_k^I и T_k^{II} k-того разряда регистров *I* и *II*, схемы совпадения *И* на полупроводниковых вентилях и клапана импульсов *К*.

При передаче кода числа из регистра *I* в регистр *II* надо в каждом разряде триггеры регистра *II* поставить в то состояние, в котором находятся триггеры регистра *I*. Предварительно от схемы управления на вход *а* триггеров регистра *II* подается отрицательный импульс *Гашение II*, который ставит все триггеры этого регистра в нулевое состояние.

Затем на вход схемы *И* подается вспомогательный сигнал «*I* в *II*» в виде прямоугольного широкого импульса, соответствующего высокому уровню напряжения. На второй вход схемы *И* поступает уровень напряжения с единичного выхода триггера T_k^I регистра *I*. Если этот триггер находится в состоянии *0*, то на схему *И* поступит высокий уровень напряжения и на вы-

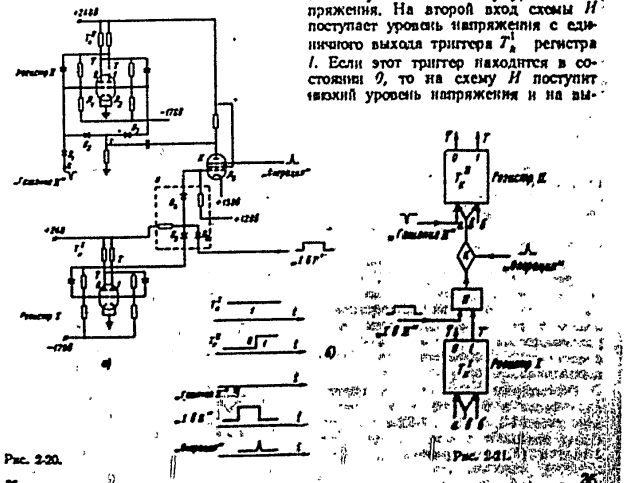


Рис. 2-20.

* Если первый триггер находится в состоянии *0*, а второй — в состоянии *1*, то напряжение высокого уровня приложено к вентилям B_1 и B_2.

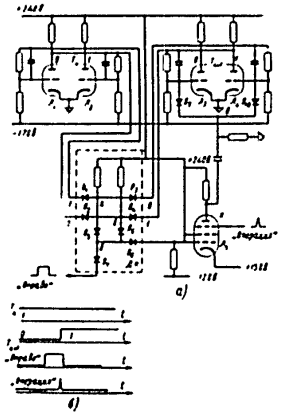


Рис. 2-22.

ходе схемы *И* (на первой сетке клапана *К*) будет низкий уровень напряжения. При подаче на третью сетку клапана импульса *Операция* лампа $Л_3^a$ не пропустит импульс, и триггер T_a^a останется в состоянии *0*.

Если триггер T_b^1 регистра *I* находится в состоянии *1*, высокий уровень напряжения поступает с его выхода на вход схемы *И* и далее на первую сетку клапана *К*. В этом случае при подаче на третью сетку клапана положительного импульса *Операция* на аноде клапана возникает отрицательный импульс, перебрасывающий триггер регистра *II* в состояние *1*, в котором находится триггер соответствующего разряда регистра *I*. На рис. 2-20,*б* показана временная диаграмма для рассмотренной схемы, а на рис. 2-21 представлена соответствующая функциональная схема.

Рассмотрим схему сдвига вправо кода числа в регистре, используемую в машине М-3. На рис. 2-22,*а* для простоты показаны только два триггера *k*-того и (*k*+1)-го разряда *n*, кроме того, опущены входные цепи триггеров, не участвующие в управлении рассматриваемой операцией.

При сдвиге вправо на один разряд триггер T_{k+1} должен быть установлен в состояние, в котором до этого находился триггер T_k. Состояния триггеров T_k и T_{k+1} сравниваются схемой несоответствия *ДН*. Пусть триггеры находятся в различных состояниях. В этом случае при подаче на управляющий вход схемы несоответствия вспомогательного положительного прямоугольного широкого сигнала *Вправо*, соответствующего высокому уровню напряжения, на первую сетку клапана *К* подается высокий уровень напряжения. При подаче на третью сетку лампы $Л_k$ положительного импульса *Операция* на аноде лампы возникает отрицательный импульс, меняющий состояние триггера T_{k+1} на противоположное, т. е. на такое, в котором находится триггер T_k.

Если триггеры T_k и T_{k+1} находятся в одинаковом состоянии, на первой сетке клапана *К* будет низкий уровень напряжения и подача импульса *Операция* не будет сопровождаться появлением импульса на входе триггера T_{k+1}. Состояние триггера T_{k+1} останется без изменений. На рис. 2-22,*б* показана временная диаграмма рассмотренной схемы, а на рис. 2-23 дана функциональная схема.

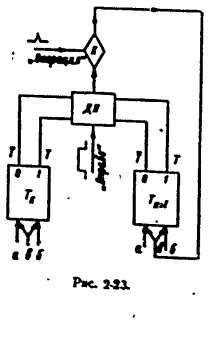


Рис. 2-23.

2-5. Особенности выполнения арифметических операций на вычислительной машине. Понятие о дополнительном и обратном кодах

В настоящем параграфе мы рассмотрим, каким образом выполнение всех арифметических операций в вычислительной машине может быть сведено к операциям сложения.

Как уже указывалось, знак плюс изображается в машине нулем, а знак минус — единицей в знаковом разряде. Положительное двоичное число с запятой, фиксированной перед старшим разрядом:

$$G^+ = +0,\ \gamma_1\gamma_2\ldots\gamma_n\ (\gamma_i = 1 \text{ или } 0) \quad (2\text{-}1)$$

в машине представляется в виде:

$$(G^+)_{пр} = 0,\ \gamma_1\gamma_2\ldots\gamma_n. \quad (2\text{-}2)$$

Аналогично отрицательное двоичное число

$$G^- = -0,\ \gamma_1\gamma_2\ldots\gamma_n \quad (2\text{-}3)$$

представляется в машине в виде:

$$(G^-)_{пр} = 1,\ \gamma_1\gamma_2\ldots\gamma_n. \quad (2\text{-}4)$$

Способ представления чисел (2-2) и (2-4) называется прямым кодом представления соответственно положительных и отрицательных чисел.

Операция вычитания сводится к операции простого арифметического сложения при помощи специальных кодов (обратного и дополнительного), используемых для представления отрицательных чисел в машине.

Чтобы представить двоичное отрицательное число (2-3) в обратном коде, надо поставить в знаковый разряд единицу, а во всех других разрядах заменить единицы нулями, а нули — единицами:

$$(G^-)_{об\,р} = 1,\ \sigma_1\sigma_2\ldots\sigma_n, \quad (2\text{-}5)$$

где $\sigma_i = 1$ при $\gamma_i = 0$ и $\sigma_i = 0$ при $\gamma_i = 1$.

При записи отрицательного числа в дополнительном коде ставят единицу в разряд знака, а цифровую часть числа заменяют дополнением модуля числа до целой единицы. Отрицательное число $G^- = -0,\ \gamma_1\gamma_2\ldots\gamma_n$ в дополнительном коде имеет вид:

$$(G^-)_{доп} = 1,\ \varepsilon_1\varepsilon_2\ldots\varepsilon_n, \quad (2\text{-}6)$$

где

$$0,\ \varepsilon_1\varepsilon_2\ldots\varepsilon_n = 1 - 0,\ \gamma_1\gamma_2\ldots\gamma_n. \quad (2\text{-}7)$$

Легко заметить, что

$$(G^-)_{доп} = (G^-)_{об\,р} + 2^{-n}, \quad (2\text{-}8)$$

где n — число разрядов в числе. Таким образом, дополнительный код может быть получен из обратного добавлением к нему единицы младшего разряда.

Установим связь между самим отрицательным числом G^- и числами $(G^-)_{об\,р}$ и $(G^-)_{доп}$, представляющими его обратный и дополнительный код.

Вычитая (2-3) из (2-5), имеем:

$$(G^-)_{об\,р} - G^- = 1,\ \sigma_1\sigma_2\ldots\sigma_n - (-0,\ \gamma_1\gamma_2\ldots\gamma_n) = 1,11\ldots1 = 2 - 2^{-n},$$

так как $\sigma_i + \gamma_i = 1$.

Следовательно,

$$(G^-)_{об\,р} = G^- + 2 - 2^{-n}. \quad (2\text{-}9)$$

Вычитая (2-3) из (2-6), имеем:

$$(G^-)_{доп} - G^- = 1,\ \varepsilon_1\varepsilon_2\ldots\varepsilon_n - (-0,\ \gamma_1\gamma_2\ldots\gamma_n).$$

Учитывая (2-7), получаем:

$$(G^-)_{доп} = G^- + 2. \quad (2\text{-}10)$$

Используя обратный или дополнительный код, можно операции вычитания и сложения чисел различных знаков свести к простому сложению кодов чисел.

Рассмотрим использование обратного кода при алгебраическом сложении двух чисел G и Q, когда одно из них или оба числа — отрицательные. Для этого случая может быть сформулировано следующее правило (предполагаем, что модуль алгебраической суммы меньше единицы).

При алгебраическом сложении двух двоичных чисел с использованием обратного кода положительные слагаемые представляются в прямом коде, а отрицательные — в обратном и производится арифметическое суммирование этих кодов, включая разряды

знаков, которые при этом рассматриваются как разряды целых единиц. При возникновении переноса из разряда знака единица переноса прибавляется к младшему разряду суммы кодов[1]. В результате получается алгебраическая сумма в прямом коде, если эта сумма положительна, и в обратном коде, если она отрицательна.

Это правило легко проверить. Пусть имеем двоичные n-разрядные числа $G^-<0$ и $Q^-<0$ (с фиксированной запятой). Их алгебраическая сумма $-1<G^-+Q^-<0$. В соответствии с (2·9) можно написать

$$(G^-)_{об.р}+(Q^-)_{об.р}=G^-+2-$$
$$-2^{-n}+Q^-+2-2^{-n}=$$
$$=[2]+(2-2^{-n})+(G^-+Q^-)-2^{-n}.$$

Двойка, стоящая в квадратных скобках, соответствует переносу из знакового разряда. В этом случае согласно сформулированному правилу единица переноса добавляется к младшему разряду суммы.

В результате получаем

$$(G^-)_{об.р}+(Q^-)_{об.р}=(G^-+Q^-)+$$
$$+(2-2^{-n})-2^{-n}+2^{-n}$$

или

$$(G^-)_{об.р}+(Q^-)_{об.р}=(G^-+Q^-)+$$
$$+2-2^{-n},$$

откуда согласно (2·9)

$$(G^-)_{об.р}+(Q^-)_{об.р}=(Q^-+Q^-)_{об.р}. \quad (2\text{·}11)$$

Итак, сложение с круговым переносом обратных кодов двух отрицательных чисел дает обратный код их алгебраической суммы.

Рассмотрим алгебраическую сумму двух чисел — положительного G^+ и отрицательного Q^-. Тогда

$$(G^+)_{пр}+(Q^-)_{об.р}=G^++Q^-+2-2^{-n}. \quad (2\text{·}12)$$

Если $(G^++Q^-)>0$, то $2+(G^++Q^-)>2$ и возникает перенос из разряда знака. Так как перенос круговой, то вместо двойки надо в рассматриваемое выражение добавить 2^{-n}.

[1] Такой перенос называется круговым.

В результате имеем.

$$(G^+)_{пр}+(Q^-)_{об.р}=(G^++Q^-)_{пр}$$
$$(\text{при } G^++Q^->0). \quad (2\text{·}13)$$

Если $(G^++Q^-)<0$, то $(2+G^++Q^-)<2$ и переноса не возникает. В этом случае из (2·12) получаем:

$$(G^+)_{пр}+(Q^-)_{об.р}=(G^++Q^-)_{об.р} \quad (2\text{·}14)$$
$$(\text{при } G^+-Q^-<0).$$

Итак, сложение кодов согласно приведенному правилу дает алгебраическую сумму в прямом коде, если эта сумма положительна, и в обратном коде, если она отрицательна.

Рассмотрим теперь использование дополнительного кода для алгебраического сложения. Приведем соответствующее правило (полагаем, что модуль алгебраической суммы меньше единицы).

При алгебраическом сложении двух двоичных чисел с использованием дополнительного кода положительные слагаемые представляются в прямом коде, а отрицательные — в дополнительном и производится арифметическое суммирование этих кодов, включая разряды знаков, которые при этом рассматриваются как разряды целых единиц. При возникновении переноса из разряда знака единица переноса отбрасывается. В результате получается алгебраическая сумма в прямом коде, если эта сумма положительна, и в дополнительном коде, если эта сумма отрицательна.

В самом деле, если $G^-<0$ и $Q^-<0$, то согласно (2·10)

$$(G^-)_{доп}+(Q^-)_{доп}=G^-+2+Q^-+2=$$
$$=2+(G^-+Q^-+2).$$

Так как $-1<G^-+Q^-<0$, то величина, стоящая в скобках, меньше двух, но больше единицы. Двойка, стоящая вне скобки, образует перенос из разряда знака, который будет отброшен. В результате получаем:

$$(G^-)_{доп}+(Q^-)_{доп}=(G^-+Q^-)+2=$$
$$=(G^-+Q^-)_{доп}. \quad (2\text{·}15)$$

Если $G^+>0$, а $Q^-<0$, то

$$(G^+)_{пр}+(Q^-)_{доп}=(G^++Q^-)+2.$$

Если при этом $0<G^++Q^-<1$, то стоящая вне скобки двойка дает единицу переноса из разряда знака, которая отбрасывается, после чего получаем:

$$(G^+)_{пр}+(Q^-)_{доп}=(G^++Q^-)_{пр} \quad (2\text{·}16)$$
$$(\text{при } G^++Q^->0).$$

Если же $-1\leq G^++Q^-<0$, то $(Q^++Q^-)+2<2$ и переноса из разряда знака не возникает. В этом случае

$$(G^+)_{пр}+(Q^-)_{доп}=(G^++Q^-)_{доп} \quad (2\text{·}17)$$
$$(\text{при } G^++Q^-<0).$$

Таким образом, мы показали, что операция вычитания, рассматриваемая как алгебраическое сложение, может быть заменена операцией преобразования прямых кодов отрицательных чисел в обратные или дополнительные и простым арифметическим сложением соответствующих кодов чисел. Если результат получается в обратном или дополнительном коде (алгебраическая сумма отрицательна), то в случае необходимости он может быть переведен в прямой код.

При умножении (или делении) двоичных чисел знак произведения (или частного) получается как сумма (без учета переноса) кодов знаков множимого и множителя (или делимого и делителя) (табл. 2-1).

При умножении двоичных чисел с фиксированной запятой сомножители и произведение представляются в прямом коде.

Само умножение в вычислительной машине обычно сводится к последовательности операций прибавления множимого к сумме уже ранее вычисленных частных произведений и сдвига получаемых сумм вправо на один разряд. Код множителя разряд за разрядом (обычно начиная с младших) подается на элемент схемы, проверяющей наличие в поданном разряде *1* или *0*. Если в данном разряде стоит *1*, производится прибавление кода множимого к сумме, стоящей в сумматоре. Затем производится сдвиг новой суммы вправо на один разряд. Если в данном разряде множителя стоит *0*, прибавление множимого не производится, а лишь выполняется сдвиг кода сумматора вправо на один разряд.

Сказанное иллюстрируем следующим примером:

1,101001 × 0,010011	Множимое (−0,101001) Множитель (+0,010011)
+ 1. / 0. / 1.	Определение кода знака произведения
000000	Установка нуля в сумматоре.
+ 101001 / 101001	Прибавление множимого к коду сумматора (в шестом разряде множителя стоит 1)
010100	Сдвиг кода сумматора вправо на один разряд.
+ 101001 / 111101	Прибавление множимого к коду сумматора (в пятом разряде множителя стоит 1)
011110	Сдвиг кода сумматора вправо на один разряд. Прибавление множимого не производится (в четвертом разряде множителя стоит 0)
001111	Сдвиг кода сумматора вправо на один разряд. Прибавление множимого не производится (в третьем разряде множителя стоит 0)
000111	Сдвиг кода сумматора вправо на один разряд.
+ 101001 / 110000	Прибавление множимого к коду сумматора (во втором разряде множителя стоит 1)
011000	Сдвиг кода сумматора вправо на один разряд. Прибавление множимого не производится (в первом разряде множителя стоит 0)
001100	Сдвиг кода сумматора вправо на один разряд
1,001100	Произведение (−0,001100)

Таблица 2-1

Операции со знаками	(+)×(+)=(+) (+):(+)=(+)	(+)×(−)=(−) (+):(−)=(−)	(−)×(+)=(−) (−):(+)=(−)	(−)×(−)=(+) (−):(−)=(+)
Операции с кодами знаков	0+0=0	0+1=1	1+0=1	1+1=1*

* 1+1=0+единица переноса в старший разряд, которая при сложении кодов знаков отбрасывается.

В машинах с фиксированной запятой деление возможно лишь в случае, когда делитель больше делимого, так как в противном случае частное больше единицы и произойдет переполнение разрядов. Деление двоичных чисел с фиксированной запятой на вычислительной машине сводится к чередованию операций вычитания делителя из делимого или остатка от предыдущего вычитания и сдвига остатка влево на один разряд. При этом частное получается последовательно разряд за разрядом, начиная со старшего разряда.

Существует ряд способов выполнения деления на вычислительной машине. Один из этих способов состоит в следующем. Если остаток получается положительным, то в очередной разряд частного ставится единица. Затем остаток сдвигается влево на один разряд и снова производится вычитание делителя. Если остаток получается отрицательным, то в соответствующий разряд частного ставится нуль. К отрицательному остатку добавляется делитель, т. е. восстанавливается последний положительный остаток. Затем восстановленный остаток сдвигается влево на один разряд и снова вычитается делитель.

При вычитании делителя к коду остатка прибавляется дополнительный или обратный код делителя.

Рассмотрим пример. Будем пользоваться для операции вычитания дополнительным кодом (см. таблицу в правом столбце).

В результате получен прямой код частного 1,011111, что соответствует числу — 0,011111.

В машинах с плавающей запятой выполнение арифметических операций усложняется, так как, кроме собственно арифметических операций с мантиссами, которые выполняются так же, как и в машинах с фиксированной запятой, должны быть выполнены соответствующие операции с порядками и некоторые вспомогательные операции — выравнивание порядков со сдвигом мантиссы вправо и нормализация чисел (см. § 1-5).

При сложении и вычитании чисел с плавающей запятой машина должна произвести операцию выравнивания порядков чисел, сдвигая вправо мантиссу числа с меньшим порядком на число разрядов, равное разности порядков этих чисел. Затем мантиссы складываются или вычитаются, а результат нормализуется. При умножении (делении) порядки чисел складываются (вычитаются), а мантиссы перемножаются (делятся). Затем результат нормализуется. При алгебраическом сложении порядков отрицательные порядки представляются дополнительным или обратным кодом.

		Разряд частного
1,010110 0,101101	Делимое (—0,010110) Делитель (+0,101101)	
1. + 0. 1.	Определение кода знака частного	
0,101100 + 1,010011	Сдвиг делимого влево на один разряд Делитель (дополнительный код)	
1,111111 + 0,101101	Остаток (дополнительный код). Остаток <0 Делитель (прямой код)	0
0,101100 1,011000 + 1,010011	Восстановленный остаток Сдвиг остатка влево на один разряд (прямой код). Целая единица запасена в разряде знака Делитель (дополнительный код)	
0,101011 1,010110 + 1,010011	Остаток (прямой код). Остаток >0 Сдвиг остатка влево на один разряд (прямой код) Делитель (дополнительный код)	1
0,101001 1,010010 + 1,010011	Остаток (прямой код). Остаток >0 Сдвиг остатка влево на один разряд (прямой код) Делитель (дополнительный код)	1
0,100101 1,001010 + 1,010011	Остаток (прямой код) Остаток >0 Сдвиг остатка влево на один разряд (прямой код) Делитель (дополнительный код)	1
0,011101 0,111010 + 1,010011	Остаток (прямой код). Остаток >0 Сдвиг остатка влево на один разряд (прямой код) Делитель (дополнительный код)	1
0,001101	Остаток (прямой код). Остаток >0	1

В вычислительных машинах отрицательные числа (мантиссы) хранятся в запоминающем устройстве большей частью в прямом коде и в такой форме используются при операциях умножения и частично деления. При алгебраическом сложении и вычитании отрицательные числа предварительно переводятся в дополнительный или обратный коды. Отрицательный результат этих операций также получается в дополнительном или обратном кодах и перед помещением в запоминающее устройство переводится в прямой код.

Отрицательные порядки чисел проще хранить в запоминающем устройстве в дополнительном или обратном кодах, так как с порядками совершаются лишь операции сложения и вычитания.

2-6. Арифметические устройства

Арифметическим устройством называется узел вычислительной машины, в котором производятся арифметические и некоторые другие операции над числами.

В машине с плавающей запятой по существу имеется два арифметических устройства — одно для операций над мантиссами, а другое — над порядками.

В предыдущем параграфе было показано, как при помощи специальных кодов (дополнительного и обратного) все арифметические операции могут быть сведены к сложению. Поэтому основным элементом арифметического узла является сумматор кодов чисел, а в случае машин с плавающей запятой также и сумматор кодов порядков.

Арифметическое устройство содержит триггерные регистры в качестве узлов для временного хранения кодов чисел, над которым совершается операция. В ряде случаев эти триггерные регистры одновременно являются частью самих суммирующих устройств.

Перечислим операции, обычно выполняемые арифметическим устройством: 1) установка нуля на регистрах арифметического узла; 2) прием кодов чисел и команд из запоминающего устройства на регистр арифметического узла; 3) взятие дополнительного (или обратного) кода; 4) сдвиг кода числа вправо и влево; 5) сложение кодов чисел; 6) передача кодов чисел в запоминающее устройство.

Арифметические устройства могут выполнять также и некоторые другие операции. Например, в машине М-3 регистр арифметического узла используется для передачи кодов чисел и команд в запоминающее устройство при вводе программы в машину, для переноса кодов из одной ячейки запоминающего устройства в другую, для передачи чисел из запоминающего устройства на выводную печатающую машину.

В малых машинах последовательного действия арифметическое устройство выполняет лишь часть из указанных выше операций.

Перейдем к рассмотрению принципа работы основного элемента арифметического устройства — сумматора.

При сложении двух чисел независимо от системы счисления в каждом разряде производится сложение трех цифр: цифры данного разряда первого слагаемого, цифры данного разряда второго слагаемого и цифры (единица или нуль) переноса из соседнего младшего разряда. В результате операции сложения для одного разряда получаются цифра этого разряда в сумме и цифра (нуль или единица) переноса в следующий старший разряд.

Для рассмотрения сложения двух двоичных чисел достаточно проследить операцию сложения для одного какого-либо разряда этих чисел. В зависимости от значений складываемых цифр и наличия или отсутствия единицы переноса из предыдущего младшего разряда результат сложения (сумма и перенос в старший разряд) будет различным. Возможны восемь вариантов, приведенных в табл. 2-2.

Первые четыре варианта соответствуют сложению разрядов двух двоичных чисел при отсутствии переноса единицы из младшего разряда.

Легко заметить, что схема, приведенная на рис. 2-24, обеспечивает вы-

Таблица 2-2

Перенос из предыдущего младшего разряда	Первое слагаемое	Второе слагаемое	Сумма	Перенос в следующий старший разряд
0	0	0	0	0
0	1	0	1	0
0	0	1	1	0
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1
1	0	1	0	1
1	1	1	1	1

полнение операции сложения для первых четырех вариантов табл. 2-2.

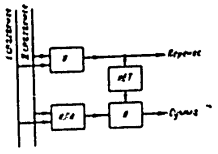


Рис. 2-24.

Например, в варианте, соответствующем четвертой строке таблицы, первое и второе слагаемые равны единице и, следовательно, на соответствующие шины одновременно поступают положительные импульсы (импульсы единицы). В этом случае на выходе подключенной к этим шинам схемы *И* тоже появится положительный импульс, что указывает на наличие единицы переноса в старший разряд. Выход схемы *И* подан на вход схемы *НЕТ*, которая при подаче на ее вход положительного импульса создает на выходе отрицательный импульс. На входы второй схемы *И* поступит от схемы *ИЛИ* положительный импульс, а от схемы *НЕТ* — отрицательный. Поэтому на шину суммы импульс подан не будет. Приведенная на рис. 2-24 схема называется одноразрядным «полусумматором», так как она реализует сложение лишь двух цифр из трех (третья цифра переноса), участвующих в разрядном суммировании.

Полное разрядное суммирование с учетом цифры переноса может быть выполнено за два такта схемой, образованной из двух полусумматоров и схемы разделения типа *ИЛИ* (рис. 2-25). Необходимо подчеркнуть, что эта схема работает в два такта. В первом такте на вход схемы подаются импульсы, соответствующие двоичным цифрам в суммируемых разрядах первого и второго чисел, и первый полусумматор вырабатывает сигналы суммы и переноса по правилам первых четырех строк табл. 2-2 *. Затем во втором такте второй полусумматор суммирует цифры начальной суммы и переноса из предыдущего младшего разряда.

Так как одновременно в схеме на рис. 2-25 складываются только две цифры, то она носит название сумматора с двумя входами. При помощи логических схем можно осуществить одноразрядный сумматор, в котором все три цифры (первого и второго слагаемых и переноса) складываются одновременно. Такое устройство называется сумматором с тремя входами. Его функциональная схема изображена на рис. 2-26. Схема расшифровывает и выдает на выходе нужные сигналы суммы и переноса для любой из восьми комбинаций входных сигналов, указанных в табл. 2-2.

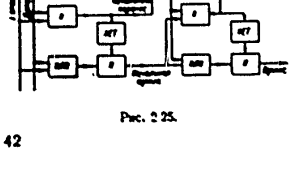


Рис. 2-25.

Если, например, на все три входные шины поданы сигналы единицы (соответствует восьмой строчке табл. 2-2), то на выходе схем совпадения *1*, *2*, *3*, *4* и схемы разделения *6* будут также сигналы единицы. На выходе схемы разделения *7* появится сигнал единицы, т. е. появится сигнал переноса единицы в старший разряд. На входе схемы *НЕТ* будет сигнал единицы, а на ее выходе — сигнал нуля. Поэтому на выходе схемы совпадения *5* также будет сигнал нуля. Однако на вход схемы разделения *9* с выхода схемы совпадения *1* поступит сигнал единицы, который пройдет на выход и на зажиме *Сумма* будет сигнал единицы. Таким же образом можно проследить работу схемы при других комбинациях входных сигналов.

До сих пор рассматривались одноразрядные сумматоры и сложение в одном разряде. Теперь будет рассмотрено суммирование кодов чисел, содержащих некоторое число разрядов.

Различают сумматоры последовательного и параллельного действия.

В сумматорах последовательного действия операции суммирования выполняется поочередно для каждого разряда в отдельности, начиная с младшего. Такое суммирующее устройство может использовать всего один одноразрядный сумматор, на входы которого через определенные промежутки времени поочередно подаются импульсы, соответствующие двоичным цифрам складываемых разрядов. Можно сказать, что в последовательном сумматоре процесс сложения развертывается во времени.

* Эта сумма и перенос являются не окончательными (их называют начальными), так как они получены без учета цифры переноса из предыдущего младшего разряда. Заметим, что начальные сумма и перенос одновременно не могут быть равны единице.

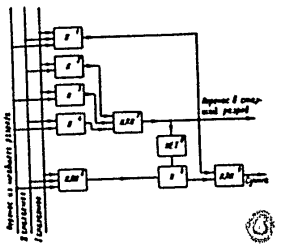


Рис. 2-26.

В сумматоре параллельного действия сложение производится для всех разрядов чисел одновременно. Естественно, что при этом необходимо иметь столько решающих и управляющих цепей, сколько разрядов содержат суммируемые числа. В параллельном сумматоре процесс сложения развертывается по множеству решающих и управляющих цепей, иначе говоря в пространстве. Надо при этом иметь в виду, что даже в параллельных сумматорах обычно не удается полностью избежать последовательного развертывания во времени переноса единицы из младших разрядов в старшие.

Время, необходимое для выполнения сложения, в параллельном сумматоре значительно меньше, чем в последовательном. Поэтому всегда, когда необходимо получить большую скорость вычислений, применяют параллельные сумматоры, хотя это связано с увеличением объема аппаратуры.

Действие последовательного сумматора поясняется схемой на рис. 2-27 [Л. 13]. Рабочие такты устройству задаются генератором синхронизирующих импульсов.

На одноразрядный сумматор с тремя входами через клапаны K_1 и K_2, управляемые синхронизирующими (тактирующими) импульсами q, через определенные промежутки времени последовательно друг за другом подаются разряды (соответствующие импульсы) складываемых чисел. Кроме того, на третий вход схемы подается сигнал переноса (нуля или единицы) из предыдущего младшего разряда. Этот сигнал был выработан на предыдущем такте работы схемы, когда производилось суммирование для предыдущего младшего разряда, а затем соответствующий импульс переноса сохранен (т. е. задержан) при помощи цепи задержки (*3* — на рис. 2-27) и выдан

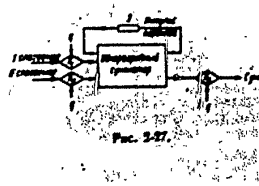


Рис. 2-27.

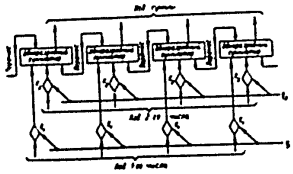


Рис. 2-28.

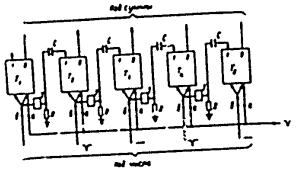


Рис. 2-29.

в нужный момент на вход схемы. Сигнал суммы выдается на внешнюю цепь клапаном K_3, управляемым также синхронизирующими импульсами.

Схема рис. 2-28 поясняет принцип действия параллельного сумматора, состоящего из одноразрядных сумматоров с тремя входами. На входы каждого одноразрядного сумматора поступают коды цифр соответствующих разрядов первого и второго чисел и сигнал переноса из предыдущего младшего разряда.

Коды первого и второго чисел подаются на сумматор через группы клапанов K_1 и K_2, открываемых импульсами q_1 и q_2. В каждом разряде суммирующий элемент имеет два выхода, с которых снимаются сигнал, определяющий цифру (*1* или *0*) соответствующего разряда суммы, и сигнал переноса в следующий старший разряд.

В рассматриваемой схеме время, необходимое для выполнения операции сложения, зависит от времени установления процесса в самом одноразрядном сумматоре и от числа разрядов. Действительно, в случае, когда перенос распространяется от самого младшего разряда к самому старшему, время операции сложения будет определяться суммой времен установления процессов во всех разрядах сумматора.

До сих пор нами рассматривались последовательные и параллельные сумматоры, выходные сигналы которых определялись комбинацией одновременно поданных на вход кодов обоих складываемых чисел. Такие устройства называются сумматорами комбинационного типа. Они основаны на использовании логических схем.

Наряду с комбинационными в вычислительных машинах широко применяются сумматоры накапливающего типа. В этих устройствах коды слагаемых поступают на сумматор отдельно в разные моменты времени.

При подаче на вход нового числа образуется сумма этого числа с числом, ранее стоявшим в сумматоре. Накапливающий сумматор может поочередно суммировать любое количество поступающих на его вход чисел. Полученная сумма сохраняется на сумматоре и после снятия сигналов кода слагаемого.

В накапливающих сумматорах используются триггерные счетные ячейки. Принцип действия накапливающего параллельного сумматора поясняется схемой на рис. 2-29. Схема состоит из пяти триггеров T_1, T_2, T_3, T_4 и T_5. На счетные входы (входы *в*) триггеров подается параллельный импульсный код числа (отрицательный импульс — единица, отсутствие импульса — нуль). Каждый следующий отрицательный импульс, попадая на счетный вход триггера, перебрасывает его в противоположное состояние. В сумматоре подача каждого следующего отрицательного импульса на вход какого-нибудь триггера означает увеличение суммы на единицу соответствующего разряда.

Пусть триггер какого-нибудь разряда находится в состоянии *1*. Если на его вход поступает отрицательный импульс, т. е. добавляется единица в этом разряде, то триггер должен перейти в состояние *0*, но, кроме того, должен появиться импульс переноса единицы в соседний старший разряд. Таким образом, всякий раз, когда состояние какого-либо триггера сумматора меняется с *1* на *0*, на входе соседнего старшего разряда должен появляться дополнительный отрицательный импульс (импульс переноса). Переход триггеров из состояния *1* в *0* контролируется дифференцирующей цепочкой *RC*, подключенной к единичным выходам триггеров.

При переходе триггера из состояния *1* в состояние *0* напряжение на его единичном выходе падает и дифференцирующая цепочка вырабатывает отрицательный импульс (импульс переноса), который поступает на счетный вход триггера соседнего старшего разряда, что соответствует добавлению в этот разряд единицы. Отрицательный импульс переноса перебрасывает триггер в противоположное состояние. При переходе триггера из состояния *0* в состояние *1* напряжение на его единичном выходе возрастает, и на выходе дифференцирующей цепочки появляется положительный импульс, который не изменяет состояние соседнего триггера. Иначе говоря, в этом случае импульс переноса не возникает.

Необходимо, чтобы импульсы переноса поступали на вход триггеров после окончания в них переходного процесса, вызванного импульсами подаваемого на вход сумматора кода числа. В противном случае импульс переноса, попадая на вход триггера в момент, когда еще не закончился переходный процесс от импульса кода числа, не окажет влияния на принимаемое триггером состояние.

Чтобы избежать «потери» импульса переноса, включают цепи задержки *З* (рис. 2-29), задерживающие импульсы переноса на время, необходимое для окончания переходного процесса в триггере.

Подачей отрицательного импульса на нулевые входы (входы *а*) всех триггеров сумматор ставится в состояние нуля. Если теперь на счетные входы триггеров подать импульсный код числа, то сумматор прибавит к нулю это число.

Например, при подаче на вход кода числа 01010 триггеры T_1, T_3 и T_5 останутся в состоянии нуля, а триггеры T_2 и T_4 перейдут в состояние единицы, так как на их счетный вход будут поданы отрицательные импульсы. Теперь состояние триггеров сумматора будет изображать число 01010.

Если затем подать на вход сумматора код числа 01011, то это число прибавится к числу 01010, т. е. получится сумма

$$\begin{array}{r} 1\ 1 \\ 01010 \\ + \\ 01011 \\ \hline 10101 \end{array}$$

Действительно, отрицательные импульсы в коде числа 01011 перебросят триггеры T_2, T_4 и T_5 в противоположное состояние. При этом триггер T_5 окажется в состоянии *1*. Триггеры T_2 и T_4 перейдут в состояние *0*, но при этом возникнут отрицательные импульсы переноса, которые поставят в состояние *1* триггеры T_1 и T_3. В результате на сумматоре будет стоять сумма 10101.

Если импульс переноса поступает на вход триггера, стоящего в состоянии *1*, то последний переходит в состояние *0* и при этом возникает новый импульс переноса в следующий старший разряд. При определенных условиях имеет место «пробег» единицы переноса через весь триггерный регистр сумматора от самого младшего к самому старшему разряду. Легко заметить, что время, необходимое для выполнения операции сложения, будет определяться, если пренебречь временем переходных процессов в триггерах, суммой времени задержек импульсов переноса. Если в каждой цепи задержки импульс переноса задерживается на время $t_з$, то в худшем случае (при пробеге единицы переноса через весь сумматор) продолжительность операции сложения будет:

$$T_{сл} = t_з(n-1),$$

где n — число разрядов сумматора.

Для уменьшения времени выполнения операции сложения отказываются от последовательной во времени передачи переноса из одного разряда в другой и применяют схемы, в которых перенос производится во все разряды одновременно [Л. 17]. Это достигается разделением собственно процесса сложения на два такта: 1) сумматор выполняет поразрядное сложение (без учета переносов) и вырабатывает сигналы переносов; 2) к стоящей на

сумматоре поразрядной сумме одновременно добавляются переносы и образуется окончательная сумма

Поясним это на конкретном примере. Сложим два двоичных числа

10011011(0001)1101	I слагаемое
00100100(0111)0001	II слагаемое
10111111(1110)1110	Поразрядная сумма
00000000(0001)0010	Перенос
10111111(0000)1110	Окончательная сумма

Рассматривая образование окончательной суммы, замечаем, что прибавление единицы переноса к разряду поразрядной суммы вызывает распространение переноса влево от разряда, где он возник, до первого слева разряда с цифрой нуль. При этом во всех этих разрядах, включая первый слева нулевой разряд, цифра меняется на обратную

После того как в первом такте сложения получены поразрядная сумма и импульсы переноса, можно при помощи логических схем выделить группы разрядов, в которых должен быть изменен цифровой код на обратный, и одновременно подать на счетные входы триггеров этих разрядов импульс переноса

На рис 2-30 представлена схема, поясняющая принципы работы накапливающего параллельного сумматора с одновременным переносом во всех разрядах При подаче синхронизирующего импульса q_1 на клапаны K_1 импульсный код числа поступает на счетный вход триггеров сумматора При этом в каждой триггерной ячейке происходит поразрядное суммирование кода стоящего на сумматоре, с кодом введенным кодом Одновременно при помощи дифференцирующих цепочек $Д$ образуются в соответствующих разрядах импульсы переноса. Эти импульсы подаются на клапаны K_2. Так как для открытия клапанов необходимы положительные сигналы, то на нашей схеме дифференцирующие ячейки подключены к нулевым выходам триггеров и импульсу переноса соответствует положительный импульс на входе клапана K_2. Через некоторый промежуток времени, достаточный для окончания переходных процессов, вызванных импульсами кода числа, подается синхронизирующий импульс q_2, открывающий те из клапанов K_2, на другую сетку которых поданы сравнительно широкие импульсы от дифференцирующих цепочек. Через открывшиеся клапаны K_2 импульсы переноса попадают на вход триггера соседнего старшего разряда и, кроме того, проходят справа налево по цепочке клапанов AA' и одновременно попадают на подсоединенные к этой цепочке входы соответствующих триггерных ячеек, перебрасывая их в противоположное состояние. Прохождение импульсов переноса вдоль цепи AA' контролируется клапанами K_3. Каждый из этих клапанов пропускает импульс переноса влево, если соответствующий триггер стоит в состоянии *1*, и не пропускает, если триггер находится в состоянии *0*. Для этого уровни напряжения с единичных выходов триггеров подаются на управляющие входы клапанов K_3. Таким образом, клапаны K_2 и K_3 образуют логические схемы, выделяющие группы разрядов, в которых коды должны быть изменены на обратные импульсами переноса. Код суммы выдается с сумматора через клапаны K_5, открываемые синхронизирующим импульсом q_3. При опрокидывании триггерных ячеек под действием импульсов переноса в дифференцирующих ячейках возникнут новые импульсы переноса, которые уже не должны оказывать влияния на состояние схемы. Это обеспечивается тем, что клапаны K_3 не пропускают новых импульсов переноса.

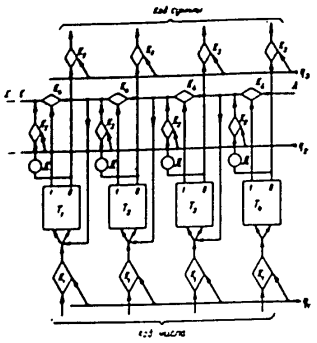


Рис 2-30

В рассматриваемой схеме время «пробега единицы переноса» через регистр сумматора определяется в основном временем переходных процессов в цепочке последовательно работающих клапанов K_3.

Комбинационный сумматор в сочетании с триггерным регистром, «запоминающим» коды чисел, обладает свойством накопителя, т. е. может прибавлять новые слагаемые к ранее полученной сумме и хранить получаемый результат. В качестве примера рассмотрим комбинационный параллельный сумматор с триггерным регистром, используемый в машине М-3 [Л. 3 и 4].

В этой схеме, так же как и в предыдущем сумматоре, процесс сложения производится за два такта. Однако содержание этих тактов иное. В первом такте определяются единицы переноса, которые сохраняются на специальном триггерном регистре — регистре запоминания единиц переноса. Во втором такте выполняется поразрядное суммирование (без образования переносов) цифр первого и второго слагаемых и цифры переноса, полученной в первом такте.

Рассмотрим сложение двух двоичных чисел:

01011011	Первое слагаемое
+ 00101011	Второе слагаемое
10000110	Сумма

Эту операцию можно разбить на две части в соответствии с указанными выше тактами работы сумматора.

Первый такт. Определяем единицы переноса:

01011011	Первое слагаемое
00101011	Второе слагаемое
11110110	Перенос

причем в отличие от численного примера, рассмотренного выше в связи с рис. 2-29, здесь в строчке „перенос" показаны не только первичные единицы переноса, появляющиеся, когда в одноименных разрядах первого и второго слагаемых стоят единицы, а также все единицы переноса, возникающие при сложении рассматриваемых двух чисел.

Второй такт. Поразрядное сложение (без образования новых переносов) обоих слагаемых и единиц переноса:

01011011	Первое слагаемое
11110110	Перенос
00101011	Второе слагаемое
10000110	Сумма

Рассматривая последние две таблицы, можно сформулировать следующие логические правила образования переноса и суммы.

1 В данном разряде появляется единица переноса, если в предыдущем разряде первого и второго слагаемых и переноса имеются хотя бы две единицы.

2. Для образования суммы нужно двоичную цифру в каждом разряде первого слагаемого изменить на обратную, если в этом разряде во втором слагаемом и переносе стоят разные цифры, и оставить без изменения, если эти цифры одинаковые

Первое правило реализуется в первом такте работы сумматора, а второе — во втором. Операции в обоих тактах совершаются при помощи логических схем.

В рассматриваемом сумматоре суммирующее устройство совмещено с триггерными регистрами запоминания первого и второго слагаемых [1], причем триггерный регистр первого слагаемого одновременно является и накопителем [2]. Кроме того, имеется специальный триггерный регистр для запоминания единиц переноса.

Схема, осуществляющая управление регистром запоминания единиц переноса в соответствии с правилом *1*, приведена на рис. 2-31. На схеме пока-

[1] Триггерные регистры запоминания чисел необходимы и при накапливающем сумматоре типа рис. 2-30, но там эти регистры в меньшей степени функционально связаны с операцией сложения.

[2] В коде команд машины М-3 (см. приложение 1) обозначаемые далее на рис. 2-31 и 2-32 регистр первого слагаемого и суммы и регистр второго слагаемого называются соответственно регистром второго числа и суммы и регистром первого числа.

ваны триггеры и управляющие лампы для n-ного и младшего $n+1$-го разрядов. Образование единиц переноса осуществляется при помощи простых схем состоящих из схем совпадения и разделения и специальных ламповых

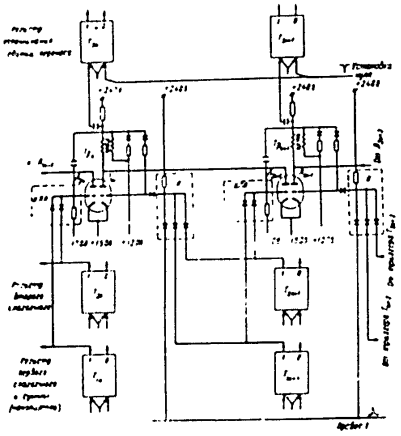


Рис 2-31

схем формирования импульсов. В каждом разряде для этой цели используется двойной триод. В аноде правой лампы (лампы $Л_1$) включен импульсный трансформатор. Отрицательный импульс с анодной цепи лампы $Л_1$ подается на единичный вход триггера соответствующего разряда регистра запоминания единиц переноса. Положительный импульс со вторичной обмотки трансформатора поступает на сетку левой лампы (лампы $Л_2$). Левая лампа $Л_2$ младшего $n+1$-го разряда подключена параллельно с лампой $Л_1$ n-ного разряда на одну и ту же анодную нагрузку. Схемы совпадения (с тремя входами) включены в цепях сеток лампы $Л_1$, а схемы разделения — в цепях лампы $Л_2$. Кроме того, как мы увидим ниже, сами лампы $Л_2$ работают тоже, как схемы совпадения.

Предварительно отрицательным импульсом регистр запоминания единиц переноса ставится в состояние нуля. Образование единиц переноса в регистре переноса начинается после подачи положительного импульса *Пробег 1*. Этот сигнал подается одновременно на соответствующие входы схем совпадения во всех разрядах сумматора.

Пусть, например, триггеры T_1 и T_2 $n+1$-го разряда первого и второго слагаемых стоят в состоянии единицы и нет переноса из $n+2$-го разряда. Тогда на выходе схемы $И$, включенной в цепь сетки лампы $Л_{1n}$, появится высокое напряжение. Лампа $Л_{1n}$ откроется и образующийся при этом на ее аноде отрицательный импульс поставит в состояние единицы триггер T_{3n} регистра запоминания единиц переноса. Одновременно на сетку лампы $Л_{2n}$ со вторичной обмотки трансформатора $Тр_n$ поступает положительный импульс и, если хотя бы один из триггеров T_{1n} или T_{2n} регистров первого и второго числа находится в состоянии единицы, то потенциал сетки лампы $Л_{2n}$ оказывается достаточным, чтобы под действием этого положительного импульса лампа $Л_{2n}$ открылась. Лампа $Л_{2n}$ включена параллельно с лампой $Л_{1n-1}$ на общую анодную нагрузку (цепь трансформатора $Тр_{n-1}$) и возникающий в этой цепи отрицательный импульс поставит в положение единицы триггер ближайшего старшего $n-1$-го разряда регистра запоминания единиц переноса (триггер T_{3n-1}). Процесс переноса, возникший в $n-1$-м разряде, указанным путем будет распространяться в сторону старших разрядов, пока не достигнет такого разряда, в котором оба триггера T_{1i} и T_{2i} стоят в положении нуля.

Действительно, если оба триггера T_{1i} и T_{2i} какого-нибудь разряда находятся в состоянии нуля и в анодной цепи лампы $Л_{1i}$ возникает импульс переноса, устанавливающий триггер T_{3i} в положение единицы, то возникающий при этом положительный импульс не сможет пройти через лампу $Л_{2i}$, так как с обоих триггеров T_{1i} и T_{2i} поданы низкие уровни напряжения. Процесс распространения единицы переноса в рассматриваемой группе разрядов прекратится.

Важно отметить, что образование единиц переноса может начинаться сразу в нескольких разрядах и одновременно распространяться справа налево. Время установления единиц переноса в триггерном регистре T_3 определяется продолжительностью переходных процессов в цепях ламп $Л_1$ и $Л_2$. В наиболее неблагоприятном режиме, когда единица переноса „пробегает" от самого младшего к самому старшему разряду, время образования единиц переноса будет равно сумме времен запаздывания в цепях ламп $Л_1$ и $Л_2$ всех разрядов.

Во втором такте логические схемы во всех разрядах одновременно устанавливают триггеры регистра суммы

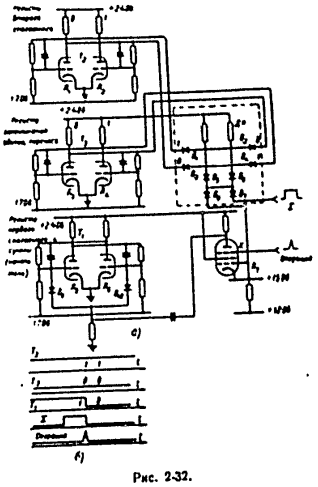


Рис. 2-32.

(накопителя) в состояние, которое определяется согласно правилу 2.

Рассмотрим действия логических схем во втором такте работы сумматора. На рис. 2-32,*а* представлена соответствующая схема для одного разряда сумматора. Логическая схема несоответствия *ДН* проверяет несовпадение состояний триггеров регистра второго слагаемого и регистра запоминания единиц переноса. Выход схемы несоответствия подан на первую сетку клапана *К*. На управляющий вход схемы несоответствия подается вспомогательный широкий положительный сигнал Σ. Если в это время триггеры T_1 и T_2 находятся в противоположных состояниях, то со схемы несоответствия на первую сетку вентиля подается высокий уровень напряжения, в противном случае — низкий уровень. Затем еще до того, как сигнал Σ исчезнет, на третью сетку клапана подается положительный сигнал *Операция* и вы-

полняется поразрядное суммирование (второй такт работы сумматора).

Если триггеры T_2 и T_3 находятся в разных положениях, то с клапана поступит отрицательный импульс на счетный вход триггера T_1 и последний опрокинется в противоположное состояние. Если же триггеры T_2 и T_3 находятся в одинаковом состоянии, клапан окажется закрытым по первой сетке низким уровнем напряжения и состояние триггера T_1 не изменится.

Комбинационные параллельные сумматоры, по-видимому, являются наиболее перспективными, так как лежащие в их основе логические схемы дают большие возможности для построения быстродействующих арифметических устройств.

2-7. Запоминающие устройства

Запоминающее устройство вычислительной машины можно осуществить при помощи триггерных схем, но в этом случае значительно увеличилось бы число ламп. Наиболее простым и одновременно достаточно надежным является запоминающее устройство с магнитным барабаном. По способу работы такое устройство аналогично звукозаписывающей аппаратуре. Двоичные числа записываются на поверхности вращающегося барабана, покрытой слоем ферромагнитного материала. Запись и чтение чисел производятся при помощи «головок» — миниатюрных электромагнитов, располагаемых у поверхности барабана с небольшим зазором (рис. 2-33,*а*).

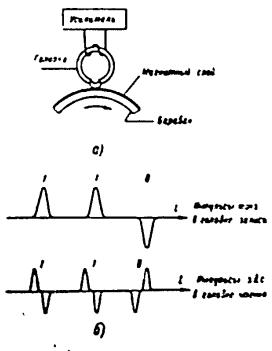


Рис. 2-33.

При прохождении через обмотку записывающей головки импульса тока в воздушном зазоре возникает магнитный поток. Часть потока замыкается через проходящий в этот момент под головкой участок ферромагнитного слоя барабана и намагничивает его, образуя на поверхности маленький магнитик. Полярность такого магнитика определяется полярностью импульса тока в обмотке головки записи. Для записи на барабан единиц и нулей в обмотку головки посылают импульсы тока противоположной полярности. В этом случае полярность магнитиков на поверхности барабана определяет значение записанных цифр (*1* или *0*). Чтение информации, записанной на поверхности барабана, производится при помощи «головок чтения». Головки записи и чтения имеют одинаковую конструкцию, а в ряде устройств одни и те же головки используются одновременно для записи и чтения. При прохождении намагниченных участков поверхности барабана в головках чтения наводятся напряжения, полярность (фаза) которых показывает, записана ли на данном участке барабана единица или нуль (рис. 2-33,*б*).

В магнитных запоминающих устройствах используется также и другой способ записи, при котором ферромагнитный слой предварительно намагничивается до насыщения в одну сторону. В эту же сторону направлено намагничивающее действие импульсов тока при записи, например, нулей. Импульсы единиц, имеющие противоположную полярность, намагничивают соответствующие участки поверхности барабана в противоположном направлении до насыщения. При таком способе записи в головках чтения наводится напряжение лишь при чтении единиц. Соответствующие кривые индукции B, создаваемой намагниченным слоем в головке чтения, и э. д. с. в ее обмотке показаны на рис. 2-34.

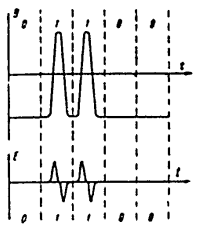


Рис. 2-34.

Существует несколько систем записи чисел на поверхности барабана. В одной из таких систем ячейки, в которые записываются двоичные числа, располагаются по образующим поверхности барабана. Значения всех разрядов двоичного числа записываются одновременно. При такой системе необходимо иметь столько головок записи и чтения, сколько двоичных разрядов имеют записываемые числа. Головки располагаются параллельно образующей барабана. На каждой дорожке поверхности барабана, проходящей под головкой, записываются цифры (*0* или *1*) соответствующего разряда всех чисел, помещаемых в запоминающее устройство.

Обычно объем запоминающего устройства с магнитным барабаном составляет 1 024 + 4 096 чисел или команд. Однако магнитные барабаны выполняются и для значительно большего объема памяти.

Для отыскания нужной ячейки при записи и чтении используются отметки (маркеры), постоянно нанесенные на барабан путем намагничивания или механическим путем и располагаемые на так называемой маркерной дорожке. На отдельной дорожке делается нулевая маркерная отметка; за нулевую отметку можно принять и одну из отметок на маркерной дорожке, но тогда эта отметка выполняется отличной от других, например путем намагничивания в противоположном направлении.

Один из вариантов схемы управления запоминающим устройством на магнитном барабане показан на рис. 2-35 [Л. 5]. Маркерные отметки создают импульсы в маркерной головке, которые, пройдя блок усиления и формирования УФ, поступают на вход триггерного счетчика маркерных импульсов. Число разрядов счетчика зависит от объема памяти. Например, при объеме памяти в 2 048 чисел счетчик имеет 11 разрядов. При каждом обороте барабана импульс от нулевой отметки ставит счетчик в нулевое состояние.

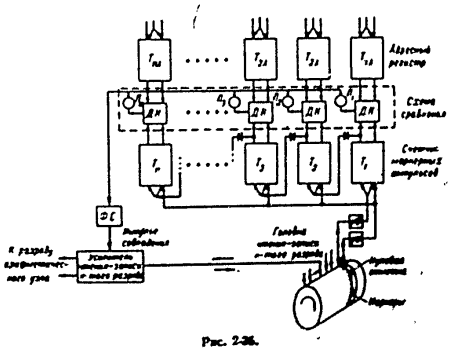


Рис. 2-35.

Номер (адрес) ячейки памяти, из которой должно быть прочитано число или куда число должно быть записано, поступает на триггерный «адресный регистр»; при помощи «схемы сравнения» состояние адресного регистра сравнивается с состоянием триггеров счетчика маркерных импульсов. Совпадение состояний триггеров этих регистров указывает на то, что под головки подошла нужная ячейка памяти. В этом случае схема сравнения формирует «импульс совпадения» и в соответствии с выполняемой операцией происходит чтение или запись числа.

Схема сравнения состоит из дешифраторов несоответствия *ДН*, на входы которых поданы уровни напряжений с анодов соответствующих триггеров адресного регистра и счетчика маркерных импульсов. Выходы дешифраторов присоединены к катодным повторителям, имеющим общую катодную нагрузку. В момент, когда все триггеры адресного регистра и счетчика окажутся в одинаковых состояниях, на выходе всех дешифраторов, а следовательно, и на сопротивлении нагрузки повторителей установится низкий уровень напряжения и в блоке *ФС* сформируется импульс совпадения.

При правильной работе счетчика в момент чтения сигнала нулевой метки (нулевого маркера) счетчик должен находиться в состоянии нуля. Это контролируется специальной схемой (на рис. 2-35 не показана), которая при неправильной работе счетчика блокирует работу схемы сравнения и включает звуковой сигнал.

Запоминающее устройство на магнитном барабане имеет существенный недостаток — сравнительно большое время выборки чисел из памяти, которое определяется от момента подачи приказа чтения числа до момента подхода к читающим головкам соответствующей ячейки памяти. Это время в среднем равно продолжительности полуоборота барабана. Необходимо учитывать, что выполнение одной операции требует, как правило, нескольких обращений к памяти. Время выборки числа из памяти на магнитном барабане или записи числа составляет обычно 5—10 *мсек*. В результате вычислительные машины с запоминающим устройством на магнитном барабане могут работать лишь со скоростью, не превышающей нескольких десятков или сотен операций в секунду. При этом не используются возможности быстродействия, заложенные в электронных арифметических и управляющих устройствах, которые могут выполнять тысячи операций в секунду.

Повышение скорости работы вычислительной машины не только увеличивает ее производительность, но и уменьшает вероятность отказов машины в процессе решения определенной задачи.

В качестве быстродействующих запоминающих устройств применяются ультразвуковые линии задержки, электронно-лучевые трубки и другие приборы. Если на экране одной электронно-лучевой трубки можно записать, например, $32 \times 32 = 1\,024$ цифр *0* или *1*, то для записи 1 024 двоичных 32-разрядных чисел окажется необходимым иметь 32 электронно-лучевые трубки. Время записи и чтения в запоминающем устройстве на электронно-лучевых трубках составляет около 10 *мксек*.

До последнего времени быстродействующие запоминающие устройства осуществлялись главным образом на электронно-лучевых трубках. В настоящее время все более широкое применение находят быстродействующие запоминающие устройства, в которых используются ферритовые сердечники. Для этой цели употребляются миниатюрные тороидальные сердечники, имеющие внешний диаметр 2—3 *мм* и менее. Ферритовые сердечники имеют почти прямоугольную петлю гистерезиса, и при некоторой идеализации можно считать, что кривая намагничивания сердечника соответствует рис. 2-36.

Если на обмотку сердечника подать серию импульсов с чередующейся полярностью, создающих в сердечнике напряженность магнитного поля, превышающую H_c (рис. 2-36), то каждый приходящий импульс перебрасывает сердечник из одного состояния намагничивания в противоположное. Благодаря прямоугольной петле гистерезиса сердечник устойчиво сохраняет сообщенное ему состояние намагничивания. Можно присвоить одному состоянию

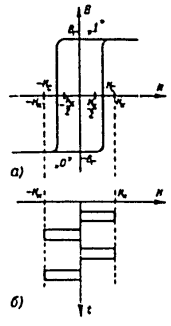


Рис. 2-36.

намагничивания значение *1*, а противоположному — значение *0* и использовать сердечник как элемент запоминающего устройства.

На рис. 2-37 приведена принципиальная схема запоминающего устройства в виде плоской матрицы, образованной сеткой из вертикальных и горизонтальных проволочных шин. В точках пересечения шин расположены тороидальные сердечники, причем через каждый сердечник проходят одна вертикальная и одна горизонтальная шина. Для чтения записанных сигналов служит общая обмотка, охватывающая все сердечники.

Управление записью и чтением в рассматриваемом устройстве производится по способу совпадения сигналов. Пусть, например, необходимо записать сигнал *1* в сердечнике *а*. В этом случае на горизонтальную и вертикальную шины сердечника *а* подаются совпадающие по времени сигналы $+I_s/2$.

Эти сигналы в сердечнике *а* складываются, образуя суммарный сигнал I_s. Величина сигнала I_s выбирается таким образом, чтобы удовлетворялись условия (рис. 2-36)

$$H_s > H_c \text{ и } \frac{H_s}{2} < H_c.$$

где H_s и $\frac{H_s}{2}$ — напряженности магнитного поля сердечника соответственно при сигналах I_s и $I_s/2$.

Если сердечник *а* находился в состоянии *0*, то после подачи сигналов $I_s/2$ на соответствующие шины суммарный сигнал перебросит сердечник *а* в состояние *1*. Если же сердечник *а* находился в состоянии *1*, то после подачи сигналов его состояние не изменится. Состояние всех других сердечников, расположенных на выбранных шинах, остается без изменений, так как сигнал $I_s/2$ недостаточен для изменения направления намагничивания сердечника. Запись сигнала *0* производится отрицательными импульсами $-I_s/2$.

Для чтения сигнала, записанного в сердечнике *а*, на те же шины подаются совпадающие по времени сигналы $-I_s/2$, имеющие полярность, противоположную сигналам записи *1*. Если в сердечнике *а* был записан сигнал *1*, то сигналы чтения, суммируясь, перемагнитят сердечник *а*, при этом в обмотке чтения наводится импульс. Если же в сердечнике *а* был записан *0*, сигналы чтения не меняют его состояния и в обмотке чтения импульс отсутствует.

При чтении *1* происходит опрокидывание сердечника в состояние *0*. Чтобы помещенная в память информация после чтения автоматически восстанавливалась, употребляются специальные схемы. Например, чтение может производиться двухполярными импульсами ($-I_s/2$, $+I_s/2$), причем

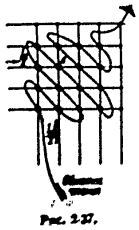


Рис. 2-37.

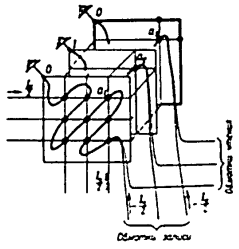


Рис 2-38

импульсы $+I_x/2$ (восстанавливающие состояние *1*) подаются на шины лишь при возникновении импульса *1* в обмотке чтения

На рис 2 38 показан один из вариантов схемы «объемной матричной памяти» с одновременной (параллельной) записью и чтением цифр всех разрядов двоичного числа Объемная матрица состоит из одинаковых плоских матриц, число которых равно числу двоичных разрядов чисел, хранимых в памяти Число сердечников в каждой плоской матрице равно количеству адресов памяти, т е количеству хранимых в памяти чисел Соответственно расположенные горизонтальные и вертикальные шины плоских матриц соединяются между собой последовательно Каждая плоская матрица имеет обмотку записи и обмотку чтения, охватывающие все ее сердечники (рис 2 38)

Выбор нужного адреса в объемной матрице производится по схеме совпадения одновременной подачей на соответствующие вертикальную и горизонтальную шины сигналов $+I_x/2$ или $-I_x/2$. При этом осуществляется одновременный выбор для записи или чтения одинаково расположенных сердечников во всех плоских матрицах, например сердечников a_1, a_2, a_3, . . в которых хранятся цифры соответствующих разрядов двоичного числа

При чтении на соответствующие шины подаются импульсы $-I_x/2$, при этом в обмотках чтения плоских матриц импульсы возникают, если в считываемых сердечниках хранились «единицы», и не возникают, если хранились «нули».

После чтения следует цикл восстановления информации

Записи предшествует цикл чтения без восстановления — в результате все сердечники выбранного адреса устанавливаются в состояние *0* Затем на соответствующие шины подаются импульсы $I_x/2$, при этом, если в сердечники данных разрядов записывается *0* то в обмотки записи подается импульс $-I_x/2$ и не подается никакого импульса, если записывается 1

Управление матричной памятью осуществляется специальными матричными переключателями, выполняющими задачу преобразования адреса числа (заданного в виде определенной комбинации уровней напряжений триггеров адресного регистра) в импульсы на соответствующих горизонтальной и вертикальной шинах матричной памяти

Запоминающие устройства на ферритовых сердечниках имеют ряд преимуществ по сравнению с устройствами на электронно-лучевых трубках, так как позволяют повысить надежность работы памяти, уменьшить количество и габариты электронной аппаратуры В отличие от устройства с магнитным барабаном запоминающее устройство на ферритовых сердечниках не имеет движущихся частей Оно позволяет уменьшить время выборки данных из памяти до нескольких (6—8) *мксек*.

2-8. Устройства ввода и вывода

Для ввода в машину программы, записанной на перфорированную ленту, употребляются приборы с фотоэлементами При прохождении мимо фотоэлементов отверстий перфорированной ленты на выходных зажимах устройства ввода возникают соответствующие комбинации импульсов, которые поступают в вычислительную машину Употребляется также ввод данных с магнитной ленты.

На рис. 2-39, *а* в качестве примера показана система перфорации цифр, знаков плюса и минуса и служебных операций на пятипозиционной ленте, употребляемая в машине М-3. Цифры

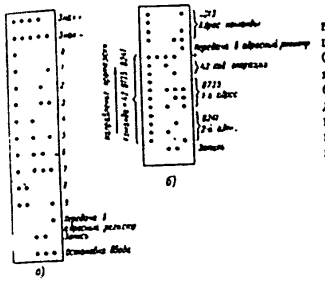


Рис 2-39.

десятичных и восьмеричных чисел перфорируются в двоично-кодированной форме Отверстия в левом крайнем столбце служат признаком, что в данной строчке записано число или знак числа.

Коды служебных операций (передача в адресный регистр, запись, остановка ввода и др.) не имеют отверстий в левом крайнем столбце перфорированной ленты. Таким образом, каждой цифре восьмеричного и десятичного чисел, знакам и служебным операциям соответствует одна строчка отверстий на ленте.

На рис. 2-39,*б* показан пример перфорации команды +42 0735 0241 *, которая должна быть помещена в ячейку памяти 1013. Вначале перфорируется цифра за цифрой номер ячейки (1013), куда должна быть передана команда. Затем перфорируется служебная операция *Передача в адресный регистр*. После этого перфорируются знак +, последовательно все цифры команды и, наконец, служебная операция *Запись*. Затем такой цикл повторяется для следующей команды и т. д.

* В соответствии с кодом команд машины М-3 (см. приложение) эта команда означает: содержимое ячейки 0241 разделить на содержимое ячейки 0735. Частное записать в ячейку 0241 запоминающего устройства и отпечатать. В начале команды перфорируется знак +, чтобы команда занимала то же число разрядов, что и числа.

При протяжке ленты аппаратура ввода прочитывает и передает в машину одну за другой строчки кодов. Сначала в машину передается адрес ячейки (1013), в которую должна быть помещена команда. После ввода в машину кода служебной операции «запись» вводимая команда запишется в запоминающее устройство в ячейку, номер которой установлен на адресном регистре (1013).

Аналогичным образом вводятся в машину числа [1]. Ввод можно осуществлять без пробивки на ленте адресов, по которым размещаются вводимые числа и команды. В этом случае размещение чисел и команд по адресам запоминающего устройства производится машиной при помощи вспомогательной программы.

Для автоматической печати данных, поступающих с вычислительной машины, можно использовать электромагнитные печатающие аппараты. Для ускорения вывода данных вместо буквопечатающих электромагнитных аппаратов употребляется запись данных на магнитную ленту или фотопленку.

2-9. Устройства управления

Отдельные узлы машины — запоминающее устройство, арифметическое устройство, устройства ввода и вывода — имеют свои собственные блоки управления. Общее устройство управления машиной осуществляет взаимодействие ее отдельных узлов с тем, чтобы процесс вычислений осуществлялся автоматически по заранее составленной программе, помещенной в запоминающее устройство.

Устройство управления в определенной последовательности вырабатывает управляющие импульсы, приводящие в действие соответствующие узлы машины для осуществления следующих элементарных операций:

1. Чтение кода очередной команды из запоминающего устройства и передача его в соответствующие узлы машины.

[1] На пульте машины М-3 имеется переключатель на ввод восьмеричных и десятичных чисел.

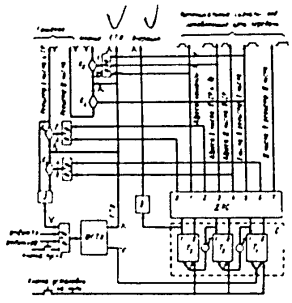


Рис 2-40

2 Чтение кодов чисел из запоминающего устройства и передача их на регистры арифметического узла

3 Выполнение операции, указанной в коде команды (арифметических операций, передачи числа в другую ячейку памяти и т д)

4 Передача полученного результата в запоминающее устройство и на выходное печатающее устройство (если это предусмотрено командой)

5 Образование адреса следующей команды, выполняемой машиной

В вычислительных машинах употребляются две системы управления асинхронная и синхронная При асинхронном управлении осуществляется контроль за окончанием элементарных операций, на которые разбивается процесс выполнения команды (чтение команды, чтение чисел, арифметическая операция, запись числа и др) Машина переходит к выполнению следующей элементарной операции лишь после получения сигнала о выполнении предыдущей При синхронном управлении контроль за окончанием отдельных операций отсутствует На выполнение каждой элементарной операции отводится определенное время, соответствующее одному или нескольким тактирующим (синхронизирующим импульсам), а затем машина переходит к следующей операции.

Рассмотрение различных принципов построения устройств управления вычислительными машинами выходит за рамки настоящей книги Мы ограничимся кратким описанием принципа действия устройства управления машины М-3, построенного по асинхронной системе [Л 3 и 5].

Устройство центрального управления машины М-3, помимо блока, который вырабатывает импульсы, управляющие работой отдельных узлов машины, содержит также два адресных регистра а) селекционный регистр (СР), в котором устанавливается номер ячейки запоминающего устройства, откуда производится чтение или куда записывается результат; б) пусковой регистр (ПР), в котором образуется адрес следующей команды

Код операции передается в блок операций (БО) устройства управления арифметическим узлом (устройство местного управления), которое по получении управляющего импульса от устройства центрального управления расшифровывает этот код и осуществляет управление операцией, выполняемой арифметическим узлом (сложение, умножение и т д.)

На рис 2-40 приведена упрощенная схема устройства центрального управления (адресные регистры не показаны), а на рис. 2-41 — ее временная диаграмма Цикл работы устройства центрального управления при выполнении полной двухадресной команды состоит из восьми тактов, которые определяются состояниями двоичного трехразрядного счетчика C, образованного триггерами T_1, T_2, T_3 Перед началом рабочего цикла счетчик находится в нулевом состоянии, а затем при каждом такте его состояние изменяется на единицу После окончания цикла работы счетчик снова оказывается в состоянии нуля.

Импульсы ответов об окончании элементарных операций чтения и записи — из запоминающего устройства, печатания — из устройства ввода и вывода *УВВ*, а также из схемы центрального управления поступают на вход схемы *ФГТИ*, которая формирует из них главные тактовые положительные импульсы управления *ГТИ* (всего восемь импульсов в рабочем

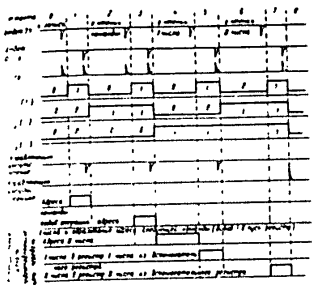


Рис 2-41

цикле) и с некоторой задержкой отрицательные импульсы, поступающие на вход счетчика.

Напряжения с анодов триггеров счетчика поданы на входы диодной логической схемы *ДЛС*, и в зависимости от состояния счетчика на соответствующем выходе этой схемы появляется потенциальный сигнал высокого уровня, который проходит через дешифраторы и открывает нужные клапаны. Тем самым производится подготовка к выполнению определенных элементарных операций (передача кодов из одного регистра в другой, гашение регистра и др) Сами же эти операции производятся главными тактовыми импульсами, которые проходят через подготовленные таким образом клапаны.

Для пуска машины необходимо в пусковом регистре установить адрес первой команды. Счетчик посредством *Кнопки установки на нуль* ставится в нулевое состояние. От *Кнопки пуска* на вход схемы управления подается отрицательный импульс, который действует как импульс ответа запоминающего устройства, после чего машина начинает работать автоматически.

В нулевом такте рабочего цикла (счетчик C находится в положении нуля) из запоминающего устройства в схему управления подается отрицательный импульс *Ответ ЗУ*, указывающий на окончание записи результата, полученного при выполнении предыдущей команды. От этого импульса формируется в блоке *ФГТИ* главный тактирующий (положительный) импульс *ГТИ*, который поступает на клапан K_1, на второй вход которого в нулевом состоянии счетчика с выхода *0* схемы *ДЛС* подается напряжение высокого уровня. Сформированный клапаном K_1 отрицательный импульс *Гашение* ставит на нуль регистр первого числа арифметического узла и селективный регистр адреса. Этот же импульс через цепь задержки подается на вход схемы управления (импульс *Ответ ФГТИ*).

Схема *ФГТИ* с некоторым сдвигом по времени относительно момента образования импульса *ГТИ* создает отрицательный импульс, перебрасывающий счетчик в состояние *1*, и начинается первый такт работы. В этом состоянии счетчика на выходе *1* схемы *ДЛС* появляется сигнал, открывающий соответствующие клапаны и подготавливающий передачу адреса очередной команды из пускового регистра в селекционный.

Затем на вход управления приходит из цепи задержки сигнал *Ответ ФГТИ*, соответствующий выработанному схемой в предыдущем такте управляющему импульсу *Гашение*. Из импульса *Ответ ФГТИ* схема формирует второй положительный импульс *ГТИ*, который осуществляет передачу адреса очередной команды в селекционный регистр [1].

Импульс *ГТИ* проходит через клапан K_2, открытый в состоянии *1* счетчика, и образует управляющий импульс *Чтение*, который вызывает чтение команды во вспомогательный регистр арифметического узла. Затем со сдвигом по времени образуется отрицательный импульс, перебрасывающий счетчик в состояние *2*.

Работа схемы управления прекращается до прихода импульса *Ответ ЗУ*, поступающего в конце второго такта из запоминающего устройства в момент чтения команды. Из импульса *Ответ ЗУ* схема *ФГТИ* формирует тре-

[1] Работа схемы передачи кодов из одного регистра в другой описана в § 2-4.

тий положительный импульс *ГТИ*, а затем с некоторым сдвигом по времени образуется импульс, перебрасывающий счетчик в состояние *3*

Третий импульс *ГТИ* поступает на клапан K_1, открытый в состоянии *2* счетчика, и на выходе клапана снова формируется отрицательный импульс гашения регистра первого числа и селекционного регистра Этот же импульс, пройдя цепь задержки, попадает на вход схемы управления (импульс *Ответ ФГТИ*), когда счетчик уже находится в положении *3* Из него схема формирует четвертый импульс *ГТИ*, который, пройдя через соответствующие клапаны, открытые потенциальным сигналом высокого уровня на выходе *3* схемы *ДЛС* (в положении счетчика *3*), осуществляет передачу из вспомогательного регистра кода операции в блок операции, адреса первого числа в селекционный регистр и добавление единицы младшего разряда в пусковой регистр (образование адреса следующей команды) Кроме того, импульс *ГТИ* на клапане K_2 формирует управляющий импульс *Чтение*, вызывающий чтение первого числа во вспомогательный регистр Затем с некоторым сдвигом по времени формируется импульс, перебрасывающий счетчик в состояние *4*, после чего схема управления ожидает сигнала от запоминающего устройства о чтении первого числа

В момент чтения первого числа на вход схемы управления поступает сигнал *Ответ ЗУ* и образуется пятый импульс *ГТИ*, который проходит через клапаны, открытые потенциальным сигналом высокого уровня на выходе *4* схемы *ДЛС* и производит передачу адреса второго числа из вспомогательного регистра в селекционный регистр Этот же импульс на клапане K_3 формирует импульс *Ответ ФГТИ*, поступающий в цепь задержки

Затем с небольшим сдвигом по времени счетчик переходит в состояние *5* После этого из цепи задержки на вход схемы подается сигнал *Ответ ФГТИ* и образуется шестой импульс *ГТИ*, который проходит через клапаны, открытые сигналом высокого уровня на выходе *5* схемы *ДЛС*, и производит передачу первого числа из вспомогательного регистра в регистр первого числа. Одновременно импульс *ГТИ* в клапане K_2 формирует импульс *Чтение* (второго числа), а в клапане K_4 — импульс *Гашение регистра второго числа*, подготовляя этот регистр к приему второго числа.

Затем с небольшим сдвигом по времени счетчик переходит в положение *6* и схема ждет ответного сигнала запоминающего устройства, приходящего в момент чтения второго числа

Сигнал *Ответ ЗУ* образует в схеме *ФГТИ* седьмой импульс *ГТИ*, который в клапане K_3 образует импульс *Ответ ФГТИ*, поступающий в цепь задержки Счетчик переходит в положение *7*, при котором открываются клапаны передачи числа из вспомогательного регистра в регистр второго числа, а сама передача происходит под действием восьмого импульса *ГТИ* после того, как из цепи задержки на вход схемы поступает импульс *Ответ ФГТИ* Затем образуется импульс, перебрасывающий счетчик в положение *0*

В момент переброса триггера T_1 счетчика в нулевое положение запускается блокинг-генератор *Б*, создающий положительный управляющий импульс *Операция*, которым приводится в действие устройство местного управления, выполняющее операции в арифметическом узле в соответствии с кодом, установленным в блоке операции

2-10. Основные характеристики цифровых вычислительных машин

По своему назначению вычислительные машины могут быть универсальными или специализированными, т е предназначенными лишь для определенного типа задач. В последнем случае конструкция машины упрощается.

К основным характеристикам вычислительных машин относятся количество разрядов, с которыми оперирует машина (обычно 30—40 двоичных разрядов), объем внутреннего (оперативного) и внешнего запоминающих устройств, скорость работы, способ введения запятой, употребляемая в машине система счисления, а также особенности кода машины (одноадресный, двухадресный или трехадресный) и характер выполняемых операций. Кроме того, вычислительная машина характеризуется числом используемых ламп, потребляемой энергией, необходимой площадью для размещения машины и количеством обслуживающего персонала

Универсальные цифровые вычислительные машины можно разбить на три класса: большие, средние, малые.

Большие вычислительные машины предназначаются главным образом для решения задач современной физики и наиболее сложных задач современной техники Эти машины работают со скоростью в несколько тысяч или десятков тысяч операций в секунду и содержат несколько тысяч электронных ламп.

Примером машины такого класса служит машина БЭСМ, построенная под руководством акад. С. А. Лебедева [Л. 16]. Машина работает со скоростью 8—10 тыс. операций в секунду и имеет 4 000 ламп.

Машина БЭСМ имеет быстродействующее запоминающее устройство на ферритовых сердечниках емкостью 1 024 чисел. Кроме того, имеется запоминающее устройство на магнитном барабане емкостью в 5 120 чисел и на магнитных лентах практически неограниченной емкости. Машина оперирует с двоичными числами с плавающей запятой, причем 32 двоичных разряда используются для представления мантиссы числа, 5 разрядов—для порядка числа и 2 разряда — для знаков мантиссы и порядка.

Малые вычислительные машины предназначаются главным образом для инженерных и экономических расчетов.

Для уменьшения объема аппаратуры эти машины обычно выполняются с фиксированной запятой. Число электронных ламп около 1 000 или менее.

Основным запоминающим устройством является магнитный барабан. Специального внутреннего (оперативного) быстродействующего запоминающего устройства эти машины не имеют или же имеют такое устройство из небольшое число адресов (10—100).

При отсутствии ячеек быстрой памяти скорость работы машин этого типа в зависимости от скорости вращения барабана составляет обычно 30—60 полных арифметических операций в секунду.

В малых машинах имеет место тенденция к уменьшению времени обращения к магнитному барабану путем применения двух головок на каждой дорожке барабана и переключения чтения и записи на головки, к которым раньше подойдет ячейка с нужным адресом. Кроме того, для этой же цели употребляется система программирования, при которой код команды содержит адрес следующей команды. В этом случае удается распределить команды и числа по барабану таким образом, чтобы возможно больше операций производилось за один оборот барабана.

К классу малых вычислительных машин относятся советские машины М-3 и «Урал», предназначенные для инженерных исследований. Основные характеристики этих машин приведены в следующих параграфах.

Вычислительные машины среднего класса имеют характеристики, промежуточные между большими и малыми машинами. Например, вычислительная машина М-2, разработанная под руководством чл.-корр. АН СССР И. С. Брука, имеет 1 676 ламп и скорость работы 2 000 операций в секунду. Эта машина имеет внутреннее оперативное запоминающее устройство на электронно-лучевых трубках емкостью 512 чисел. Машина может работать с плавающей или фиксированной запятой. Разрядная сетка машины имеет 34 двоичных разряда [Л. 4].

2-11. Универсальная цифровая вычислительная машина М-3

Малогабаритная универсальная цифровая вычислительная машина М-3 разработана и построена коллективом сотрудников ЛУМС АН СССР и НИИ ЭП Госплана СССР [Л. 3 и 5]. Машина предназначена для вычислительных лабораторий, конструкторских бюро и научно-исследовательских институтов.

Основные данные машины М-3 следующие

Точность вычислений	До 9 десятичных разрядов
Разрядность чисел при вводе в машину и выводе на печать	7 десятичных разрядов или 10 восьмеричных
Система счисления для чисел при вводе в машину и выводе на печать	Десятичная и восьмеричная
Система счисления для чисел и команд в машине	Двоичная
Разрядность чисел в машине	30 двоичных разрядов и один разряд знака числа
Форма представления чисел в машине	С запятой, фиксированной перед старшим разрядом
Система программирования	Двухадресная и одноадресная
Количество команд	48
Скорость выполнения операций в арифметическом узле	
сложение	60 *мксек*
вычитание	75—120 *мксек*
умножение	1 900 *мксек*
деление	2 000 *мксек*
Общая скорость работы (при скорости вращения барабана 1 000 об/мин)	1 800 (в среднем) полных арифметических действий с записью результата в память в минуту
Емкость запоминающего устройства на магнитном барабане	2 048* чисел или двухадресных команд
Скорость ввода с перфорированной ленты	
электромеханический транслятор	30 чисел в минуту
фототранслятор	1 200 чисел в минуту
Выводное устройство:	
Скорость печати и перфорации результата	30 чисел в минуту
Число ламп в машине	770
Потребляемая мощность	10 *ква*
Площадь, занимаемая шкафами машины	3,3 *м*²

* В первых образцах 1 125.

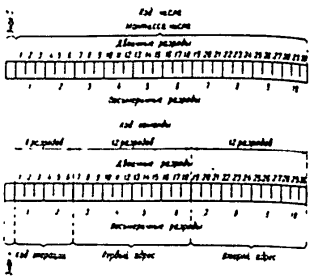


Рис 2-42

Объем запоминающего устройства может быть увеличен до 4 096 добавлением второго запоминающего устройства на магнитном барабане или ферритовых сердечниках

Машина М-3 является машиной параллельного действия Она имеет арифметическое и запоминающее устройства параллельного типа Система управления машиной — асинхронная

Предусмотренная возможность добавления к комплекту машины узла быстродействующей ферритовой памяти позволяет повысить скорость работы М 3 до 1 500—2 000 операций в секунду, что соответствует скорости работы арифметического и управляющего устройств машины.

На рис 2-42 изображены коды чисел и команд машины М-3. Числа в машине представляются в виде 30 разрядных двоичных чисел с запятой фиксированной перед старшим разрядом 31-й разряд служит для кодирования знака числа (в этом разряде *0* соответствует плюсу, а *1* — минусу)

Основная система программирования — двухадресная. Для записи команд используются 30 двоичных разрядов числа, при этом код знака числа безразличен Шесть старших двоичных (или два восьмеричных) разрядов числа используются для кода операции, остальные разряды разбиваются на две группы по 12 двоичных разрядов (или по четыре восьмеричных) и используются для записи адресов первого и второго чисел.

Машина выполняет все арифметические операции (сложение, вычитание, умножение, деление), а также ряд логических и вспомогательных операций — передача числа из одной ячейки памяти в другую, логическое умножение, безусловная и условная передачи управления, ввод чисел с перфоленты. Операция условного перехода в зависимости от знака предыдущей операции передает управление по первому или второму адресу команды условного перехода.

Команды, т. е. коды операций и адреса, записываются в восьмеричной системе. Тип операции (сложение, умножение и др.) кодируется второй восьмеричной цифрой кода операции. В случае арифметических операций и логического умножения первая восьмеричная цифра кода определяет дополнительную особенность выполняемой операции: производится ли запись в запоминающее устройство, печатание и т. д. (см. табл. 2-3).

Таблица команд машины М-3 дана в приложении 1.

Машина М-3 может работать как двухадресная и как одноадресная с накопителем. В последнем случае используется результат предыдущего действия, сохраненный на регистре арифметического узла. Одноадресные операции формально записываются как двухадресные, при этом во второй адрес ставятся нули.

Принципы действия арифметического, запоминающего и управляющего устройств машины М-3 кратко описывались в соответствующих параграфах настоящей главы

Таблица 2-3

Строение кодов операций машины М-3

Восьмеричная цифра кода операции	Вид операции машины (кодируется второй восьмеричной цифрой кода операции)	Особенность операции (кодируется первой цифрой кода операции)[1]
0	Сложение	Операция над числами из первого и второго адресов с записью результата по второму адресу
1	Вычитание	То же, что для 0, но без записи результата в запоминающее устройство
2	Деление	Операция над числом из первого адреса и результатом предыдущей операции, хранящимся в арифметическом устройстве, с записью результата по второму адресу
3	Умножение	То же, что для 2, но без записи результата в запоминающее устройство
4	Условная и безусловная передачи управления	То же, что для 0. Кроме того, результат печатается
5	Передача числа из одной ячейки запоминающего устройства в другую	То же, что для 1, но действия выполняются над модулями чисел
6	Логическое умножение	То же, что для 2. Кроме того, результат печатается
7	Ввод	То же, что для 3, но действие производится над модулями чисел

[1] Относится только к арифметическим операциям и логическому умножению.

2-12. Универсальная цифровая вычислительная машина «Урал»

Вычислительная машина «Урал»[1] относится к классу малых автоматических цифровых вычислительных машин. Эта машина предназначена для инженерных исследований и имеет следующие основные данные[2]

Точность вычислений	До 10 десятичных разрядов
Разрядность десятичных чисел при вводе в машину и выводе на печать	9 десятичных разрядов (последний разряд вводится четным)
Система счисления для чисел при вводе в машину и выводе на печать	Десятичная и восьмеричная
Система счисления для чисел и команд в машине	Двоичная
Разрядность чисел в машине	35 двоичных разрядов и один разряд для знака числа
Форма представления чисел в машине	С запятой, фиксированной перед старшим разрядом
Система программирования	Одноадресная
Общая скорость работы (при скорости вращения барабана 6 000 *об/мин*)	100 одноадресных одноктактных операций
Количество команд	30
Число тактов при выполнении отдельных операций	
все операции, кроме деления, нормализации и групповых	1 такт
нормализация	2 такта
деление	4 такта
групповые	Зависит от команды
Длительность одного такта	10 *мксек*
Емкость запоминающего устройства на магнитном барабане	1 024 числа или 2 048 одноадресных команд
Накопитель на магнитной ленте	40 000 тридцатишестиразрядных чисел или 80 000 команд
Накопитель на перфорированной ленте	До 10 000 чисел или команд
Скорость ввода с перфорированной и магнитной ленты	4 500 чисел в минуту
Скорость печати результатов	100 чисел в минуту
Число ламп в машине	700
Источник энергии	Сеть трехфазного тока, напряжение 220 *в* ± 10%, частота 50 *гц*
Потребляемая мощность	7,5 *ква*

На рис. 2-43 представлены коды чисел и команд машины «Урал». Числа в машине представляются посредством 35 двоичных разрядов (соответствует 10,5 десятичным разрядам). Запятая фиксирована перед старшим разрядом числа, 36-й разряд отводится для представления знака (плюс изображается нулем, минус — единицей). Таким образом, диапазон модулей чисел, с которыми машина может оперировать, составляет от 2^{-35} до 1. На рис. 2-43,*а* и *б* показаны размещения двоичного числа в полной и неполной ячейках.

Код команды (рис. 2-43,*в*) занимает половину разрядной сетки, т. е. 18 двоичных разрядов, причем 5 разрядов используются для кода операции и 11 разрядов — для адреса числа. Кроме того, по одному разряду отводится для кодирования признака полной ячейки и признака блокировки переполнения. В полную ячейку запоминающего устройства помещаются две команды.

На рис. 2-44 представлена блок-схема арифметического узла машины «Урал». Арифметический узел этой машины состоит из накапливающего сумматора, основного регистра, регистра частного, выходного регистра, местного программного датчика (блока местного управления) и входного сдвигателя.

Машина «Урал» имеет арифметический узел последовательно-параллельного действия. Из запоминающего устройства на магнитном барабане (а также с магнитной ленты) коды чисел подаются в главный регистр через входной сдвигатель по девять двоичных разрядов параллельно и четырьмя группами последовательно. Аналогичным образом, т. е. по девять разрядов параллельно четырьмя группами последовательно, производится передача кодов числа из сумматора в магнитный барабан и на магнитную ленту. В машине применяется накапливающий сумматор параллельного действия.

В сумматоре для представления положительных чисел используется прямой код, а для представления отрицательных — обратный. В запоминающих устройствах и положительные и отрицательные числа представляются в прямом коде.

Синхронизирующие (тактирующие) импульсы подаются с магнитного барабана (один такт за один оборот). Сложение выполняется за один такт, а деление — за четыре такта.

Полная таблица команд машины «Урал» дана в приложении 2. При сложении или вычитании число, находящееся в ячейке с адресом, указанным в команде, прибавляется к числу, стоящему в сумматоре, или вычитается из него. При умножении число в сумматоре умножается на число из ячейки с адресом, указанным в команде; произведение образуется в сумматоре. При делении делимое находится в сумматоре, а делитель в ячейке с адресом, указанным в команде.

Кроме того, умножение может производиться также следующим образом: число, находящееся в регистре арифметического узла, умножается на число из ячейки с адресом, указанным в команде. Произведение прибавляется к числу, находящемуся в сумматоре. Эта операция удобна при вычислении сумм парных произведений $a_1b_1+a_2b_2$, так как позволяет сократить число обращений к запоминающему устройству.

Достоинством машины «Урал» является наличие таких операций, которые в ряде случаев могут способствовать некоторому сокращению объема программы и повышению скорости работы (нормализация, сдвиг кодов на заданное число разрядов, суммирование парных произведений, повторение циклов счета заданное число раз). Напри-

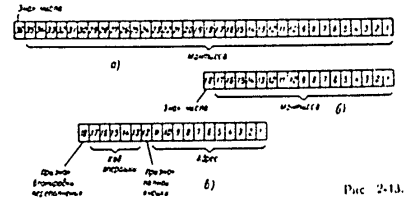


Рис. 2-43.

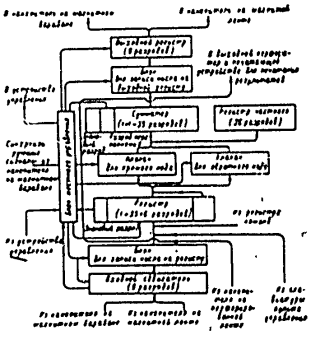


Рис. 2-44.

мер, наличие операций сдвига кода на заданное число разрядов и нормализации ускоряет процесс вычислений при работе с искусственной плавающей запятой[1].

[1] При написании настоящего параграфа использована статья Ю. Я. Базилевского (Л. 1).

[2] Данные приводятся по проспекту «Универсальная цифровая машина „Урал"», Всесоюзная Промышленная выставка, 1956.

[1] Вычисления с плавающей запятой могут производиться на машине с фиксированной запятой при помощи специальных подпрограмм.

ГЛАВА ТРЕТЬЯ

ТЕХНИКА ПРОГРАММИРОВАНИЯ

3-1. Простейший пример программы

В гл. I было дано общее определение программы решения на машине математических задач, описаны различные коды автоматических цифровых вычислительных машин и методы хранения команд в машине. В настоящей главе мы рассмотрим основные приемы, при помощи которых составляются программы, и при этом, если это специально не оговорено, будем вести все рассуждения для рассмотренной в гл. I условной машины с плавающей запятой. Начнем с простейшего примера, приведенного в § 1.3. Как мы отмечали, для вычисления определителя

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

требуется выполнить следующие действия

1	$\times$	a	d	ad
2	$\cdot$	b	c	bc
3	$-$	ad	bc	$ad-bc$

Чтобы реализовать эти вычисления на машине, нужно в первую очередь поместить числа a, b, c и d в запоминающее устройство машины. Пусть эти числа хранятся соответственно в ячейках памяти α, β, γ и δ. Кроме того, необходимо выделить ячейки для хранения получаемых произведений ad и bc и окончательного ответа $ad-bc$. Пусть это будут ячейки ζ_1, ζ_2 и ξ соответственно. Для вывода из машины полученного результата мы к трем указанным действиям добавим команду, которая будет отпечатывать полученный результат, т. е. отпечатывать содержимое ячейки ξ. Для прекращения работы машины в конце программы надо поставить команду «Стоп». Теперь можно составить программу вычисления на машине определителя второго порядка. Эта программа приведена ниже в таблице. Она содержит пять команд. Слева от каждой команды указан номер ячейки, в которой она хранится, а справа — результат операции.

Номер ячейки	Операция	$1A$	$2A$	$3A$	Результаты операции
$k+1$	$\times$	a	b	ζ_1	ad
$k+2$	$\times$	β	γ	ζ_2	bc
$k+3$		ζ_1	ζ_2	ξ	$ad-bc$
$k+4$	Печать	ξ	—	—	$ad-bc$
$k+5$	Стоп				
α	a				
β	b				
γ	c				
δ	d				

В таблице под командами указано, в каких ячейках хранятся числа.

Для постановки на машине такой программы нужно сделать следующее. Сначала ячейкам α, β, γ, δ, ζ_1, ζ_2 и ξ должны быть приданы конкретные номера. После этого надо закодировать в численной форме каждую команду. В результате получается программа в виде последовательности пяти двоичных чисел. Для хранения программы должны быть выделены какие-то пять последовательных ячеек памяти. Их номера обозначены через $k+1 - k+5$. Наконец, надо ввести программу и числа a, b, c и d в отведенные для них ячейки, после чего можно приступить к решению задачи. Для этого набирают на адресном регистре номер $k+1$, т. е. номер ячейки, в которой хранится первая команда программы, и включают машину. Машина будет выполнять команду за командой, выбирая их из последовательных ячеек памяти, начиная с ячейки $k+1$. Таким образом, она вычислит число $ad-bc$, отпечатает его (команда $k+4$), и, наконец, команда $k+5$ остановит машину. Заметим, что для решения этой задачи понадобилось 12 ячеек памяти: пять для хранения команд, четыре для хранения исходных чисел и три для хранения результатов вычислений.

В качестве второго примера будет рассмотрена программа умножения двух комплексных чисел:

$$(a_1+ib_1)(a_2+ib_2)=a_3+ib_3,$$

где, как известно,

$$a_3=a_1a_2-b_1b_2, \quad b_3=a_1b_2+a_2b_1.$$

Хранить в машине комплексные числа как таковые нельзя и приходится хранить отдельно вещественную часть и коэффициент при мнимой части. Для этого можно использовать, например, ячейки α_1, β_1, α_2 и β_2.

Точно так же и результат перемножения должен быть получен в двух отдельных ячейках α_3 и β_3. Поэтому программа перемножения двух комплексных чисел будет иметь вид

$k+1$	$\cdot$	α_1	α_2	ζ_1	a_1a_2
$k+2$	$\cdot$	β_1	β_2	α_3	b_1b_2
$k+3$		ζ_1	α_3	α_3	$a_1a_2-b_1b_2=a_3$
$k+4$	$\cdot$	α_1	β_2	ζ_1	a_1b_2
$k+5$	$\times$	α_2	β_1	β_3	a_2b_1
$k+6$	$+$	ζ_1	β_3	β_3	$a_1b_2+a_2b_1=b_3$
$k+7$	Печать	α_3			a_3
$k+8$	Печать	β_3			b_3
$k+9$	Стоп				
α_1	a_1				
β_1	b_1				
α_2	a_2				
β_2	b_2				

В этой программе результаты промежуточных действий a_1a_2 и a_1b_2 помещены в ячейку ζ_1. Такие ячейки, в которых хранятся результаты промежуточных действий, называются обычно рабочими ячейками. Произведение a_1b_2 можно поместить в ту же ячейку ζ_1, так как хранящееся в ней число a_1a_2 нам больше не нужно.

В команде $k+2$ произведение b_1b_2 помещено в ячейку α_3, временно использованную, таким образом, в качестве рабочей ячейки.

Совершенно очевидно, что вычислять рассмотренный выше определитель или произведение комплексных чисел на машине значительно сложнее, чем вычислять вручную. В связи с этим надо отметить следующее. Обычно вычислительные процессы состоят

если составляя программу, отвести под каждое действие отдельную команду, то такая программа не будет иметь никакой практической ценности. С одной стороны, для ее составления потребуется такое же время, какое нужно для осуществления счета вручную, а с другой — машина должна иметь огромное число ячеек в запоминающем устройстве. Однако почти всякий вычислительный процесс состоит из многократно повторяющихся кусков или циклов небольшого числа операций,

в которых последовательность действий всегда одна и та же, но на различных этапах счета изменяются числа, над которыми производятся эти действия. Например, пусть численное решение обыкновенного дифференциального уравнения $y'=f(x, y)$ выполняется методом Эйлера по формуле

$$y_{i+1}=y_i+f(x_i, y_i)(x_{i+1}-x_i).$$

Для определения в каждой из точек $x_0, x_1, \ldots, x_n$ значений искомой функции $y(x)$ нужно выполнить одну и ту же последовательность действий, заменяя лишь при переходе от точки x_{i+1} к точке x_{i+2} значения x_i, y_i на x_{i+1} и вновь вычисленное значение y_{i+1}. Используя указанную особенность вычислительных процессов, мож-

ходимо уметь составлять такие программы (и в этом заключается все искусство программирования), в которых небольшим числом команд охватывается большое число действий. Это достигается при помощи двух основных приемов, рассмотренных в следующих параграфах.

3-2. Преобразование содержимого ячеек

Первый из этих приемов может быть легко понят на следующем простейшем примере. Пусть нужно вычислить на машине степень a^{20}. Для этого надо прежде всего поместить число a в некоторую ячейку α запоминающего устройства. В некоторой ячейке β мы будем последовательно образовывать степени a, a^2, a^3, ... Программа, вычисляющая a^{20}, будет иметь вид (справа выписан результат каждого действия):

$k+1$	ПЧ	,1*	—	β	1
$k+2$	×	α	β	β	a
$k+3$	ПУ	$k+2$	—		a^2
$k+4$					a^3
					.
$k+21$		α	β	β	a^{19}
$k+22$	Стоп				

Первая команда в ячейку β передает единицу. Вместо того, чтобы указывать в этой команде номер той ячейки, в которой хранится единица, написана просто 1, взятая в кавычки. Вообще впредь мы будем через «a» обозначать ту ячейку, в которой хранится число a.

Единица передана в ячейку β для того, чтобы можно было в команде $k+2$ умножить ее содержимое на содержимое ячейки α, т. е. образовать в ячейке β число a. Такая программа содержит 22 команды, но, как легко видеть, 20 из них совершенно одинаковы. Вместо того, чтобы выписывать 20 одинаковых команд, можно попытаться выполнить 20 раз одну и ту же команду. Для этого пользуются операцией передачи управления, при помощи которой программа приводится к следующему упрощенному виду (справа выписаны результаты действия на последовательных этапах счета):

$k+1$	ПЧ	,1*	—	β	1
$k+2$	×	α	β	β	a	a^2	a^3	. .
$k+3$	ПУ	$k+2$	—					

В этой программе команда $k+3$ передает управление в ячейку $k+2$, т. е. будет вновь и вновь выполняться команда $k+2$, но при этом содержимое ячейки β будет все время меняться. Если перед первым выполнением команды $k+2$ в ней стояло число 1, то перед вторым выполнением той же команды в ячейке β будет уже число a, перед третьим a^2 и т. д. Таким образом, в ячейке β последовательно образуются степени a, a^2, a^3, ... Так составленная программа имеет тот существенный недостаток, что машина не остановится, когда в ячейке β образуется нужное нам число a^{20}, а будет продолжать работать неопределенно долго, все время выполняя одни и те же операции $k+2$ и $k+3$; как говорят, машина «зациклится». Чтобы избежать этого, следует воспользоваться введенной в § 1-9 операцией сравнения. Для этого выделяется какая-нибудь ячейка γ, называемая счетчиком, в которую после каждого выполнения команды $k+2$ должна прибавляться единица (предварительно ячейку γ надо «очистить», т. е. передать в нее нуль). В ячейке γ будет подсчитываться, сколько раз выполнена команда $k+2$. Остается ввести команду сравнения, которая будет сравнивать число, образованное в ячейке γ, с числом 20. Пока это число меньше 20, операция сравнения будет отсылать к команде $k+2$ (в третьем адресе команды сравнения надо поставить ячейку $k+2$), но как только в ячейке γ образуется число 20, операция сравнения пропустит. Теперь программа будет иметь вид:

k	ПЧ	,0*	—	γ	0
$k+1$	ПЧ	,1*	—	β	1
$k+2$	×	α	β	β	a	a^2	a^3	. . .	a^{19}
$k+3$	+	,1*	γ	γ	1	2	3		20
$k+4$	≠	γ	,20*	$k+2$					
$k+5$	Стоп								

Как видно, в этой программе команды $k+2-k+4$ выполняются 20 раз. Такие группы команд, которые при работе машины выполняются несколько раз, в § 3-1 названы циклом. Только благодаря введению таких циклов удается составлять программы, содержащие небольшое число команд, по которым, однако, машина выполняет громадное число действий.

Если поместить числа 0, 1 и 20 соответственно в ячейки ζ_1, ζ_2 и ζ_3, то программа примет окончательный вид:

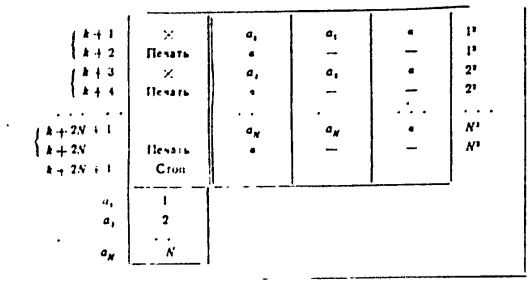
k	ПЧ	ζ_1	—	γ	0
$k+1$	ПЧ	ζ_2	—	β	1
$k+2$	×	α	β	β	a	a^2	a^3	. . .	a^{19}
$k+3$	+	ζ_2	γ	γ	1	2	3		20
$k+4$	≠	γ	ζ_3	$k+2$					
$k+5$	Печать	β							
$k+6$	Стоп								
ζ_1	0								
ζ_2	1								
ζ_3	20								
a	a								

Здесь опять в качестве k, $k+1$, ..., $k+6$, ζ_1, ζ_2, ζ_3, α, β и γ должны быть взяты определенные номера ячеек и программа должна быть закодирована, т. е. представлена в виде последовательности двоичных чисел. Заметим, что операция сравнения $k+4$ содержит в третьем адресе номер ячейки $k+2$, т. е. одну из тех ячеек, в которых хранится сама программа. Теперь числа 0, 1, 20 и программа могут быть введены в предназначенные для этого ячейки памяти и можно приступить к решению задачи. Набрав на пульте управления номер k и включив машину, можно совершенно не принимать никакого участия в дальнейшей работе машины. По составленной программе машина вычислит величину a^{20}, отпечатает ее и остановится. Следует особо обратить внимание на то, что число команд в этой программе совершенно не зависит от величины показателя степени 20. Чтобы вычислить по той же программе любую степень a^n, нужно лишь в ячейке ζ_3 (а не в самой программе!) заменить число 20 на n.

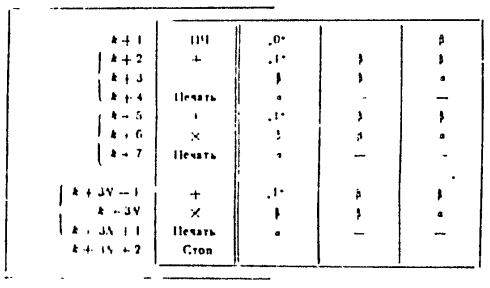
Рассмотрим несколько более сложный пример. Пусть нужно вычислить квадраты последовательных чисел

$$1^2, 2^2, 3^2, \ldots, N^2.$$

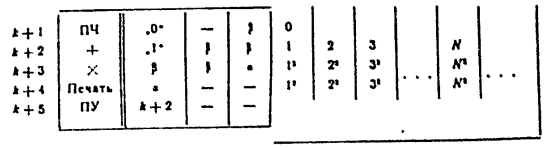
Для этого числа 1, 2, ..., N помещаются соответственно в ячейки $ЗУ$ с номерами $a_1, a_2, \ldots, a_N$. Программа может иметь следующий вид:



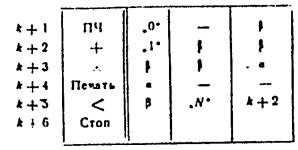
Как видно, программа вычисления квадратов N чисел требует $3N+2$ ячейки запоминающего устройства (N ячеек $a_1, a_2, \ldots, a_N$ для хранения чисел 1, 2 ... N, ячейку α и $2N+1$ ячейку для хранения команд). Вся программа состоит из N раз повторяющегося куска, содержащего две команды. Вторые команды в нем одни и те же, а первые отличаются тем, что в них умножаются сами на себя числа 1, 2,...,N. Однако числа 1, 2,...,N можно образовывать в одной и той же ячейке β последовательно, прибавляя к ее содержимому единицу; тогда в командах $k+1$, $k+3$, . . $k+2N-1$ ячейки a_1, a_2, . . ., a_N могут быть заменены ячейкой β и программа примет следующий вид



Теперь уже вся программа состоит из N раз повторяющегося цикла, содержащего три совершенно одинаковые команды (первая команда «очищает» ячейку β, т. е. передает в нее 0). При повторении цикла вид этих команд не меняется и вся программа может быть записана пятью командами:



Здесь по окончании цикла команда $k+5$ вновь возвращает к началу цикла (команда $k+5$ передает управление в ячейку $k+2$), но теперь уже в ячейке β находится не нуль, как это было при первом выполнении команды $k+2$, а единица. Следовательно, прибавление в β единицы образует в ней двойку, и при повторном выполнении программы будет вычислен 2^2 и т. д. Так же как и предыдущая, эта программа «зацикливается» и, чтобы избежать этого, следует снова воспользоваться операцией сравнения.

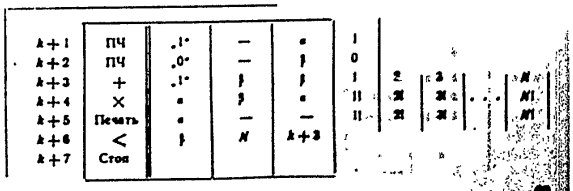


В конце каждого этапа счета число N сравнивается с содержимым ячейки β, т. е. с числом, квадрат которого уже вычислен. Пока это число меньше N, операция сравнения нас отсылает к приказу $k+2$, и мы вновь проходим весь цикл, но когда через N этапов в ячейке β образуется число N, операция сравнения пропустит и машина остановится. Так, с помощью шести команд составлена программа, по которой вычисляются квадраты любого нужного нам количества чисел, после чего машина останавливается (при этом число команд в программе от числа N не зависит, при изменении N изменяется лишь «константа сравнения» N).

Здесь опять многократно использована одна и та же группа команд (приказы $k+2 \div k+4$) за счет того, что при каждом новом прохождении этих команд число, хранящееся в ячейке β, изменяется, т. е. опять построена программа, содержащая многократно выполняемый цикл. Чтобы этот цикл был пройден нужное число раз, использована операция сравнения, в которой сравнивается содержимое ячейки β с числом N. В начале каждого цикла в ячейку β прибавляется 1, и, таким образом, в ячейке β фактически считается, сколько раз был выполнен весь цикл. На том же принципе построена программа по вычислению факториалов N последовательных чисел

$$1!,\ 2!,\ \ldots,\ N!$$

Так как $(n+1)!=n!(n+1)$, то достаточно последовательно образовывать в какой-нибудь ячейке α числа $1, 2, \ldots N$ и умножать на них уже полученные ранее факториалы,



3-3. Программы с автоматическим выбором числа циклов

В настоящем параграфе будет рассмотрен пример несколько более сложной программы, в которой число повторений цикла заранее неизвестно. Предположим, что нужно вычислить с заданной абсолютной погрешностью ε значение функции e^x для одного какого-нибудь значения аргумента x, пользуясь разложением в ряд

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots$$

В зависимости от величины x для получения заданной степени точности приходится вычислять различное число членов, причем заранее это число членов неизвестно. Для определения достигнутой при данном x точности вычислений нужно после прибавления очередного члена оценивать весь остаток, и если этот остаток больше ε, то вычислять еще один член и так до тех пор, пока остаток не станет меньше ε.

Если $|x| < 1$, то весь остаток

$$R_n = \frac{x^n}{(n+1)!} + \frac{x^{n+2}}{(n+2)!} + \ldots$$

по модулю меньше, чем последний неотброшенный член, т. е.

$$|R_n| < \frac{x^n}{n!}$$

Следовательно, надо прекратить процесс суммирования, как только последний вычисленный член станет меньше ε, а для этого надо сравнивать с ε все члены ряда последовательно. В приведенной ниже программе число x помещено в ячейку α, а сумма накапливается в ячейке β. Далее последовательные целые числа 1, 2, 3, ... образуются в ячейке γ_1, а n-ный член ряда $\frac{x^n}{n!}$ — в ячейке γ_4. Так как в ячейках β, γ_1, γ_3 могли остаться от решения предыдущих задач какие-нибудь числа, то либо они должны быть очищены передачей в них нуля (так надо поступить с ячейкой γ_1), либо в них можно передать числа, которые должны быть в них образованы на первом этапе счета (в ячейку β передается значение первого члена ряда, т. е. 1), либо, наконец, в них можно передать такое число, значение которого будет использовано в дальнейшем (в ячейку γ_3 передается 1). Во всяком случае каждая программа должна начинаться с команд, в которых „подготавливаются" ячейки, используемые в программе, но содержание которых к началу работы программы неизвестно. В дальнейшем мы не будем больше объяснять значение таких команд, „подготавливающих" ячейки. Программа вычисления e^x будет иметь вид (предполагается, что $|x| < 1$)

Программа вычисления e^x

$k+1$	ПЧ	,1•	—	β	1			
$k+2$	ПЧ	,0•	—	γ_1	0			
$k+3$	ПЧ	,1•	—	γ_3	1			
$k+4$	+	,1•	γ_1	γ_1	1	2		
$k+5$		γ_3	•	γ_3	x	$\frac{x^2}{1^2}$		
$k+6$		γ_3	γ_1	γ_3	$\frac{x}{1^1}$	$\frac{x^2}{2!}$		
$k+7$	+	γ_3	β	β	$1+\frac{x}{1!}$	$1+\frac{x}{1!}+\frac{x^2}{2!}$	...	e^x
$k+8$	|<|	,x•	γ_3	$k+4$				
$k+9$	Печать	β	—	—				
$k+10$	Стоп							
•	x							

Здесь команда $k+8$ сравнивает число ε с модулем очередного вычисленного члена $\frac{x^n}{n!}$ (ячейка γ_3) и, пока этот член больше ε, отправляет нас к началу цикла (в третьем адресе этой команды стоит ячейка $k+4$). Таким образом, путем сравнения заданного числа ε с числом $\frac{x^n}{n!}$, образуемом в процессе счета, машина сама определяет необходимое число повторений цикла. Из этого уже становится ясно, какую громадную роль играют операции условного перехода в цифровых машинах, позволяющие в нужный момент, заранее не известный, переключить счет с одного счета программы к другому.

Для лучшего уяснения структуры этой программы можно составить ее так называемую блок-схему или логическую схему (рис. 3-1).

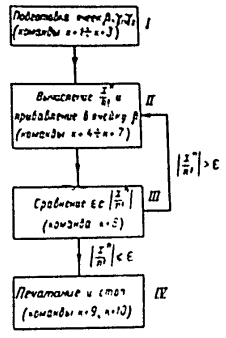


Рис. 3-1.

Совершенно аналогично, используя разложение

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots$$

составляется программа, вычисляющая значение функции $\sin x$ в точке x. Разница заключается в том, что здесь последующий член получается из предыдущего умножением на $-x^2$ и делением на $(n-1)(n+2)$.

Программа вычисления $\sin x$

$k+1$	ПЧ	•	—	β	x		
$k+2$	↘	•	•	γ_1	x^2		
$k+3$	−ПЧ	γ_1	—	γ_3	$-x^1$		
$k+4$	ПЧ	,0•	—	γ_2	0		
$k+5$	ПЧ	,1•	—	γ_3	1		
$k+6$	×	•	γ_3	•	$-x^3$,	$\frac{x^3}{3!}$,	...
$k+7$	+	,2•	γ_2	γ_2	2, 4, ...		
$k+8$	+	,2•	γ_3	γ_3	3, 5, ...		
$k+9$	:	•	γ_2	•	$-\frac{x^2}{2}$,	$\frac{x^4}{3!\,4}$,	...
$k+10$	:	•	γ_3	•	$-\frac{x^2}{3!}$,	$\frac{x^4}{5!}$,	...
$k+11$	+	•	β	β			$\sin x$
$k+12$	|<|	,x•	•	$k+6$			
$k+13$	Печать	β					
$k+14$	Стоп						
•	x						

Читатель без труда сам составит блок-схему этой программы.

При помощи преобразования вычислительных формул можно уменьшить число команд в программе. Разности отношений знаменателей двух последовательных членов ряда для синуса образуют арифметическую прогрессию. Действительно, если положить

$$\sin x = \frac{x}{c_1} - \frac{x^3}{c_2} + \frac{x^5}{c_3} - \frac{x^7}{c_4} + \ldots,$$

$$c_i = (2i-1)!$$

и обозначить

$$a_i = \frac{c_{i+1}}{c_i},$$

то

$$\Delta_i = a_{i+1} - a_i = \frac{c_{i+2}}{c_{i+1}} - \frac{c_{i+1}}{c_i} =$$

$$= \frac{(2i+3)!}{(2i+1)!} - \frac{(2i+1)!}{(2i-1)!} = (2i+3)(2i+2) - (2i+1)2i = 8i+6.$$

В силу этого закон образования коэффициентов $\frac{1}{c_1}, \frac{1}{c_2}, \frac{1}{c_3}, \ldots$ ряда будет следующим:

$$\frac{1}{c_{i+1}} = \frac{1}{c_i a_i};$$

$$a_i = a_{i-1} + \Delta_{i-1};$$

$$\Delta_i = 8i + 6,$$

$$i = 1, 2, 3, \ldots$$

Будем образовывать Δ_i в ячейке γ_3 и a_i — в ячейке γ_4. Так как $\Delta_0 = 6$, то вначале в ячейку γ_3 надо передать -2, а так как $\Delta_1 = a_1 = 6$, то $a_0 = 0$ и, следовательно, в ячейку γ_4 надо передать вначале 0. Программа принимает следующий вид

Программа вычисления $\sin x$

$k+1$	ПЧ	9	—	3	x	
$k+2$		3	3	γ_2	x^2	
$k+3$	$-$ПЧ	γ_2	—	γ_1	$-x^2$	
$k+4$	ПЧ	γ_3	—	γ_3	-2	
$k+5$	ПЧ	γ_4	—	γ_4	0	
$k+6$		3	γ_1	3	$-x^2$	$\frac{x^3}{c_2}$
$k+7$	+	γ_3	γ_3	γ_3	$\Delta_0 = 6$	$\Delta_1 = 14$
$k+8$	+	γ_4	γ_4	γ_4	$a_0 = 0$	$a_1 = 20$
$k+9$		3	γ_2	3	$\frac{-x^2}{c_1 a_1} = \frac{-x^2}{c_2}$	$\frac{x^3}{c_2 a_2} = \frac{x^3}{c_3}$
$k+10$	+	3	9	9	$\frac{x}{c_1} - \frac{x^3}{c_2}$	$\frac{x}{c_1} - \frac{x^3}{c_2} + \frac{x^5}{c_3} - \ldots \mid \sin x$
$k+11$	$1<1$	γ_5	8	$k+6$		
$k+12$	Печать	9	—	—		
$k+13$	Стоп					
8	x					

Программа сократилась на одну команду, но так как эта команда входит в многократно выполняемый цикл, то такое сокращение может дать существенный выигрыш, особенно для машин с невысокой скоростью работы.

3-4. Операция сложения команд

В § 1-7 при рассмотрении различных операций, которые выполняет цифровая вычислительная машина, была введена операция сложения команд. При этой операции складываются отдельно порядки чисел (в команде соответствующие разряды выражают код операции) и отдельно мантиссы чисел (соответствующие разряды выражают адреса команды) и после такого сложения нормализация не производится. При помощи этой операции удается преобразовать вид команд. Впредь, имея в виду операцию сложения команд, мы будем брать слова «прибавление» и «сумма» в кавычки.

Пусть в ячейку α помещена команда

α				
	*	γ_1	γ_2	γ_3

Если операции умножения присвоить код 10 0001 и принять

$$\gamma_1 = 0\ldots011;\ \gamma_2 = 0\ldots0111,$$

$$\gamma_3 = 0\ldots01,$$

то кодом этой команды будет число

100001	0 . . . 011	0 . . . 0111	0 . . . 01
	p	A	

Иными словами, в ячейке α находится число $2^q A$. „Прибавим" к этому числу при помощи операции сложения команд число

$$2^q B = \underbrace{000000}_{q}\ \underbrace{00\ldots01}_{B},$$

хранящееся в ячейке β, и запишем полученную „сумму" снова в ячейку α, т. е. выполним команду:

СК	α	β	α

Порядок полученного числа равен:

$$p + q = 100001,$$

а мантисса

$$\begin{array}{l} 0\ldots0110\ldots01110\ldots01 \\ \underline{0\ldots0000\ldots00000\ldots01} \\ 0\quad\ 0110\ldots01110\ldots10 \end{array}$$

причем это число не нормализуется. Таким образом, в ячейке α образовалось число

*	100001	0 . . . 011	0 . . . 0111	0 . . . 010

Код операции и первые два адреса в ячейке α не изменились, а третий адрес увеличился на единицу, т. е. в ячейке α теперь находится код команды:

*		γ_1	γ_2	γ_3+1

Итак, при „прибавлении" при помощи операции сложения команд к команде, хранящейся в ячейке α, нормализованного числа $2^q \cdot 2^{-36}$, изменяется вид этой команды. Число $2^q \cdot 2^{-36}$ для нашей условной машины можно назвать „единицей третьего адреса" (в нем стоит одна единица в младшем разряде третьего адреса) и обозначить $1/11A$. Если, как и раньше, через „a" обозначен номер той ячейки, в которой хранится число a, то выполненная операция сложения команд принимает вид:

СК	*	„1/11A"	α

Теперь ясно, почему после операции сложения команд нельзя выполнять нормализацию. Действительно, в рассмотренном случае пришлось бы сдвинуть мантиссу на десять разрядов налево и вычесть десять единиц из порядка, а это полностью изменило бы как код операции, так и адреса в команде.

Если „прибавить" к α число

$$2^q\, 2^{-24} = \underbrace{000000}_{6}\ \underbrace{0\ldots0}_{12}\ \underbrace{0\ldots01}_{12}\ \underbrace{0\ldots0}_{12},$$

называемое единицей второго адреса — $1/11A$ — (единица стоит в младшем разряде второго адреса), и записать сумму не в ячейку α, а в ячейку δ, т. е. выполнить команду

СК	*	„1/11A"	δ

то в ячейке δ образуется следующая команда:

δ	×	γ_1	γ_2+1	γ_3

Можно преобразовать один из адресов команды α не на одну единицу, а на

несколько, например, изменить первый адрес на 3 единицы, для чего к коду команды α надо „прибавить" число (оно обозначается $3/A$)

$$\underbrace{0\ \ 0}_{6}\ \underbrace{0\ \ 011}_{12}\ \underbrace{0\ \ 0}_{12}\ \underbrace{0\ldots0}_{12}$$

Можно видоизменить сразу два или три адреса. Например, можно „прибавить" к команде α одну единицу первого ($1/A$) и две единицы третьего адреса ($2/III/A$), т. е. число

$$\underbrace{0\ \ 0}_{6}\ \underbrace{0\ \ 01}_{12}\ \underbrace{0\ \ 0}_{12}\ \underbrace{0\ \ 010}_{12}$$

Тогда команда α примет вид

α + 1	$\gamma_1 + 1$	γ_2	$\gamma_3 + 2$

Таким образом, подбором соответствующего числа можно нужным образом видоизменять адреса или даже код операции любой команды.

3-5. Преобразование команд в программах

Во всех проведенных выше программах при каждом новом прохождении любого цикла изменялись числа, хранящиеся по некоторым адресам команд этого цикла, но вид самих команд при этом не изменялся. Другой способ циклического прохождения одной и той же группы команд заключается в том, что на отдельных этапах счета изменяется уже вид самих команд. Этот способ, использующий операцию сложения команд, сначала будет рассмотрен на элементарном примере. Пусть нужно перемножить на машине 50 чисел $x_1, x_2, \ldots, x_{50}$. Их можно поместить соответственно в ячейки $c+1, c+2, \ldots, c+50$. Для решения этой задачи легко составить программу, которая будет содержать 52 команды

$k+1$	ПЧ	„1"	—	*	1
$k+2$	×	$c+1$	*	*	x_1
$k+3$	×	$c+2$	*	*	x_1x_2
$k+4$	×	$c+3$	*	*	$x_1x_2x_3$
. . .	. . .	. . .	.	. . .	. . . . .
$k+51$	×	$c+50$	*	*	$x_1x_2 \ldots x_{50}$
$k+52$	Стоп				

При помощи операции сложения команд можно сократить эту программу. Пусть операции умножения присвоен код 000001; в качестве α взята ячейка с номером $\underbrace{100\ldots0}_{12}$ и в качестве c—нулевая ячейка $\underbrace{00\ldots0}_{12}$. Тогда коды команд $k+2$ и $k+3$ имеют вид:

$$k+2\ \ \underbrace{000001}_{6}\ \underbrace{00\ldots001}_{12}\ \underbrace{10\ldots0}_{12}\ \underbrace{10\ldots0}_{12}$$

$$k+3\ \ \underbrace{000001}_{6}\ \underbrace{00\ldots010}_{12}\ \underbrace{10\ldots0}_{12}\ \underbrace{10\ldots0}_{12}$$

Эти две команды, рассматриваемые как числа, отличаются друг от друга только тем, что в первом адресе у одной стоит число 1, а у другой—число 2 (записанные в двоичном коде). Иными словами, код команды $k+3$ больше кода команды $k+2$ на единицу первого адреса. Следовательно, код команды $k+3$ может быть получен при помощи операции сложения команд «прибавлением» $1/A$ к числу, хранящемуся в ячейке $k+2$ (т. е. к коду команды $k+2$)

СК	$k+2$	„$1/A$"	$k+2$

Теперь в ячейке $k+2$ стоит код команды $k+3$. Если «прибавить» к содержимому ячейки $k+2$ единицу первого адреса, т. е. еще раз выполнить только что выписанную команду, то получится код команды $k+4$ и т. д. Таким образом, «прибавляя» $1/A$ к содержимому ячейки $k+2$, можно последовательно образовать в ней все команды $k+3, k+4, \ldots, k+51$. Если после каждого такого преобразования передавать управление к ячейке $k+2$, то при помощи команды, которая в ней будет в этот момент находиться, последовательно образуются все произведения $x_1, x_1x_2, x_1x_2x_3, \ldots$ Теперь программа будет содержать только четыре команды:

$k+1$	ПЧ	„1"	—	*	1			
$k+2$	×	$c+1$	*	*	x_1	x_1x_2	$x_1x_2x_3$	. . .
$k+3$	СК	$k+2$	„$1/A$"	$k+2$				
$k+4$	ПУ	$k+2$	—	—				

Так как здесь в ячейке $k+4$ стоит безусловная передача управления, то произойдет уже знакомое нам «зацикливание» и, образовав произведение $x_1x_2, \ldots, x_{50}$, машина не остановится, а будет умножать это произведение на числа, хранящиеся в ячейках $c+51, c+52, \ldots$ и т. д. (в этих ячейках стоят числа, оставшиеся там от решения на машине предыдущей задачи). Чтобы предохранить программу от такого зацикливания, надо ввести операцию сравнения, которая будет подсчитывать число выполненных умножений и в нужный момент остановит машину. Окончательно программа будет иметь вид:

k	ПЧ	„0"	—	β	0				
$k+1$	ПЧ	„1"	—	*	1				
$k+2$	×	$c+1$	*	*	x_1	x_1x_2	.	x_1x_2	x_{50}
$k+3$	СК	$k+2$	„$1/A$"	$k+2$					
$k+4$	+	„1"	β	β	1	2		50	
$k+5$	≠	β	„50"	$k+2$					
$k+6$	Стоп								

Здесь, как и в предыдущих параграфах, в счетчике подсчитывалось число прохождений цикла и это число сравнивалось с числом 50. Но операцию сравнения можно реализовать и другим более компактным способом. К моменту, когда будут перемножены все 50 чисел $x_1x_2, \ldots, x_{50}$ команда $k+2$ будет иметь вид:

$k+2$	×	$c+51$	*	*

т. е. в ячейке $k+2$ будет стоять число (число 51 в двоичной системе имеет вид 110011):

$$\underbrace{000001}_{6}\ \underbrace{000000110011}_{12}\ \underbrace{10\ldots0}_{12}\ \underbrace{10\ldots0}_{12}$$

Вместо того, чтобы вводить счетчик β и сравнивать его содержимое с числом 50, мы можем сравнивать содержимое ячейки $k+2$ с написанным выше числом, и когда эти числа станут равны, операция сравнения пропустит и машина остановится. Иными словами, сравнивается код „переменной" команды

$k+2$	×	$c+i$	*	*

$i = 1, 2, \ldots, 51$

образующейся в ячейке $k+2$, с кодом „стандартной" команды

×	$c+51$	*	*

которая помещена, например, в ячейку $k+10$. Теперь эта программа будет иметь вид:

$k+1$	ПЧ	„1"	—	*
$k+2$	×	$c+1$	*	*
$k+3$	СК	$k+2$	„$1/A$"	$k+2$
$k+4$	≠	$k+2$	$k+10$	$k+2$
$k+5$	Стоп			
$k+10$	×	$c+51$	*	*

Declassified in Part - Sanitized Copy Approved for Release @ 50-Yr 2014/02/10 : CIA-RDP81-01043R003800160006-3

При помощи такого приема программа уменьшилась на две команды. Правда, одну ячейку $k+10$ нам пришлось занять под „стандартную" команду

Таким же способом можно составить программу, по которой подсчитывается число элементов последовательности $x_1, x_2, \ldots, x_{100}$, модуль которых равен или меньше некоторого положительного числа ε (как и выше предполагается, что число x_i помещено в ячейку $c+i$)

2

$k+1$	ПЧ	,0*	—	s
$k+2$	$\|\cdot\|$	,*	$c+1$	$k+4$
$k+3$	+	,1*	+	s
$k+4$	СК	$k+2$	,111*	$k+2$
$k+5$		$k+2$	$k+8$	$k+2$
$k+6$	Печать	+		—
$k+7$	Стоп			
$k+8$	$<$	,*	$c+101$	$k+4$

Здесь приказ $k+2$ сравнивает число ε с числом, хранящимся в ячейке $c+i$, т. е. с x_i. Если $\varepsilon \geq |x_i|$, то сравнение пропускает и в ячейку a прибавляется 1, если же $\varepsilon < |x_i|$, то сравнение нас отсылает к команде $k+4$.

В обоих случаях далее выполняется команда $k+4$, преобразующая команду $k+2$. Ко второму адресу команды $k+2$ прибавляется единица и, следовательно, адрес $c+i$ заменяется на $c+i+1$. После преобразования команды $k+2$ происходит сравнение кода команды $k+2$ с кодом стандартной команды, находящейся в ячейке $k+8$.

Так как код команды есть число положительное, то при прибавлении в ячейку единицы второго адреса это число увеличивается. Следовательно, пока во втором адресе команды $k+2$ будет стоять номер ячейки $c+i$, меньший, чем номер ячейки $c+101$, сравнение $k+5$ будет отсылать вновь к команде $k+2$.

После стократного выполнения команды $k+2$ в ячейке a будет получено требуемое число. При этом команда $k+2$ примет такой же вид, как и стандартная команда, хранящаяся в ячейке $k+8$. Поэтому команда сравнения $k+5$ пропустит, число, полученное в ячейке a, будет отпечатано, и машина остановится.

При помощи операции сложения команд можно построить программу, которая из n чисел $x_1, x_2, \ldots, x_n$ выбирает наименьшее (как и выше, элемент x_i помещен в ячейку $c+i$):

3

$k+1$	ПЧ	$c+1$	—	a
$k+2$	$<$	s	$c+2$	$k+4$
$k+3$	ПЧ	$c+2$	—	a
$k+4$	СК	$k+2$	,111*	$k+2$
$k+5$	СК	$k+3$	,11*	$k+3$
$k+6$	-	$k+3$	$k+8$	$k+2$
$k+7$	Стоп			
$k+8$	ПЧ	$c+n+1$	—	a

Здесь команда $k+1$ передает число x_1 в ячейку a. Затем x_1 сравнивается с числом x_2. Если $x_2 \leqslant x_1$, то команда сравнения пропустит и в ячейку a будет передано число x_2. Если же $x_2 > x_1$, то от команды $k+2$ управление перейдет к команде $k+1$ и в ячейке a останется число x_1. Следовательно, в ячейку a попадает меньшее из двух чисел x_1 и x_2. Далее команды $k+4$ и $k+5$ преобразуют команды $k+2$ и $k+3$ таким образом, что при повторном прохождении последних число, находящееся в ячейке a, сравнивается уже с x_3. Если элемент x_3 меньше или равен указанному числу, то он будет передан в ячейку a. Следовательно, в ячейке a образуется меньшее из трех чисел x_1, x_2, x_3. В дальнейшем в ячейке a получается меньшее из чисел $x_1, x_2, \ldots, x_n$.

Для прекращения процесса в нужный момент введена команда $k+6$, сравнивающая переменную команду $k+3$ со стандартной командой $k+8$. Для краткости мы говорим о сравнении команд, хотя на самом деле сравниваются не сами команды, а их коды.

Программа лишь немного усложнится, если поставить дополнительную задачу образовать в ячейке δ, в ее разрядах, соответствующих третьему адресу[1], адрес $c+i$ наименьшего элемента x_i. В предыдущей программе этот наименьший элемент помещается в ячейку a; чтобы выделить номер ячейки с наименьшим элементом x_i, воспользуемся операцией логического умножения. Команда $k+3$ передает элемент x_i из ячейки, в которой он хранится, в ячейку a. Если умножить логически команду $k+3$ на число

$$\underbrace{000000}_{6}\ \underbrace{11}_{12}\ \underbrace{1}\ \underbrace{00}_{12}\ \underbrace{0}\ \underbrace{00}_{12}\ \underbrace{0-11}\ ,111*$$

и поместить результат в ячейку δ, то на месте первого адреса в ячейке δ образуется адрес $c+i$ того элемента, который в этот момент передается в ячейку a. Остается сдвинуть содержимое ячейки δ на 21 разряда вправо. Этот сдвиг делается в конце программы, чтобы не повторять его каждый раз и не тратить зря время

4

$k+1$	ПЧ	$c+1$	—	a
$k+2$	$\wedge$	$k+1$	,11 ,111*	δ
$k+3$	ζ	a	$c+2$	$k+6$
$k+4$	ПЧ	$c+2$	—	a
$k+5$	$\wedge$	$k+4$	,11 ,11*	δ
$k+6$	СК	$k+3$	,111*	$k+3$
$k+7$	СК	$k+4$	,11*	$k+4$
$k+8$	$\leqslant$	$k+4$	$k+11$	$k+3$
$k+9$	→	δ	24	δ
$k+10$	Стоп			
$k+11$	ПЧ	$c+n+1$	—	a

Как легко видеть, если в последовательности $x_1, x_2, \ldots, x_n$ имеется несколько равных между собой наименьших элементов $x_i, x_j, \ldots, x_k$, то в ячейке δ образуется адрес элемента, имеющего наибольший индекс из $i, j, \ldots, k$.

3-6. Примеры более сложных программ

Рассмотрим пример, содержащий несколько циклов, выполняемых в программе попеременно. Пусть нужно вычислить значения функций $\mathrm{sh}\,x$ и $\mathrm{ch}\,x$ в некоторой точке x, пользуясь их разложением в ряд:

$$\mathrm{sh}\,x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots;$$

$$\mathrm{ch}\,x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots$$

Для этого можно построить программу, вычисляющую общий член $\frac{x^n}{n!}$ ($n = 0, 1, 2\ldots$) обоих рядов и поочередно прибавляющую члены с нечетной степенью в некоторую ячейку β_1, а члены с четной степенью — в ячейку β_2. Такая программа, во-первых, вычисляет одновременно значения обеих функций и, во-вторых, будет значительно короче, чем две отдельные программы, вычисляющие одна $\mathrm{sh}\,x$, а другая $\mathrm{ch}\,x$. Таким образом, $\mathrm{sh}\,x$ будет формироваться в ячейке β_1, а $\mathrm{ch}\,x$ — в ячейке β_2. В процессе вычислений мы в ячейке $\gamma+1$ будем образовывать величину n, а в ячейке $\gamma+2$ величину $\frac{x^n}{n!}$.

Чтобы осуществить попеременное прибавление членов $\frac{x^n}{n!}$ то в ячейку β_1, то в ячейку β_2, проще всего воспользоваться следующим приемом. В некоторую ячейку δ — будем называть ее управляющей ячейкой — предварительно помещается 1. Каждый раз после вычисления очередного числа $\frac{x^n}{n!}$ содержимое управляющей ячейки умножается на -1 и затем сравнивается с 0. Так как в ячейке δ попеременно образуется $+1$ и -1, то сравнение будет попеременно либо пропускать нас (в этом случае мы будем суммировать члены с нечетными степенями), либо отсылать к другому месту программы (там мы будем образовывать сумму членов с четными степенями).

Вычисления заканчиваются, как только следующий член ряда становится меньше или равным некоторой величине ε, определяющей точность вычислений.

На рис. 3-2 приведена схема программы. Здесь через (δ) обозначается число, хранящееся в ячейке δ.

[1] Т. е. в группе разрядов, в которых записывается третий адрес команды.

Программа вычисления sh x и ch x

I	$k+1$	ПЧ	,0•	—	γ_1	0, в γ_1 формируется sh x			
	$k+2$	ПЧ	,1•	—	γ_2	1, в γ_2 формируется ch x			
	$k+3$	ПЧ	,0•	—	$\gamma+1$	0, в $\gamma+1$ образуется n			
	$k+4$	ПЧ	,1•	—	$\gamma+2$	1, в $\gamma+2$ образуется $\frac{x^n}{n!}$			
	$k+5$	ПЧ	,1•	—	ξ	1, ξ — управляющая ячейка			
II	$k+6$	+	,1•	$\gamma+1$	$\gamma+1$	1	2	3	
	$k+7$	×	•	$\gamma+2$	$\gamma+2$	x	$\frac{x^2}{1}$	$\frac{x^3}{2!}$	
	$k+8$		$\gamma+2$	$\gamma+1$	$\gamma+2$	$\frac{x}{1}$	$\frac{x^2}{2!}$	$\frac{x^3}{3!}$	
	$k+9$	$\|<\|$	$\gamma+2$	,1•	$k+16$				
III	$k+10$	−	,−1•	ξ	ξ	−1	+1	−1	
	$k+11$	•	,0•	ξ	$k+14$				
IV	$k+12$	+	$\gamma+2$	γ_1	γ_1	$\frac{x}{1}$		$\frac{x}{1!}+\frac{x^3}{3!}$	... sh x
	$k+13$	ПУ	$k+6$	—	—		$1+\frac{x^2}{2!}$		ch x
V	$k+14$	+	$\gamma+2$	γ_2	γ_2				
	$k+15$	ПУ	$k+6$	—	—				
VI	$k+16$	Стоп							
	•	x							

В следующем примере показан один прием программирования, весьма часто употребляемый при решении различных задач. Пусть нужно составить программу, вычисляющую значения функции th x в n точках $x+h$, $x+2h$, ..., $x+nh$.

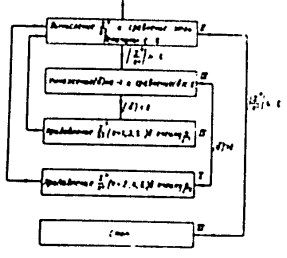


Рис. 3-2.

Так как

$$\text{th}\, x = \frac{\text{sh}\, x}{\text{ch}\, x},$$

то программа должна состоять из следующих отдельных этапов счета:

1. Образование очередного аргумента $x+ih$ и прибавление 1 в счетчик, определяющий число i табулированных значений аргумента.

2. Вычисление sh $(x+ih)$ и ch $(x+ih)$.

3. Деление друг на друга вычисленных значений sh $(x+ih)$ и ch $(x+ih)$, печатание результата и сравнение i с n.

4. Если $i \sim n$, то управление передается к началу программы, если же $i > n$, то машина останавливается. Таким образом, блок-схема программы будет иметь вид, представленный на рис. 3-3.

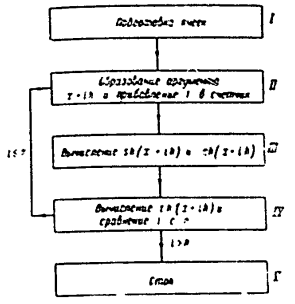


Рис. 3-3.

Если назвать оператором каждую из выделенных здесь групп команд I - V, то легко видеть, что оператор III полностью совпадает с уже имеющейся у нас программой вычисления sh x и ch x.

Условимся для краткости называть программу, вычисляющую значения th x, основной, а программу, вычисляющую sh x и ch x, — подпрограммой. При включении подпрограммы в основную программу должно быть предусмотрено следующее.

Во-первых, при составлении основной программы нужно учитывать распределение ячеек в подпрограмме, т. е. учитывать, в каких ячейках хранится сама подпрограмма, в каких ячейках хранится аргумент x (ячейка 2), образуются sh x и ch x (ячейки β_1 и β_2 и т. п.

Во-вторых, в конце оператора II нужно ввести команду, переключающую счет к началу подпрограммы, т. е. к ячейке $k+1$. Наконец, в-третьих, должно быть предусмотрено переключение счета от конца подпрограммы к началу оператора IV. Если подпрограмма уже введена в запоминающее устройство машины, то подготовку такого переключения проще всего выполнить автоматически, поручив ее основной программе.

Для этого перед командой, передающей управление от оператора II к подпрограмме, ставится команда, которая заменяет последнюю команду подпрограммы (операция „Стоп" в ячейке $k+16$) на команду, возвращающую управление к прерванному месту основной программы. Такая команда, возвращающая счет, должна быть предварительно составлена и помещена в запоминающее устройство вместе с основной программой. Теперь последняя примет следующий вид (основная программа помещена в ячейки с номерами $p+1$, $p+2$, ..., а подпрограмма — в ячейки с номерами $k+1$, $k+2$, ...):

I	$p+1$	ПЧ	,0•	—	$\gamma+3$	В $\gamma+3$ помещен счетчик числа i табулированных точек
	$p+2$	ПЧ	,x•	—	α	Подготовка ячейки α, в которой формируется аргумент $x+ih$:
II	$p+3$	+	,1•	$\gamma+3$	$\gamma+3$	1 | 2 | ... | n
	$p+4$	+	,h•	α	α	$x+h$ | $x+2h$ | ... | $x+nh$

Следующая команда должна передать управление в ячейку $k+1$ подпрограммы, но предварительно в ячейку $k+16$ передается содержимое ячейки ξ, в которой должна находиться команда, возвращающая счет к началу оператора III. Эту команду можно выписать лишь после того, как будет составлена основная программа. Остальная часть программы имеет следующий вид:

Вместо III	$p+5$	ПЧ	ξ	-	$k+16$	Управление передано к началу подпрограммы
	$p+6$	ПУ	$k+1$	—	—	
...	$p+7$		β_1	β_1	β_1	
	$p+8$	Печать	β_1	—	—	
	$p+9$	<	$\tau+3$	α^*	$p+3$	
...	$p+10$	Стоп				
ξ		ПУ	$p+7$	—	—	Команда, передающая управление от конца подпрограммы к основной программе

Теперь ясно, что управление от подпрограммы должно передаваться к ячейке $p+7$. Именно такая команда помещается в ячейку ξ. Каждый раз при переходе к новой точке содержимое ячейки ξ вновь передается в ячейку $k+16$. Между тем это достаточно выполнить только 1 раз, для чего можно поручить местачи команды $p+3$ и $p+5$, а в команде $p+8$ вместо $p+3$ в третьем адресе поставить $p+4$. Составленная программа верна, но несколько расточительна с точки зрения времени работы. Команда, передающая содержимое ячейки ξ в ячейку $k+16$, изрочно поставлена непосредственно перед командой, передающей управление к подпрограмме, так как такое расположение команд бывает необходимым в случае попеременного обращения к какой-нибудь подпрограмме из различных мест основной программы. Рассмотренный в этом примере метод, при котором в составляемую программу включены уже имеющиеся у нас программы, называется методом стандартных подпрограмм и будет рассмотрен подробнее в следующей главе.

3-7. Примеры программ для машины М-3

В § 2-11 были приведены основные характеристики универсальной цифровой вычислительной машины М-3, а также структура кодов, чисел и команд для этой машины (рис. 2-42). М-3 является машиной с фиксированной запятой, которая может оперировать лишь с числами, по модулю меньшими единицы. Иными словами, порядок каждого числа принимается равным нулю, а потому вообще не изображается в машине.

Если в результате арифметических действий образуется число, по модулю большее или равное единице, то происходит переполнение разрядной сетки и машина останавливается. Поэтому в машине числа, по модулю превосходящие единицу, должны изображаться с некоторым масштабом. О выборе масштабных коэффициентов будет сказано ниже.

Напомним кратко некоторые особенности кода этой машины. Перечень команд машины М-3 приведен в приложении 1. Машина М-3 является двухадресной машиной, команды которой имеют вид:

Код операции	I адрес	II адрес

По первому адресу команды хранятся слагаемое, вычитаемое, множитель и делитель, а по второму — слагаемое, уменьшаемое, множимое и делимое. Результат каждого действия (кроме действий с модулями чисел) может быть без дополнительной команды записан в запоминающее устройство (по второму адресу команды) и отпечатан.

Если результат действия записывается по второму адресу, то число, стоявшее там, «забивается». Если оно нужно в последующих вычислениях, то его надо предварительно «запасти», т. е. передать в другую ячейку памяти.

В машине М-3 результат каждого действия сохраняется на регистре в арифметическом узле к началу следующей операции. Благодаря этому сохраненный на регистре результат предыдущего действия может быть использован в последующей операции аналогично числу, стоящему по второму адресу команды, т. е. играть роль слагаемого, уменьшаемого, множимого и делимого. В этом случае команда, выполняемая машиной, становится, по существу, одноадресной. Используя в качестве одного из чисел результат предыдущего действия, мы уменьшаем число обращений к запоминающему устройству (магнитному барабану) и тем самым сокращаем время, необходимое на выполнение команды.

Результат операции, использующей результат предыдущего действия, может быть, в свою очередь, либо записан по второму адресу, либо — нет. В последнем случае во второй адрес команды ставится нули.

Приводя примеры программ для машины М-3, будем пользоваться не числовыми кодами операций, а их условным обозначением (см. приложение 1). При этом, если результат действия не записывается в запоминающее устройство, то после символического знака операции ставится запятая, например +,. Если в команде используется результат предыдущего действия, сохраненный на регистре арифметического узла, то перед символом операции ставится стрелка, например ↓ ×.

В машине М-3 каждое арифметическое действие может производиться с модулями чисел, но результат такого действия не может быть без дополнительной команды записан в запоминающее устройство. Например, команда

|—|,	α	β

означает, что из модуля числа, хранящегося в ячейке β, вычтен модуль числа, хранящегося в ячейке α, но результат в память не записывается. В равной мере команда

↓|—|,	α	—

означает, что из модуля результата предыдущего действия вычтен модуль числа, хранящегося в ячейке α, но в память разность не записывается.

Операция условного перехода осуществляется в машине М-3 по знаку результата предыдущего действия и записывается

$c+1$	...	...	...
$c+2$	УП	α	β

означает, что если знак предыдущего действия (результат выполнения команды $c+1$) имеет знак минус, то следующей после команды $c+2$ выполняется команда, находящаяся в ячейке α. Если же результат имеет знак плюс, то следующей выполняется команда из ячейки β. Отметим, что после выполнения команды УП на регистре арифметического узла остается результат действия, предшествующего этой команде.

Условная передача управления в зависимости от результата сравнения двух чисел требует в машине М-3 двух команд:

|—|,	$\cdot p^*$	$\cdot q^*$
УП	α	β

Если $|\cdot q^*| < |\cdot p^*|$, то следующей будет выполнена команда из ячейки α, если $|\cdot q^*| \geqslant |\cdot p^*|$, то команда из ячейки β.

Для алгебраического сравнения чисел (сравнения с учетом знаков) операции условного перехода должна предшествовать обычная операция вычитания. Заметим, что если $a > 0$, то разность

$$(-a) - (-a) = -0$$

в машине М-3 будет иметь отрицательный знак. Поэтому, если при сравнении чисел неизвестны их знаки, то для выяснения равенства чисел a и b необходимо проверить знаки обеих разностей $a - b$ и $b - a$. Это важно помнить, так как операция условного перехода проверяет один лишь знак результата предыдущего действия.

В машине М-3 нет отдельной операции печати, так как каждая опера-

ция, при которой результат действия записывается по второму адресу, может сопровождаться отпечатыванием этого результата. Для обозначения операции с печатью мы после символа операции будем ставить букву П. Так, например, команда

| +П | α | | β |

означает, что производится сложение чисел, расположенных в ячейках α и β, сумма записывается в ячейку β и, кроме того, печатается.

Ввиду того, что М-3 является машиной с фиксированной запятой и все арифметические операции в ней выполняются без нормализации, то для преобразования команд нет нужды вводить специальную операцию, и преобразование команд может выполняться при помощи обычных операций сложения и вычитания. Например при прибавлении к какой-нибудь команде числа

$$11\ 11-0\ 0\ \underbrace{0\ 0}_{6}\ \underbrace{0\ 0}_{12}\ \underbrace{0\ 1\ 0\ 0}_{12}=2^{-11}+2^{-12}$$

оба адреса этой команды увеличиваются на единицу.

Рассмотрим в качестве примера программу извлечения квадратного корня из числа x для машины М-3. Уравнение

$$y=\sqrt{x}$$

можно записать в виде $f(x, y)=y^2-x=0$ и решать его, пользуясь итерационной формулой Ньютона (см. гл. 9)

$$y_{n+1}=y_n-\frac{f(x, y_n)}{f'_y(x, y_n)},$$

которая приводит к следующему рекуррентному соотношению.

$$y_{n+1}=\frac{1}{2}\left(y_n+\frac{x}{y_n}\right)$$

для определения корня квадратного из числа x.

В приведенных формулах через y_n и y_{n+1} обозначены n-ное и $n+1$-е приближенные значения величины y.

Легко доказать, что если

$$|y_{n+1}-y_n|<\varepsilon,$$

то и

$$|y-y_{n+1}|<\varepsilon.$$

Так как

$$y_{i+1}-y_i=\frac{1}{2}\left(\frac{x}{y_i}-y_i\right)=$$

$$=R(x, y_i),$$

$$y_{i+1}=y_i+R(x, y_i),$$

то достаточно составить программу, вычисляющую величину $R(x, y_i)$ и сравнивающую ее модуль с ε. Если $|R|>\varepsilon$, то $R(x, y_i)$ надо прибавить к y_i и перейти к вычислению $R(x, y_{i+1})$, если же $|R|<\varepsilon$, то вычисления можно прекратить. Для универсальности программы нужно в качестве нулевого приближения y_0 взять максимальное число, которое может быть изображено на машине М-3, чтобы отношение $\frac{x}{y_0}$, образуемое на первом этапе счета, не вышло за разряды. Вместе с тем, если x мало, то такой выбор y_0 потребует большого числа итераций. На практике для сокращения времени вычислений выбор нулевого приближения y_0 надо соразмерять с величиной числа x. Кроме того, должно выполняться условие $x<y_i$, которое легко допустимо, учитывая, что $\sqrt{x}>x$ при $x<1$. Программа извлечения квадратного корня будет иметь следующий вид:

Извлечение квадратного корня

$k+1$	ПЧ	γ_0^*	β	y_0	
$k+2$	:	β	α	$\frac{x}{y_0}$	$\frac{x}{y_1}$
$k+3$	4−	β	—	$\frac{x}{y_0}-y_0$	$\frac{x}{y_1}-y_1$
$k+4$	4×	$\frac{1}{2}^*$	γ	$\frac{1}{2}\left(\frac{x}{y_0}-y_0\right)$	$\frac{1}{2}\left(\frac{x}{y_1}-y_1\right)$
$k+5$	4+	β	β	$y_0+\frac{1}{2}\left(\frac{x}{y_0}-y_0\right)-y_1$	$y_1+\frac{1}{2}\left(\frac{x}{y_1}-y_1\right)-y_2$
$k+6$	1−1.	γ	ε^*		
$k+7$	УП	$k+2$	$k+8$		
$k+8$	4п	0^*	β		
$k+9$	Стоп				
?	x				

Здесь в ячейке α хранится число x, в ячейке γ образуется выражение $R(x, y_i)$ и в ячейке β накапливается ответ.

В командах $k+2$, $k+3$ и $k+6$ результат действия не записывается. По этому числа, стоящие по второму адресу этих команд, не забиваются и используются при следующем прохождении цикла. Отметим также, что результат команды $k+4$ записывается по второму адресу в ячейку γ, так как он нужен для сравнения с ε (команда $k+6$). Но при прибавлении в следующей операции (команда $k+5$) в ячейку β числа $\frac{1}{2}\left(\frac{x}{y_i}-y_i\right)$ последнее для сокращения времени берется не из ячейки γ, а с регистра, где оно сохранилось.

При составлении приведенной выше программы извлечения квадратного корня из числа x предполагалось, что $x<1$. В силу этого все числа, получающиеся в процессе счета, также меньше 1 и, следовательно, не выходят за допустимые на М-3 пределы. В следующем примере будут фигурировать числа, по модулю бо́льшие единицы. Составим программу, вычисляющую значение функции e^x для некоторого значения x, $|x|<1$, пользуясь разложением в ряд

$$e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\ldots$$

Здесь уже $e^x>1$. В § 3-3 n-ный член этого ряда получался из предыдущего делением его на n и умножением на x. Само же число n получалось из числа $n-1$ прибавлением к нему единицы. Но в данном случае действовать так уже нельзя, и поэтому надо поступить следующим образом. Вместо единицы в машине хранится число $\frac{1}{M}$, где M—достаточно большой масштабный коэффициент, и вместо чисел 1, 2, 3,... образуются последовательно друг из друга числа $\frac{1}{M}$, $\frac{2}{M}$, $\frac{3}{M}$,... прибавлением $\frac{1}{M}$. Член ряда $\frac{x^n}{n!}$ получается из предыдущего $\frac{x^{n-1}}{(n-1)!}$ умножением его на $\frac{x}{M}$ и делением на $\frac{n}{M}$. Таким образом, все числа, получающиеся в процессе счета, оказываются по модулю меньше единицы. Программа имеет следующий вид:

Вычисление e^x, $|x|<1$

$k+1$	ПЧ	$.1/M^*$	β
$k+2$	ПЧ	$.1\cdot M^*$	$\gamma+1$
$k+3$	ПЧ	$.0^*$	$\gamma+2$
$k+4$	/	α	$\gamma+1$
$k+5$	+	$.1/M^*$	$\gamma+2$
$k+6$	·	$\gamma+2$	$\gamma+1$
$k+7$	·	$.1/M^*$	$\gamma+1$
$k+8$	+	β	β
$k+9$	−	$.\alpha/M^*$	$\gamma+1$
$k+10$	УП	$k+11$	$k+4$
$k+11$	+П	$.0^*$	β
$k+12$	Стоп		
α	x		

Программа проста и не требует специальных пояснений. Заметим только, что в команде $k+7$ число $\frac{x^n}{n!}$ умножается на $1/M$, чтобы затем эти члены можно было суммировать в ячейке β, в которой образуется $e^x \cdot M$.

Команда $k+11$ осуществляет печатание результата, т. е. числа $e^x \cdot M$. Так как в машине М-3 нет отдельной операции печати, то в команде $k+11$ к содержимому ячейки β прибавляется нуль и отпечатывается полученная сумма.

Теперь будет рассмотрена программа, в которой при повторном прохождении цикла преобразуются уже сами команды, а не содержимое ячеек, как это было в двух предыдущих случаях. Предположим, что нужно вычислить значение полинома

$$P(x)=a_n x^n+a_{n-1}x^{n-1}+\ \ +a_1 x+a_0$$

для некоторого значения аргумента x. Чтобы упростить программу, можно воспользоваться схемой вычислений Горнера, по которой полином записывается в следующем виде:

$$P(x)=\{\ [(a_n x+a_{n-1})x+ \\ +a_{n-2}]x+\ \ +a_1\}\,x+a_0.$$

Теперь вычисление полинома сводится к последовательному прибавлению в некоторую ячейку β коэффициентов полинома (начиная со старшего) и умножению содержимого β на x. Чтобы ввести в машину коэффициенты a_i и аргумент x, следует разделить их все на такое число M, чтобы $\left|\frac{x}{M}\right|<1$, $\left|\frac{P(x)}{M^n}\right|<1$, $\left|\frac{a_i}{M}\right|<1$, а также все промежуточные результаты вычислений были меньше единицы.

Аргумент $\frac{x}{M}$ помещается в ячейку α, коэффициенты a_n/M, a_{n-1}/M, ..., a_0/M в ячейки $p+0$, $p+1$, ..., $p+n$ соответственно, а ответ накапливается в ячейке β.

Вычисление полинома

$k+1$	ПЧ	$p+0$	β	$\frac{a_n}{M}$
$k+2$	×	α	β	$\frac{a_n x}{M^2}$ $\quad (a_n x+a_{n-1})\frac{x}{M^2}$
$k+3$	:	$\frac{1}{M}$	−	$\frac{a_n x}{M}$ $\quad (a_n x+a_{n-1})\frac{x}{M}$
$k+4$	↓+	$p+1$	β	$\frac{a_n x}{M}+\frac{a_{n-1}}{M}$ $\quad [(a_n x+a_{n-1})x+a_{n-2}]\frac{1}{M}$ $\quad \cdots$ $\quad P(x)$
$k+5$	+	$.1/M^*$	$k+4$	
$k+6$	|−|	$k+4$	γ	
$k+7$	УП	$k+8$	$k+2$	
$k+8$	+П	$.0^*$	β	
$k+9$	Стоп			

Печать $P(x)$

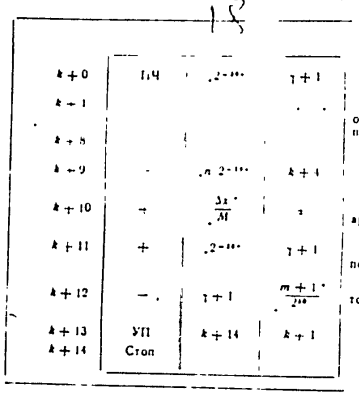
γ	↓ +	$p+n$	β
α	$\frac{x}{M}$		
$p+0$	$\frac{a_n}{M}$		
$p+1$	$\frac{a_{n-1}}{M}$		
...	...		
$p+n$	$\frac{a_0}{M}$		

Здесь команда $k+5$ последовательно образует в первом адресе команды $k+4$ номера ячеек $p+2$, $p+3$, ... Следовательно, команда $k+4$ будет последовательно прибавлять в ячейку β коэффициенты a_{n-1}/M, a_{n-2}/M, (коэффициент a_n/M предварительно передается в ячейку β командой $k+1$). В команде $k+3$ результат предыдущего действия делится на $1/M$, чтобы множитель $1/M$ входил в первой степени во все слагаемые, накапливаемые в ячейке β. Управление счетом осуществляется сравнением переменной команды $k+4$ со стандартной командой, записанной в ячейке γ. Когда в первом адресе команды $k+4$ образуется номер ячейки $p+n+1$, тогда разность $(\gamma)-(k+4)$ становится отрицательной, управление передается в ячейку $k+8$ и машина останавливается[1]. Счет прекращается только тогда, когда в первом адресе команды $k+4$ стоит номер ячейки $p+n+1$, хотя последний коэффициент a_0 хранится в ячейке $p+n$. Это связано с тем, что команда $k+4$ сначала выполняется, а потом уже преобразуется и, следовательно, когда в ее первом адресе стоит номер $p+n+1$, она не выполняется. Следует специально обратить внимание на этот факт, поскольку он часто вызывает недоразумения и ошибки.

Можно усложнить несколько задачу и составить программу, вычисляющую значения полинома $P(x)$ в $(m+1)$-ой точках $x_0, x_1, \ldots, x_m$, отстоящих друг от друга на одну и ту же величину Δx. Для этого достаточно $m+1$ раз выполнить уже составленную выше программу, каждый раз увеличивая аргумент на величину $\Delta x/M$. Чтобы сосчитать число пройденных точек $x_1, x_2, \ldots$, нужно прибавлять в счетчик не единицу, как в § 3-6 (сейчас этого сделать нельзя), а единицу последнего

[1] Напомним, что символом (α) обозначается число, хранящееся в ячейке α.

разряда, т. е. $1/2^{m}$, и сравнивать содержимое счетчика с числом $\frac{m+1}{2^{m}}$. Единица последнего разряда взята здесь лишь для определенности. С равным успехом можно прибавлять в счетчик единицу *любого* i-того разряда, т. е. число $1/2^{i}$, и сравнивать с числом $\frac{m+1}{2^{i}}$, если только последнее число не выходит за разрядную сетку. Следует сделать одно важное замечание. Как



$k+0$	ЦЧ	$\cdot 2^{-m}$	$\gamma+1$	Подготовка счетчика
$k+1$				Эти команды не выписаны, так как они такие же, как в предыдущей программе
$k+8$				
$k+9$	−	$\cdot n\,2^{-m}$	$k+4$	Восстановление команды
$k+10$	+	$\frac{3x}{M}$	γ	Переход к следующему значению аргумента
$k+11$	+	$\cdot 2^{-m}$	$\gamma+1$	Прибавление в счетчик единицы последнего разряда
$k+12$	−	$\gamma+1$	$\frac{m+1}{2^{m}}$	Проверка числа табулированных точек
$k+13$	УП	$k+14$	$k+1$	
$k+14$	Стоп			

отмечалось выше, к моменту, когда вычислено значение полинома $P(x)$ в точке x, в первом адресе команды $k+4$ стоит номер $p+n+1$. Но чтобы по этой программе можно было приступить к вычислению полинома в следующей точке x, в первом адресе команды $k+4$ должен вновь стоять номер $p+1$. Иными словами, перед тем как перейти к вычислению полинома в новой точке, необходимо восстановить первоначальный вид переменной команды $k+4$. Сделать это можно двумя различными способами. Первый заключается в том, что из команды $k+4$ вычитается число, прибавленное к ней в процессе работы программ. При вычислении значения полинома в одной точке к первому адресу команды $k+4$ была n раз прибавлена единица первого адреса и, следовательно, всего прибавлено число $n\cdot\frac{1}{2^{m}}$. Итак, чтобы восстановить первоначальный вид команды $k-4$, достаточно выполнить команду

−	$\cdot n\,\frac{1}{2^{m}}$	$k+4$

Программа, использующая этот способ, будет иметь вид

$k+0$	ЦЧ	$\cdot 2^{-m}$	$\gamma+1$	Подготовка счетчика
$k+1$				Эти команды не выписаны, так как они такие же, как в предыдущей программе
$k+8$				
$k+9$	−	$\cdot n\,2^{-m}$	$k+4$	Восстановление команды
$k+10$	+	$\frac{3x}{M}$	γ	Переход к следующему значению аргумента
$k+11$	+	$\cdot 2^{-m}$	$\gamma+1$	Прибавление в счетчик единицы последнего разряда
$k+12$	−	$\gamma+1$	$\frac{m+1}{2^{m}}$	Проверка числа табулированных точек
$k+13$	УП	$k+14$	$k+1$	
$k+14$	Стоп			

Другой способ восстановления переменной команды заключается в следующем. В некоторой ячейке $\gamma+2$ выписывается первоначальный вид команды $k+4$

$\gamma+2$	+	$p+1$	4

Теперь можно восстановить команду $k+4$, просто передав в ячейку $k+4$ содержимое ячейки $\gamma+2$. Следовательно, программа, использующая этот метод, будет такой же, как и только что написанная, но только команда $k+9$ будет теперь иметь вид:

$k+9$	ПЧ	$\gamma+2$	$k+4$

Впрочем выгоднее поставить эту команду не после прохождения цикла, а до его выполнения. Мы настолько так подробно останавливаемся на восстановлении „испорченной" в процессе счета команды, так как этот вопрос играет весьма важную роль при программировании конкретных математических задач. К этому вопросу мы еще вернемся в следующей главе.

3-8. Перевод чисел из десятичной системы счисления в двоичную и наоборот

Рассмотрим метод перевода чисел из десятичной системы счисления в двоичную и наоборот, применяемый в машине М-3.

Пусть дано положительное десятичное число

$$A=0,\ a_1a_2,\ \ldots,\ a_n,$$

меньшее единицы. Здесь a_i — десятичная цифра, принимающая одно из значений 0, 1, 2, ..., 9. Как известно,

$$A=\frac{a_1}{10^1}+\frac{a_2}{10^2}+\ldots+\frac{a_n}{10^n}.$$

Представим каждую из цифр a_i в виде четырехразрядного двоичного числа $\beta_i^{\mathrm{I}}\beta_i^{\mathrm{II}}\beta_i^{\mathrm{III}}\beta_i^{\mathrm{IV}}$. При этом получается двоичное $4n$-разрядное число

$$A'=0,\ \underbrace{\beta_1^{\mathrm{I}}\beta_1^{\mathrm{II}}\beta_1^{\mathrm{III}}\beta_1^{\mathrm{IV}}\beta_2^{\mathrm{I}}\beta_2^{\mathrm{II}}\beta_2^{\mathrm{III}}\beta_2^{\mathrm{IV}}\ldots\beta_n^{\mathrm{I}}\beta_n^{\mathrm{II}}\beta_n^{\mathrm{III}}\beta_n^{\mathrm{IV}}}_{4n}.$$

Если, например, число $A=0{,}38109$, то двоичное число A' будет иметь вид

$$A'=0,\ \underbrace{0011}_{3}\ \underbrace{1000}_{8}\ \underbrace{0001}_{1}\ \underbrace{0000}_{0}\ \underbrace{1001}_{9}$$

Число A' называется двоично-десятичным кодом числа A и, разумеется, не равно ему, но с его помощью можно найти двоичную запись числа A.

Двоичное число A' равно:

$$A'=\frac{a_1}{16}+\frac{a_2}{16^2}+\ldots+\frac{a_n}{16^n}.$$

Чтобы получить двоичную запись числа A, достаточно умножить $\frac{a_i}{16^i}$ на $\frac{16^i}{10^i}$ и сложить полученные произведения для $i=1,\ 2,\ \ldots,\ n$. При этом, конечно, числа $\frac{16^i}{10^i}$ должны быть записаны в двоичном виде, а все действия производиться в двоичной системе счисления.

Таким образом, получается число

$$(17)\qquad A=\sum_{i=1}^{n}\frac{a_i}{16^i}\cdot\frac{16^i}{10^i}=\sum_{i=1}^{n}\frac{a_i}{10^i},$$

но записанное в двоичной системе счисления.

Чтобы из двоичного кода A' выделить число $\frac{a_i}{16^i}$, достаточно логически умножить A' на $4n$-разрядное двоичное число

$$p=0,\ \underbrace{00\ldots01111}_{4i}\ 00\ldots00$$

Так как приводимая ниже программа составлена для машины М-3, то в десятичном числе $A=0,\ a_1a_2,\ \ldots,\ a_n$ нужно удержать семь разрядов. Действительно двоично-десятичный код числа A содержит $4n$ разрядов, а ячейка запоминающего устройства машины М-3 содержит 30 разрядов. Если число A содержит меньше разрядов, то недостающие разряды заменяются нулями. Например, двоично-десятичный код рассмотренного выше числа $A=0{,}38109$ имеет вид:

$$0,\ \underbrace{0011}_{3}\ \underbrace{1000}_{8}\ \underbrace{0001}_{1}\ \underbrace{0000}_{0}\ \underbrace{1001}_{9}\ \underbrace{0000}_{0}\ \underbrace{0000}_{0}\ 00$$

Фактически в программе сначала выделяется $\frac{a_i}{16^i}$, для чего число A' умножается логически на число

$$p=0,\ \underbrace{00\ldots01111}_{28}\ 00$$

у которого в разрядах 25—28 стоят единицы, а в остальных разрядах стоят нули. Затем $\frac{a_i}{16^i}$ делится на число $\frac{10}{16^i}$ $\left(\text{умножать на }\frac{16^i}{10}\text{ мы не можем, так как }\frac{16^i}{10}>1\right)$. Таким образом, получается число $\frac{a_i}{10}$. Чтобы выделить разряды, изображающие цифру a_i, достаточно сдвинуть число A' направо на четыре разряда и вновь умножить на число p. Так как в машине М-3 нет операции сдвига, ее заменяют умножением на 1/16. Таким образом, получается число $\frac{a_i}{16^i}$. Если разделить его

на $\frac{10}{16}$, то получается число $\frac{a_1}{10}$. Остается прибавить его к $\frac{a_2}{10}$, предварительно умножив последнее на $\frac{1}{10}$.

Повторив так еще 5 раз, можно окончательно получить число A, записанное в двоичной системе счисления. Ввиду простоты программы ее схема не приводится

Программа перевода чисел из десятичной системы счисления в двоичную на машине М-3

Номер ячейки	Операция	α	β	Примечание
$k+1$	ПЧ	$a+9$	$a+2$	Очистка ячейки $a+2$, в которой накапливается ответ
$k+2$	ПЧ	$a+9$	$a+4$	Очистка счетчика
$k+3$	$\times$	$a+5$	$a+2$	Умножение $\frac{a_n}{10^{n-1}}+\frac{a_{n-1}}{10^{n-1-1}}+$ $+\ \ +\frac{a_{i+1}}{10}$ на $\frac{1}{10}$ (на первом этапе эта операция лишняя)
$k+4$	$\wedge$	$a+1$	$a+3$	Выделение $\frac{a_i}{10^i}$
$k+5$	$+$	$a+8$	—	Образование $\frac{a_i}{10}$
$k+6$	$+$	$a+2$	$a+2$	Накапливание ответа
$k+7$		$a+6$	$a+1$	Сдвиг переводимого числа на четыре разряда вправо
$k+8$	$+$	$a+7$	$a+4$	Прибавление единицы в счетчик
$k+9$	$+-$	$a+10$		Содержимое счетчика сравнивается с n
$k+10$	УП	$k+4$		
$a+1$	Двоично-десятичный код переводимого числа			
$a+2$	(Краткая) двоичная запись переведенного числа			
$a+3$	$p=0,\underbrace{00\ldots0}_{24}111100$			
$a+4$	Счетчик			
$a+5$	$\frac{1}{10}=0,0001100110011001100110011001$			
$a+6$	$\frac{1}{16}=0,000100\ldots0$			
$a+7$	$\frac{2}{16}=0,00100\ldots0$			
$a+8$	$\frac{10}{16}=0,\underbrace{00\ldots0}_{26}1010000$			
$a+9$	$0=0,00\ldots0$			
$a+10$	$\frac{13}{16}=0,110100\ldots0$			

Для реализации этой программы двоично-десятичный код числа A пробивается на ленту (см. § 2-8) и вводится в ячейку $a+1$. Во второй адрес команды $k+10$ нужно поставить адрес той ячейки, в которую должно быть передано управление после окончания перевода. Двоичная запись числа образуется в ячейке $a+2$.

Пусть, наоборот, имеется положительное двоичное число

$$A=0,\beta_1\beta_2\ldots\beta_m$$

и нужно получить его десятичную запись. Число A в десятичной системе счисления имеет вид

$$A=0,a_1a_2\ldots a_n=\frac{a_1}{10}+\frac{a_2}{10^2}+\ldots+\frac{a_n}{10^n},$$

где a_i пока неизвестны и должны быть определены. Для этого число A умножается на $\frac{10}{16}$ и получается число

$$\tilde{A}=\frac{a_1}{16}+\left(\frac{a_2}{10}+\frac{a_3}{10^2}+\ldots+\frac{a_n}{10^{n-1}}\right)\frac{1}{16}.$$

Так как число, стоящее в скобках, меньше единицы, то второе слагаемое размещено в разрядах с 5-го по n-ный, а $\frac{a_1}{16}$ находится в первых четырех разрядах числа A. Если выделить эти разряды, то определится двоичная запись цифры a_1 первого десятичного разряда числа A. Чтобы найти разряд a_2, надо вычесть число $\frac{a_1}{16}$ из числа $\tilde{A}$ и получить число

$$\tilde{A}=\left(\frac{a_2}{10}+\frac{a_3}{10^2}+\ldots+\frac{a_n}{10^{n-1}}\right)\frac{1}{16}.$$

Умножив его на $\frac{10}{16}$, мы получим:

$$\tilde{A}=\frac{a_2}{10^2}+\left(\frac{a_3}{10^3}+\ldots+\frac{a_n}{10^{n-1}}\right)\frac{10}{16^2}.$$

Здесь уже число $\frac{a_2}{10^2}$ занимает разряды с 5-го по 8-й, а второе слагаемое находится справа от восьмого разряда. При выделении разрядов 5—8 будет найдена двоичная запись a_2 — второго десятичного разряда числа A. Повторение этой группы операций n раз дает двоичную запись всех n разрядов $a_1, a_2, \ldots, a_n$, т. е. двоично-десятичный код числа A.

В машине М-3 для двоично-десятичного кода могут быть использованы только 28 старших разрядов, т. е. $n=7$. Но может оказаться, что, начиная с некоторого места, все разряды переводимого числа равны нулю. Чтобы не выполнять лишних циклов, счет прекращается, как только разность между $2^{-m}+2^{-n}=0,\underbrace{00\ldots011}_{m}$ и переводимым числом становится положительной, т. е. как только все оставшиеся разряды переводимого числа, кроме двух последних (29-го и 30-го), становятся равными нулю.

Программа перевода чисел из двоичной системы счисления в десятичную на машине М-3

$k+1$	ПЧ	$a+7$	$a+2$	Очистка ячейки $a+2$
$k+2$	ПЧ	$a+3$	$a+8$	Передача числа $p=0,111100\ldots0$ в рабочую ячейку $a+8$
$k+3$	$\times$	$a+5$	$a+1$	Умножение $\left(\frac{a_{i+1}}{10}+\ldots+\frac{a_n}{10^{n-1}}\right)$ на $\frac{10}{16}$
$k+4$	$\wedge$	$a+8$	$a+9$	Выделение $\frac{a_i}{10^i}$ в рабочую ячейку $a+9$
$k+5$	$+$	$a+2$	$a+2$	Накапливание ответа
$k+6$	$+-$	$a+1$	$a+1$	Выделение $\left(\frac{a_{i+1}}{10}+\ldots+\frac{a_n}{10^{n-1}}\right)\frac{1}{16^i}$
$k+7$	$\times$	$a+4$	$a+8$	Сдвиг числа p на 4 разряда вправо
$k+8$	$+-$	$a+1$	$a+6$	Сравнение переводимого числа с числом $2^{-m}+2^{-n}$
$k+9$	УП	$k+3$	—	

$a+1$	Двоичная запись переводимого числа
$a+2$	Образуется двоично-десятичный код переводимого числа
$a+3$	$p=0,\ 111100\quad 0$
$a+4$	$\frac{1}{16}=0,\ 000100\quad 0$
$a+5$	$\frac{10}{16}=0,\ 101000\quad 0$
$a+6$	$2^{-10}+2^{-16}=0,00\quad 011$
$a+7$	$0=0,\ 00\quad 0$
$a+8$	Рабочая ячейка для хранения $0,\underbrace{0\quad 011110}_{a}\quad 0$
$a+9$	Рабочая ячейка для хранения $\frac{a_i}{10^i}$

3-9. Выделение целой и дробной частей на машинах с плавающей и фиксированной запятой

Значение операции выделения целой части будет разъяснено при рассмотрении задачи о вычислении значения функции e^x для больших значений x. Как известно, ряд

$$e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\dots$$

сходится для любых значений x, но практически, если x достаточно велико, приходится суммировать слишком большое число членов, прежде чем остаток ряда станет меньше заданной погрешности ε. Оценим, например, при каком n выполняется неравенство

$$\left|\frac{x^n}{n!}\right|<0{,}01,$$

если $x=e^5\approx 146$. На основании формулы Стирлинга

$$n!\approx\left(\frac{n}{e}\right)^n\sqrt{2\pi n}$$

n должно удовлетворять уравнению

$$(e^5)^n=0{,}01\left(\frac{n}{e}\right)^n\sqrt{2\pi n},$$

$$e^{5n}=0{,}025\left(\frac{n}{e}\right)^n\sqrt{n}$$

или после логарифмирования

$$5n=n\lg n-n+\frac{1}{2}\lg n+$$
$$+\lg 0{,}025;$$
$$12n=(2n+1)\lg n-7{,}4.$$

Если положить $n=e^5\approx 396$, то в последнем равенстве слева получается $12e^5$, а справа $12e^5-1{,}4$, откуда следует, что нужно взять около 400 членов ряда, чтобы член $\frac{x^n}{n!}$ был по модулю меньше 0,01, если $x=146$. Следовательно, если вычислять e^x по разложению в ряд, то при больших значениях аргумента x придется выполнить громадное число операций, что приведет к резкому увеличению времени счета. Между тем при составлении программ следует стремиться к максимальному сокращению времени решения данной задачи, ибо, во-первых, вероятность сбоев машины, т. е. вероятность возникновения ошибки, тем больше, чем больше время работы машины, и, во-вторых, высокая стоимость современных электронных машин предъявляет строгие требования к их эффективному использованию. Эти соображения заставляют искать более быстрый метод вычисления e^x.

Для этого можно воспользоваться операцией выделения целой части и представить аргумент x в виде:

$$x=[x]+\{x\},$$

где $[x]$ — число целое, а $\{x\}$ — правильная дробь. Тогда

$$e^x=e^{[x]+\{x\}}=e^{[x]}e^{\{x\}}.$$

Множитель $e^{\{x\}}$ вычисляется разложением в ряд весьма быстро, так как $\{x\}<1$ и, следовательно,

$$\frac{\{x\}^n}{n!}<\frac{1}{n!}.$$

т. е. уже при $n=10$

$$\frac{\{x\}^n}{n!}<\frac{1}{3\,628\,800}\approx 2{,}7\cdot 10^{-7}$$

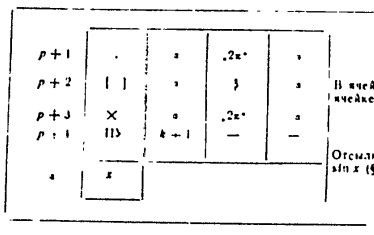
(как известно, ряд $1+\frac{1}{11}+\frac{1}{21}+\dots$ сходится очень быстро). Остается умножить величину $e^{\{x\}}$ на e (или на $\frac{1}{e}$, если $[x]<0$) ровно $[x]$ раз, что очень просто сделать, так как $[x]$ есть число целое. В нашей условной машине можно получить разложение аргумента на целую и дробную части при помощи одной команды взятия антье (см. § 1-7), но, например, в машине М 3 сделать это непосредственно уже нельзя. Тот факт, что в машине М 3 член $\frac{x^n}{n!}$ берется с масштабным коэффициентом M, так что $\left|\frac{x^n}{n!}\cdot\frac{1}{M}\right|<1$, нисколько не меняет существа вопроса, так как для определения конца счета этот член сравнивается не с ε, а с ε/M. Следовательно, и здесь желательно выделять целую часть аргумента x. Для этого можно сделать следующее. Пусть x/M хранится в ячейке a и $x>0$. В некоторую ячейку δ помещается число $\frac{1}{M}$, которое затем последовательно вычитается из содержимого ячейки a, пока в a не останется число, меньшее $\frac{1}{M}$. Каждое такое вычитание сопровождается умножением содержимого ячейки δ на e. Таким образом, одновременно в ячейке a образуется число $\frac{\{x\}}{M}$, а в ячейке δ — число $e^{[x]}/M$. Ответ будет получен, если вычислить $e^{\{x\}}/M$ по программе предыдущего параграфа (используя ее в качестве подпрограммы) и умножить результат на $e^{[x]}/M$.

25

$p+1$	ПЧ	$\frac{1}{Me}\cdot$	δ	$\frac{1}{Me}$				
$p+2$		$\frac{1}{e}\cdot$	δ	$\frac{1}{M}$	$\frac{e}{M}$	$\dots$	$\frac{e^{[x]-1}}{M}$	$\frac{e^{[x]}}{M}$
$p+3$	—	$\frac{1}{M}\cdot$	a	$\frac{x-1}{M}$	$\frac{x-2}{M}$		$\frac{\{x\}}{M}$	$\frac{\{x\}-1}{M}$
$p+4$	УП	$p+5$	$p+2$					
$p+5$	+,	$\frac{1}{M}\cdot$	—					$\frac{\{x\}}{M}$
$p+6$	↓	$\frac{1}{M}\cdot$	a					$\{x\}$
$p+7$	ПЧ	$p+12$	$k+11$					
$p+8$	ПУ	$k+1$	—					
$p+9$	×	δ	)					$\frac{e^{[x]}}{M}\cdot\frac{e^{\{x\}}}{M}$
$p+10$	↓	$\frac{1}{M}\cdot$	)					$\frac{e^x}{M}$
$p+11$	Стоп							
$p+12$	ПУ	$p+9$	—					
a	x/M							

В пояснение этой программы следует сказать несколько слов. Надо было вычитать $1/M$ из числа x/M до тех пор, пока разность не оказывалась меньше $1/M$ (напомним, что $x>0$). Между тем операция условного перехода $p+1$ передает управление по первому адресу только тогда, когда в ячейке a образуется отрицательное число. Ввиду этого в ячейку 7 вначале передается $1/Me$ (а не $1/M$), а в команде $p+5$ мы к содержимому ячейки a прибавляем $1/M$ (команда условной передачи управления $p+4$ не изменяет состояния регистра арифметического узла).

Выделение целой части положительного числа x легко выполнить, если как масштаб M взять степень числа



$p+1$	.	a	,2π*	a
$p+2$	[ ]	a	}	a
$p+3$	×	a	,2π*	a
$p+4$	ПВ	$k+1$	—	—
a	x			

В ячейке a образовано $\left\{x/2\pi\right\}$, а в ячейке } образовано $\left\lfloor x/2\pi \right\rfloor$

$$x_0 = \left\{\frac{x}{2\pi}\right\} 2\pi$$

Отсылка к подпрограмме вычисления $\sin x$ (§ 3.3)

два $M=2^k$. Тогда целая часть x будет записана в старших k разрядах ячейки и может быть выделена операцией логического умножения на число

$$\underbrace{11\ldots1}_{k}\underbrace{00\ldots0}_{}$$

При большой серии вычислений выбор масштаба $M=2^k$ менее удобен, чем $M=10^k$, так как деление x на масштаб M выполняется до ввода числа x в машину.

Операция выделения целой и дробной частей может быть использована при вычислении значения функции $\sin x$ при больших значениях x. Если в этом случае вычислять значения $\sin x$ по разложению в ряд, то придется просуммировать слишком большое число членов. Поэтому можно воспользоваться соотношением

$$\sin(2k\pi+x) = \sin x$$

и представить аргумент в виде

$$x = 2k\pi + x_0$$

Для этого проще всего разделить аргумент x на 2π, взять дробную часть этого отношения и умножить ее на 2π

$$x_0 = \left\{\frac{x}{2\pi}\right\} 2\pi$$

После того как x_0 определено, для вычисления $\sin x_0$ можно воспользоваться программой, составленной в § 3.3. Тогда программа примет вид

Для сокращения времени счета мало свести аргумент к значению, по модулю меньшему, чем 2π. Пользуясь известными тригонометрическими соотношениями, полезно свести вычисления к значению аргумента, по модулю меньшему, чем $\frac{\pi}{4}$, но мы на этом не будем останавливаться.

Методы, близкие к описанным в этом параграфе, помогают и при вычислении значения функции $\lg x$. Обычная формула разложения в ряд

$$\lg(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots$$

$$-1 < x \leq 1$$

совершенно непригодна, так как при x, близком к единице, члены этого ряда убывают как $\frac{1}{n}$, и для того чтобы

этот член стал меньше, чем, например, 10^{-4}, нужно подсчитать 10^4 членов. Для сокращения объема вычислений можно заменить здесь x на $-x$, формула примет вид:

$$\lg(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \ldots$$

При вычитании из нее предыдущей формулы получается

$$\lg\frac{1-x}{1+x} = -2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots\right).$$

Этот ряд уже сходится вдвое быстрее, так как в нем имеются только члены с нечетной степенью.

Пусть теперь надо вычислить $\lg y$ для некоторого значения $y>0$. Можно представить y в виде

$$y = e^p z,$$

где p — целое число (положительное, отрицательное или нуль) и

$$\frac{1}{e} < z \leq 1$$

Положим

$$\frac{1-z}{1+z} = x, \quad 0 < x < \frac{1}{2}$$

Тогда

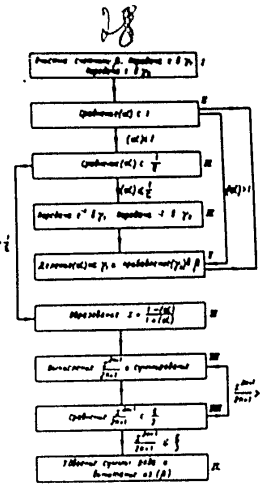


Рис. 3-4.

$$\lg y = \lg e^p z = \lg e^p + \lg z = p + \lg z = p + \lg\frac{1-\frac{1-z}{1+z}}{1+\frac{1-z}{1+z}} =$$

$$= p + \lg\frac{1-x}{1+x} = p - 2\left(x + \frac{x^3}{3} + \ldots + \frac{x^{2n-1}}{2n-1}\right) -$$

$$-2\left(\frac{x^{2n+1}}{2n+1} + \ldots\right) = p - S_n - \Delta_n;$$

$$\Delta_n = 2\left(\frac{x^{2n+1}}{2n+1} + \ldots\right) < \frac{2x^{2n+1}}{2n+1}(1 + x^2 + x^4 + \ldots) = \frac{x^{2n+1}}{2n+1}\cdot\frac{2}{1-x^2}.$$

Так как $\frac{2}{1-x^2} < 3$, то при $\frac{x^{2n+1}}{2n+1} < \frac{\varepsilon}{3}$ остаток Δ_n по модулю меньше ε.

В данном случае $|x| < \frac{1}{2}$, следовательно, при $n=6$

$$\left|\frac{x^{2n+1}}{2n+1}\right| < \frac{1}{2^{13}\cdot 13} < 10^{-4},$$

т. е. теперь для получения той же степени точности (и даже большей) достаточно взять не 10^4 членов, а только 6.

Чтобы привести y к виду $e^p z$, т. е, чтобы определить числа p и z, можно, поместив y в ячейку e, поступить одним из следующих способов:

если $y>1$, то надо делить содержимое ячейки e на e, пока частное не станет меньше или равно 1, сопровождая каждое деление прибавлением единицы в счетчик β;

если $y \leq \frac{1}{e}$, то надо делить содержимое ячейки e на e^{-1}, пока частное не станет больше $\frac{1}{e}$, сопровождая каждое де-

ление прибавлением отрицательной единицы в счетчик β.

Таким образом, в ячейке α будет образовано x, а в счетчике β число p. Затем по формуле

$$x = \frac{1-z}{z}$$

определяется число x, для которого вычисляется ряд

$$x + \frac{x^3}{3} + \frac{x^5}{5} + \dots$$

до тех пор, пока очередной вычисленный член не станет меньше $\varepsilon/3$. Полученный результат удваивается и вычитается из p. Таким образом, в ячейке β образуется $\lg y$. Логическая схема этой программы приведена на рис. 3-4 (в ячейке α находится число y).

Ниже приводится основная часть этой программы (операторы I—V).

I	$k+1$	ПЧ	α^*	—	β
	$k+2$	ПЧ	α^*	—	γ_1
	$k+3$	ПЧ	α^*	—	γ_2
II	$k+4$	<	α^*	a	$k+8$
III	$k+5$	<	$\frac{1}{c}*$	a	$k+11$
IV	$k+6$	ПЧ	α^{*-1*}	—	γ_1
	$k+7$	ПЧ	$*-1^*$	—	γ_2
V	$k+8$		a	γ_1	a
	$k+9$	+	γ_2	β	β
	$k+10$	ПУ	$k+4$	—	—
VI	$k+11$				. .

ГЛАВА ЧЕТВЕРТАЯ

ПРОГРАММИРОВАНИЕ МАТЕМАТИЧЕСКИХ ЗАДАЧ

4-1. Схема программы

В предыдущей главе на отдельных примерах показаны основные приемы составления программ. В этой главе будут изложены методы программирования и постановки математических задач на машинах.

Как неоднократно отмечалось выше, для решения на машине любой задачи должен быть выбран какой-нибудь численный метод, сводящий описание решения к ряду арифметических действий над числами. Выбор метода определяется такими соображениями, как требуемая точность, быстрота, простота и допустимость программирования на данной машине и т. п. В большинстве случаев это является сложной задачей, составляющей содержание приближенных методов анализа. Мы не будем останавливаться на этих вопросах, так как они не имеют непосредственного отношения к методике программирования задач, хотя и влияют на нее.

Итак, для решения поставленной математической задачи предложен некоторый численный метод и требуется составить программу, реализующую этот метод на машине. Для этого в первую очередь следует представить ясную картину всего процесса счета, т. е. точно установить, по каким математическим формулам ведется весь счет, в каком порядке и в зависимости от каких условий нужно обращаться к тем или иным формулам, сколько раз ведется счет по каждой из них, над какими числами производятся действия и т. п. Кроме того, нужно выделить те из чисел, получаемых в процессе счета, которые подлежат выводу из машины, и те, которые нужны только для проведения дальнейших вычислений. Если среди последних имеются числа, нужные в последующих циклах, то они должны быть сохранены в отдельных ячейках.

Для составления схемы счета весь вычислительный процесс разлагают на ряд так называемых операторов счета, каждый из которых реализует одну или несколько математических формул. При этом обычно в одном операторе объединяются такие формулы, которые в течение всего процесса счета выполняются одновременно. Кроме того, в одном операторе могут быть объединены формулы, которые оперируют с одним и тем же числовым материалом, как говорят, требуют одной и той же информации. Например, если две различные математические формулы имеют достаточно большую общую часть, то она может быть выделена в отдельный оператор. Вместе с тем формулы, которые в процессе вычислений не встречаются одновременно или никогда бывают нужны вместе, а иногда — отдельно, должны входить в разные операторы. Точно так же в разные операторы входят и такие формулы, которые служат для определения одних и тех же величин, если в зависимости от различных условий следует пользоваться либо одними, либо другими из этих формул. Вообще трудно дать формальное определение оператора счета, но это понятие станет достаточно ясным из примеров, рассматриваемых в последующих параграфах.

Мы будем обозначать операторы счета большими латинскими буквами A, B, C, ... Иногда операторам будут приписываться индексы A_i, B_{jk}, указывающие, что эти операторы изменяются в зависимости от каких-то индексов i, j, k ...

Теперь уже вся схема счета может быть описана при помощи операторов, т. е. можно указать, какие операторы, в каком порядке, сколько раз и в зависимости от каких условий должны быть выполнены для полной реализации всего вычислительного процесса. С простейшими примерами схем счета можно ознакомиться на материале предыдущей главы. Например, в § 3-7 была составлена программа вычисления полинома по схеме Горнера. Если обозначить в этой программе через A_i оператор, прибавляющий в ячейку β коэффициент a_i и умножающий содержимое этой ячейки на x, то вся схема счета будет иметь вид:

$$\prod_{i=0}^{n} A_i = A_0 \cdot A_1 \dots A_n,$$

где знак $\prod_{i=0}^{n}$ означает: произведение операторов A_i для $i = 0, 1, \dots, n$. Простота этой схемы объясняется тем, что в данном случае заранее известно число повторений цикла в вычислительном процессе. Но в § 3-3 были рассмотрены программы, в которых число повторений цикла определялось самой программой. Для составления схемы счета в этом случае недостаточно иметь одни лишь операторы счета; необходимо ввести так называемые логические операторы, которые проверяют выполнение каких-либо условий и реализуются обычно при помощи операций условной передачи управления. Например, для составления схемы счета при вычислении e^{x_0} нужно ввести оператор счета A, вычисляющий член $\frac{x_0^n}{n!}$ и прибавляющий этот член в ячейку β, и логический оператор λ, сравнивающий $\left|\frac{x_0^n}{n!}\right|$ с ε. Тогда схема счета примет вид, представленный на рис. 4-1,

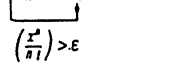


Рис. 4-1.

Уже из этих двух примеров легко видеть, что схема счета не исчерпывает всей программы, так как, помимо команд, необходимых для реализации операторов счета и логических операторов, программа содержит ряд других команд, необходимых для подготовки ячеек, переадресации команд и т. п. Чтобы иметь возможность описать полностью программу, надо на основании схемы счета составить схему программы. Грубо говоря, схема счета отличается от схемы программы тем, что последняя описывает всю программу, в то время как первая охватывает лишь собственно арифметические команды этой программы. Можно сказать, что схема программы должна описывать, помимо чисто арифметических действий, также все то, что относится к управлению этими действиями. Следовательно, для составления схемы программы, помимо операторов счета, должен быть введен еще целый ряд других операторов. К числу наиболее часто встречающихся операторов относятся следующие:

P_α — оператор восстановления, этот оператор восстанавливает первоначальный вид тех команд, записных от индекса i, которые он стирает в процессе счета;

P_γ — оператор запоминания счетчика, этот оператор засылает в счетчик начальное значение индекса i;

$I(\alpha i)$ — оператор преобразования или пересылки, этот оператор преобразует команды, зависящие от индекса i, на n единиц;

$I(i)$ — оператор, прибавляющий единицу к счетчику индекса i;

λ_i — логическое условие, проверяющее величину индекса i;

T — оператор передачи, этот оператор выбирает из всего набора констант нужную или группу и передает ее в рабочие ячейки. Этот же оператор подготавливает те ячейки, содержание которых преобразуется в процессе работы программы, например, рабочие ячейки, ячейки, в которых накапливается ответ и т. п.

При помощи этих операторов можно составить логическую схему программы. Например, если через T обозначить оператор, пересылающий a_i в ячейки β_i, а через $\Delta F(i)$ оператор, увеличивающий индекс i оператора A_i на единицу, то схема приведенной в § 3-7 программы вычисления полинома примет вид, изображенный на рис. 4-2.

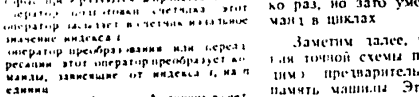


Рис. 4-2

В гл. 3 для описания программы использованы блок-схемы, которые более наглядны, но не описывают программу с такой полнотой и точностью, как логические операторные схемы. Подробнее эти схемы будут рассмотрены в следующих параграфах, а сейчас будет сделан ряд общих замечаний относительно методики программирования задач.

Во-первых, при составлении логической схемы программы следует стремиться к максимальному уменьшению времени счета, что, в частности, достигается уменьшением числа команд в программе. При этом надо иметь в виду, что особенное внимание следует обратить на число команд, входящих в многократно повторяемые циклы. Поэтому бывает иногда выгодно пойти на общее увеличение числа команд в программе, если это увеличение идет за счет частей программы, выполняемых всего один или несколько раз, но зато уменьшает число команд в циклах.

Заметим далее, что для составления точной схемы программы необходимо предварительно распределить память машины. Это значит, что мы должны отнести ячейки под все константы, которые либо входят в условие задачи (например, элементы определителя), либо нужны для ее решения на машине (например, константы сравнения). Кроме того, должны быть отведены ячейки под величины, образующиеся в процессе счета (например, в случае, когда элементы определителя заранее не известны, а даны формулы, по которым они вычисляются) под рабочие ячейки и т. п. При этом рабочими мы называем такие ячейки, в которых хранятся результаты промежуточных вычислений. Наконец, нужно предусмотреть ячейки и под саму программу. Распределение памяти, особенно важное для составления таких операторов, как оператор переадресации, требует некоторого опыта и не всегда может быть сделано совершенно точно сразу хотя бы потому, что нельзя заранее точно предусмотреть число команд программы. Впрочем при небольшом навыке сделать это не очень трудно. Если заранее ясно, что поместить всю программу (включая константы, рабочие ячейки и т. д.) во внутреннюю память машины нельзя, то часть материала помещают во внешнее запоминающее устройство. При этом для сокращения времени решения задач нужно позаботиться о том, чтобы уменьшить до минимума число обращений к внешней памяти.

Как отмечалось в § 3-7, восстановление команд, преобразуемых в процессе счета, производится одним из двух способов. Во-первых, из кода данной команды можно вычесть число, прибавленное к ней в процессе работы программы. Во-вторых, в отдельной ячейке можно хранить первоначальный, как говорят, стандартный вид команды и пересылать ее в нужное место программы. Этот последний способ следует предпочесть первому, так как он уменьшает возможность ошибки

и со стороны программиста и со стороны машины. Передачу первоначального вида команды можно выполнять в самом начале программы. В последнем случае можно остановить машину в любом месте программы и вновь приступить к решению задачи, начиная с первой команды. То же самое имеет место и при сбоях машины. Если же передача стандартных команд производится после их преобразования или же восстановление команд производится вычитанием прибавленных чисел, то при остановке машины в середине программы или при сбое вся программа должна быть заново введена в машину. Программы, в которых восстановление команд предшествует их преобразованию, называют обычно самовосстанавливающимися. Они особенно удобны для таких задач, которые требуют просчета большого числа вариантов. Этот способ подробно иллюстрируется в § 4-3.

Составление схемы программы выгоднее проводить в несколько этапов. Весь счет разбивается на укрупненные операторы и при помощи этих операторов составляется общая схема программы. После этого составляется схема каждого такого укрупненного оператора. Такое включение подсхем в основную схему иногда бывает многостепенным и сильно облегчает как составление, так и общее обозрение всей схемы. Этот метод применен в § 5-2.

Составление программы удобнее вести в два приема. Вначале составляется программа такого типа, как в гл. 3. Это, так сказать, символические программы, в которых адреса команд указаны условно $k+1$, $k+2$, ..., вместо рабочих ячеек стоят ячейки α, β, γ, ..., вместо адресов некоторых чисел стоят сами числа, взятые в кавычки, и т. д. После того как будет составлена такая программа, можно провести окончательное распределение памяти. Теперь программа может быть закодирована, пробита на перфоленту, и в нужный момент введена в машину.

Большинство математических задач содержит такие часто встречающиеся процессы, как вычисление корней, оперирование с комплексными числами, вычисление значений специальных функций и т. д. Составлять программы

этих процессов каждый раз заново нецелесообразно. Поэтому эти процессы заранее программируются и машина снабжается библиотекой таких программ, которые называют стандартными подпрограммами, так как они не зависят от специфических свойств какой-нибудь конкретной задачи. При составлении программы задачи в нее легко можно включить имеющиеся уже стандартные подпрограммы, так как это было сделано в § 3-6. Стандартные подпрограммы, обычно составленные наиболее тщательно, оказываются и весьма экономными с точки зрения расхода машинного времени.

4-2. Программа решения обыкновенных дифференциальных уравнений методом Рунге — Кутта

В качестве первого примера, иллюстрирующего методы, изложенные в предыдущем параграфе, можно рассмотреть задачу об отыскании решения обыкновенного дифференциального уравнения методом Рунге — Кутта.

Пусть надо найти решение уравнения

$$\frac{dy}{dx} = \Phi(x, y),$$

удовлетворяющее условиям

$$y(x_0) = y_0;\ x_0 \leqslant x \leqslant X.$$

Как известно (см. гл. 9) по методу Рунге—Кутта, отрезок (x_0, X) разбивается точками $x_1, x_2, \ldots, x_n = X$, $x_{l+1} - x_l = h$ и за значение функции $y(x)$ в точке x_{l+1} принимается величина

$$y_{l+1} = y_l + k,\ l = 0, 1, 2, \ldots, n-1,$$

где 32

$$k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4);$$

$$k_1 = h\Phi(x_l, y_l);$$

$$k_2 = h\Phi\left(x_l + \frac{h}{2},\ y_l + \frac{k_1}{2}\right);$$

$$k_3 = h\Phi\left(x_l + \frac{h}{2},\ y_l + \frac{k_2}{2}\right);$$

$$k_4 = h\Phi(x_l + h,\ y_l + k_3).$$

Чтобы составить схему счета, вводятся следующие операторы счета

L — оператор, прибавляющий $h/2$ к текущему аргументу.

K' — оператор, прибавляющий $k_2/2$ к y_i.

K'' — оператор, прибавляющий $k_3/2$ к y_i.

K''' — оператор, прибавляющий k_4 к y_i.

K — оператор, вычисляющий величину k_i.

Φ — оператор, вычисляющий значение функции $\Phi(x, y)$.

Λ — оператор, вычисляющий $k_i = h\Phi(x, y)$.

При помощи этих операторов очень легко составить схему счета

$$\prod_{i=1}^{4} (\Phi\gamma LK^{i}\Phi\gamma K'\Phi\gamma LK''\Phi\gamma K)_i$$

Перед тем как составить схему программы, нужно распределить ячейки памяти для чисел x и y, для которых вычисляется функция Φ, помещаются в ячейках α и β соответственно, а значение Φ образуется в ячейке δ. Чтобы вновь вычисленное значение Φ не забивало старого, следует передавать числа k_1, k_2, k_3 в ячейки δ_1, δ_2, δ_3 (четвертое значение Φ, т. е. k_4, можно не записать).

Далее, так как к аргументу x_i прибавляется всегда одна и та же величина $h/2$, то это прибавление можно делать в одну и ту же ячейку α. К y_i прибавляется каждый раз новая величина и поэтому нужно запасти y_i еще в одной ячейке γ.

Проведя такое распределение памяти, можно составить схему программы, предварительно уточнив работу рассмотренных выше операторов и введя новые

P — оператор, передающий x_i в α и y_i в β и γ.

H — оператор, вычисляющий $h/2$.

P_j — оператор, очищающий счетчик j.

R — оператор, восстанавливающий оператор T_j.

Φ — оператор, вычисляющий значения $\Phi(x, y)$ по x и y, находящимся в ячейках α и β, само значение $\Phi(x, y)$ образуется в ячейке δ.

N — оператор, образующий $k_j = h\Phi$ ($j = 1, 2, 3, 4$) в ячейке δ.

T_j — оператор, передающий k_j в ячейку δ_j ($j = 1, 2, 3$).

L — прибавляет $h/2$ в ячейку α.

K' — образует $\frac{k_1}{2}$ (делит содержимое ячейки δ пополам);

K'' — содержимое ячейки δ $\left(\text{т. е } \frac{k_2}{2}, \frac{k_3}{2}, k_4\right)$ прибавляет к y_i (хранящемуся в ячейке γ) и помещает результат в ячейку β, эти два оператора K' и K'' заменяют собой введенные выше операторы K'' и K'''.

K — оператор, вычисляющий и печатающий y_{i+1}, значение y_{i+1} помещается в β и γ.

λ_j — оператор, проверяющий значение индекса j.

$F(j)$ — оператор, преобразующий T_j.

$I(j)$ — оператор, прибавляющий 1 в счетчик j.

λ_x — оператор, сравнивающий текущий аргумент x_{i+1}, для которого вычислено значение y_{i+1}, с X.

Теперь схема программы будет иметь вид, представленный на рис. 4-3.

Имея схему программы и подробное описание каждого оператора, уже нетрудно составить саму программу. При этом вид функции Φ нас не будет интересовать, и мы будем предполагать, что программа вычисления $\Phi(x, y)$ хранится в ячейках от $l+1$ до $l+n$, причем в ячейке $l+n$ находится команда, возвращающая нас к прерванному месту программы

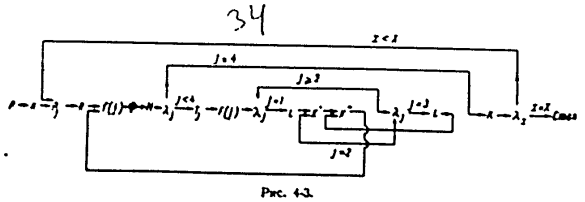


Рис. 4-3.

Программа решения обыкновенных дифференциальных уравнений методом Рунге—Кутта

P	$k+1$	ПЧ	α_0	—	α
	$k+2$	ПЧ	β_0	—	β
	$k+3$	ПЧ	γ_0	—	γ
H	$k+4$		h	2	p_1
P_j	$k+5$	ПЧ	0	—	p_0
R	$k+6$	ПЧ	$k+31$	—	$k+11$
$I(j)$	$k+7$	+	1	p_0	p_0
	$k+8$	ПУ	$l+1$	—	—
N	$k+9$	×	δ	h	δ
λ_j	$k+10$	<	3	p_0	$k+21$
T_j	$k+11$	ПЧ	δ	—	δ_j
$F(j)$	$k+12$	СК	$k+11$	1111	$k+11$
λ_j	$k+13$	<	1	p_0	$k+18$
L	$k+14$	+	p_1	α	α
K'	$k+15$	·	δ	2	δ
Λ''	$k+16$	+	γ	δ	β
	$k+17$	ПУ	$k+7$		
λ_j	$k+18$	<	p_0	3	$k+15$
L	$k+19$	+	p_1	α	α
	$k+20$	ПУ	$k+16$		
K	$k+21$	+	δ_1	δ	δ
	$k+22$	+	δ_2	δ_1	δ_1
	$k+23$	×	δ_2	2	δ_2
	$k+24$	+	δ	δ_2	δ
	$k+25$	×	δ	$\frac{1}{6}$	δ_2
	$k+26$	+	γ	δ	β
	$k+27$	ПЧ	β	—	y_{i+1}
	$k+28$	Печать	β		
λ_x	$k+29$	<	α	X	$k+5$
	$k+30$	Стоп			
	$k+31$	ПЧ	δ	—	δ_1

Первоначальный вид оператора T_1

Эта программа может быть названа стандартной подпрограммой (в смысле, указанном в конце предыдущего параграфа) решения обыкновенного дифференциального уравнения первого порядка методом Рунге—Кутта. Раз составленная она может быть отлажена и пробита на перфоленту, которая затем хранится в библиотеке стандартных подпрограмм.

Каждый раз, когда надо решить уравнение указанного типа, достаточно составить программу вычисления его правой части и поместить первую команду этой программы в ячейку $l+1$, а в конце программы поставить передачу управления в ячейку $k+9$. Эта программа также пробивается на ленту, и затем две ленты с программами вводятся независимо друг от друга

каждая на свое место в запоминающее устройство машины.

В гл. 7 будут рассмотрены логические схемы программ решения методом Рунге — Кутта систем обыкновенных дифференциальных уравнений.

4-3. Программа вычисления определителя. Преобразование команд в нескольких циклах

Общие приемы составления программы математических задач хорошо могут быть проиллюстрированы на примере вычисления определителя

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{vmatrix}$$

Для решения этой задачи может быть выбран метод Гаусса, по которому все элементы определителя, лежащие ниже главной диагонали, обращаются в нуль и последовательно определитель становится равным произведению диагональных элементов. Само обращение в нуль указанных элементов производится на основании теоремы, согласно которой из всех элементов одной строки определителя можно вычесть элементы другой строки, умноженные на любой коэффициент. Чтобы обратить в нуль элемент a_{21}, надо образовать коэффициент a_{21}/a_{11} и затем вычесть из элементов второй строки соответствующие элементы первой строки, умноженные на этот коэффициент. При вычитании из элементов третьей строки соответствующих элементов первой строки, умноженных на коэффициент a_{31}/a_{11}, обращается в нуль элемент a_{31}. Таким образом можно обратить в нуль все элементы первого столбца, лежащие ниже диагонали. Чтобы обратить в нуль элементы второго столбца, поступают следующим образом. Составляется коэффициент a_{32}/a_{22} и вычитаются из элементов третьей строки соответствующие элементы второй строки, умноженные на этот коэффициент. При этом в нуль обращается элемент a_{32}. Чтобы обратить в нуль элемент a_{42}, нужно из элементов четвертой строки вычесть соответствующие элементы второй строки, умноженные на коэффициент a_{42}/a_{22} и т. д., пока не обратится в нуль остальные элементы второго столбца (заметим, что первый элемент второй строки уже нуль и такое вычитание не „испортит" элемент из первого столбца). Обратив в нуль остальные элементы второго столбца, надо перейти к третьему и т. д. Когда будут обращены в нуль все элементы, стоящие ниже диагонали в 1-м, 2-м, $i-1$-м столбцах, а также первые $j-1$ элемент $a_{i,1}, a_{i,2}, \dots, a_{i,j-1}$ стоящие в i-том столбце ниже диагонального элемента, нужно обратить в нуль элемент a_{ij} (первый индекс означает номер строки, второй — номер столбца, см. рис. 4-1). Для этого вычисляется коэффициент a_{ij}/a_{jj} и вычитаются из элементов j-той строки соответствующие элементы i-той строки, умноженные на этот коэффициент. При этом для сокращения времени имеет смысл производить вычитание лишь тех элементов a_{ik} ($k=i+1, i+2, \dots, n$), поскольку заведомо известно, что элементы $a_{i1}, a_{i2}, \dots, a_{i,j-1}$ при этом вычитании обратятся в нуль.

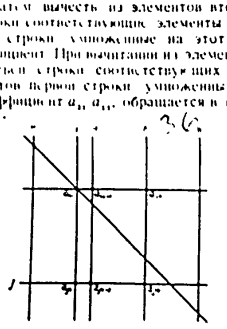


Рис. 4-1

Для составления схемы счета удобно ввести следующие операторы счета:

B_{ij} — оператор, вычисляющий коэффициент a_{ij}/a_{jj}.

A_{ijk} — оператор, вычисляющий новое значение элемента

$$a'_{ik} = a_{ik} - \frac{a_{ij}}{a_{jj}} a_{jk},$$

стоящего на пересечении i-той строки и k-того столбца.

При помощи этих операторов можно записать схему счета по приведению определителя к треугольному виду

$$\prod_{i=1}^{n} \prod_{j=i+1}^{n} B_{ij} \prod_{k=i+1}^{n} A_{ijk}$$

Введя оператор C_i, перемножающий диагональные элементы, можно составить символическую схему счета

$$\prod_{i=1}^{n} \prod_{j=i+1}^{n} B_{ij} \prod_{k=i+1}^{n} A_{ijk} \, C_i$$

Теперь нужно распределить ячейки памяти и, в частности, выделить ячейки под элементы определителя. Элементы определителя удобно расположить в последовательных ячейках запоминающего устройства в порядке их следования по строкам определителя, как это указано в конце приводимой ниже программы. Следовательно, при переходе от одного элемента к соседнему по строке номер ячейки меняется на одну единицу, а по столбцу — на n единиц.

Перед тем как приступить к составлению схемы программы, надо сделать одно важное замечание, на которое следует обратить особое внимание читателя. Как видно из символической схемы счета, операторы B_{ij} и A_{ijk} меняются не в одном цикле, а в нескольких (каждому знаку произведения, т. е. каждому из индексов i, j, k, в схеме счета соответствует свой цикл в программе).

Может случиться, что команды этих операторов зависят каждый только от одного из индексов i, j, k. Тогда их преобразование и восстановление выполняются обычным способом. Но очень часто такие операторы содержат команды, зависящие одновременно от нескольких индексов. Следовательно, они изменяются в нескольких циклах, и ниже будет показано, как производится преобразование и восстановление таких команд. Забегая несколько вперед, можно рассмотреть для этого команду $k+15$ из приводимой ниже программы. Ее первоначальный вид выписан в конце программы в ячейке β_3. Команды $k+5$, $k+9$ и $k+12$ передают ее соответственно в ячейки β'_3, β''_3 и, наконец, на ее рабочее место в программе, т. е. в ячейку $k+15$. Зафиксируем значение $i=i_0$ и пусть сначала $j=i_0+1$. Индекс k должен пробежать все значения от i_0+1 до n, это преобразование выполняет команда $k+17$. Чтобы придать индексу j значение i_0+2, нужно в команде $k+15$ восстановить значение k, равное i_0+1, и увеличить j на 1 (так как j номер строки, то изменению j на 1 соответствует изменение зависящих от j адресов на n единиц). Код этой команды при $i=i_0$ и $j=i_0+1$ запасен в ячейке β''_3, следовательно, в ней можно изменить индекс j на 1 (что делает команда $k+21$) и передать управление к команде $k+12$, передающей β''_3 в ячейку $k+15$ (фактически команда $k+23$ передает управление команде $k+10$, так как от индекса j зависит не только команда $k+15$). Пусть, наконец, индексы j и k стали равными n и нужно придать i значение i_0+1 (так как i — номер диагонального элемента, то изменению i на 1 соответствует изменение соответствующих адресов на $n+1$). Так как ячейка β''_3 уже подвергалась преобразованиям, то это изменение индекса i выполняется в ячейке β'_3 (это делает команда $k+27$) и управление передается к команде, пересылающей содержимое этой ячейки на рабочее место. Фактически команда $k+30$ передает управление к ячейке $k+7$, команда $k+9$ передает содержимое ячейки β'_3 в ячейку β''_3 (надо будет вновь менять индекс j, но уже от значения i_0+2) и, наконец, команда $k+12$ передает содержимое ячейки β''_3 на рабочее место. Команда $k+5$, передающая первоначальный вид команды $k+15$, т. е. содержимое ячейки β_3 в ячейку β'_3, делает всю программу самовосстанавливающейся. Из приведенного анализа можно вывести весьма полезное правило. Если команда преобразуется в нескольких циклах, то в начале каждого цикла она передается в некоторую ячейку, а чтобы сделать

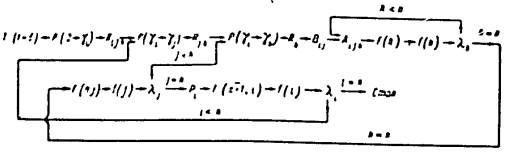


Рис 4-5.

это, нужно взять ее из той ячейки, в которую она была записана в начале предыдущего цикла. При этом в первом цикле она берется из той ячейки, в которой хранится ее первоначальный вид, а в последнем цикле она передается уже не в записную ячейку, а на ее рабочее место в программе. Внутри каждого цикла преобразование этой команды производится не в рабочем месте (исключая последний цикл), а в той ячейке, в которой она была записана в начале данного цикла.

Теперь мы можем составить схему программы (рис. 4-5), в которой через γ_i, γ_j, γ_h обозначены соответственно счетчики индексов i, j, h.

По этой схеме составляется программа, при этом учитывается характерная особенность операторов восстановления R_{ijk}, R_{jk}, R_k, о которой говорилось выше.

Программа вычисления определителя

$\gamma_i(1 \to n)\{k+1$	ПЧ	$\cdot 1^*$	—	k
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\varphi(0 \to \gamma_i)\{k+2$	ПЧ	$\cdot 0^*$	—	γ_i
$R_{ijk}\{k+3$	ПЧ	γ_i	—	γ_i'
$k+4$	ПЧ	γ_j	—	γ_j'
$k+5$	ПЧ	γ_h	—	γ_3'
$k+6$	ПЧ	γ_h	—	$k+24$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\varphi(\gamma_i+\gamma_j)\{k+7$	ПЧ	γ_i	—	γ_j
$R_{jk}\{k+8$	ПЧ	γ_j'	—	$k+13$
$k+9$	ПЧ	γ_3'	—	γ_3''
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\varphi(\gamma_h \to \gamma_k)\{k+10$	ПЧ	γ_h	—	γ_k
$R_k\{k+11$	ПЧ	γ_3'	—	$k+14$
$k+12$	ПЧ	γ_3''	—	$k+15$
$B_{ij}\{k+13$	$\times\times$	$\times$	$\times\times\times$	$\times\times$	$\cdot\cdot a_{ij}\cdot\ a_{ij}\cdot\ p_3$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$A_{ijk}\{k+14$	$\cdot\times$	$\times$	$\times\times\times$	$\times\cdot$	$\times\ p_3\ \ a_{ij}\cdot\ p_3$
$k+15$	$\times\times$	$\times$	$\times\times\times$	$\times\times$	$-\cdot a_{ij}\cdot\ p_3\ \ a_{n+2}$
$k+16$	СК	$k+14$	$\cdot IIIA^*$	$k+14$	
$k+17$	СК	$k+15$	$\cdot IIIIIA^*$	$k+15$	
$\delta(k)\{k+18$	+	$\cdot 1^*$	γ_k	γ_k	

$\lambda_k\{k-19$	$<$	γ_k	$\cdot n-1^*$	$k+14$	
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
$F(n!)\{k+20$	СК	$k+13$	$\cdot nIA^*$	$k+13$	
$k+21$	СК	γ_3''	$\cdot n!.IIIA^*$	γ_3''	
$I(1)\{k+22$	+	$\cdot 1^*$	γ_i	γ_i	
$\lambda_i\{k+23$	$<$	γ_i	$\cdot n-1^*$	$k+10$	
$D_i\{k+24$	$\times$	$\times$	$\times\times\times$	$\times\times$	
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\times k\cdot a_{ij}\cdot k$
$k+25$	СК	γ_1'	$\cdot(n+1)!IIIA^*$	γ_1'	
$I(\overline{n+1})\{k+26$	СК	γ_2'	$\cdot(n+1)!IIA^*$	γ_2'	
$k+27$	СК	γ_3'	$\cdot(n+1)!IIIA^*$	γ_3'	
$k+28$	СК	$k+24$	$\cdot(n+1)!IIA^*$	$k+24$	
$I(i)\{k+29$	+	$\cdot 1^*$	γ_i	γ_i	
$\gamma_i\{k+30$	$<$	γ_i	$\cdot n-1^*$	$k+7$	
$\ldots$				$\ldots$	
$k+31$	Стоп				

$B_{ij}\{\gamma_n$ — γ_n+n+1 — $a+1$ — p_1 — (i,j)

$A_{ijk}\{\gamma_k$ — p_1 — $a+2$ — p_2 — (i,k)

$C\{\gamma_n$ — $a+n+2$ — p_3 — $a+n+2$ — (i,j,k)

$C\{\gamma_n$ — δ — $a+1$ — k — (i)

Первоначальный вид команд. В скобках указано, от каких индексов зависит соответствующая команда

$a+1\ a_{11}$
$a+2\ a_{12}$
$\ldots$
$a+n\ a_{1n}$
— 1-я строка определителя

$a+n+1\ a_{21}$
$a+n+2\ a_{22}$
$\ldots$
$a+2n\ a_{2n}$
— 2-я строка определителя

$a+n(n-1)+1\ a_{n1}$
$a+n(n-1)+2\ a_{n2}$
$\ldots$
$a+nn\ a_{nn}$
— n-я строка определителя

γ_i — счетчик индекса i
γ_j — счетчик индекса j
γ_h — счетчик индекса h

Пунктирными линиями указаны границы циклов. Переменные команды не выписаны (на этих местах поставлена штриховка), но справа от программы указан их первоначальный вид. Ответ накапливается в ячейке δ.

4-4. Решение алгебраических и трансцендентных уравнений

Пусть известно, что на интервале (ω_0, ω) имеется конечное число корней алгебраического или трансцендентного уравнения $D(\omega)=0$ и что расстояние между двумя соседними корнями не меньше, чем $2h$. Ставится задача об определении наименьшего из этих корней. Для этого вычисляются значения функции $D(\omega)$ в точках ω_0, $\omega_1=\omega_0+h$, $\omega_2=\omega_0+2h, \ldots$ и сравниваются их знаки в двух последовательных точках. В силу сделанных предположений наименьший корень уравнения $D(\omega)=0$ лежит внутри первого из тех интервалов (ω_i, ω_i+h), на концах которого функция $D(\omega)$ имеет разные знаки. Этот интервал делится пополам на два интервала и из них выбирается тот, на котором функция меняет свой знак. Повторив так достаточное число раз, можно сделать интервал, на котором лежит корень, сколь угодно малым и, следовательно, определить искомый корень с любой степенью точности.

Приводимая ниже программа была использована при отыскании критических скоростей роторов турбогенераторов (см. гл. 5). Решение этой задачи на машине Б-ЭСМ представляло весьма жесткие требования к объему программы, ввиду чего пришлось пойти на усложнение схемы программы. Ознакомление со сложной по краткой программой будет полезно читателю. Следуя общему методу описанному в § 4.1, решение задачи начинается с построения схемы счета. На первом этапе счета вычисляется значение функции D в точке ω_0 и значение $D(\omega_0)$ сохраняется. Затем ω_0 прибавляется шаг h и вычисляется значение D при новом значении аргумента $\omega = \omega_0 + h$. Если знак D не изменился, то вновь вычисленное значение функции сохраняется, к последнему значению аргумента прибавляется шаг h и знак сохраненного значения функции D сравнивается со знаком значения вычисленного при новом значении аргумента. Так поступают до тех пор, пока не дойдут до первого интервала $(\omega_i, \omega_i + h)$, на концах которого функция D имеет разные знаки. После этого счет ведется по следующим правилам (рис. 4-6).

1. При каждом новом этапе шаг делится пополам.

2. Два последовательных значения $D(\omega)$ сравниваются и вновь вычисленное значение отбрасывается, если произошла перемена знака и сохраняется в противном случае.

3. Если произошла перемена знака $D(\omega)$, то текущий шаг вычитается из последнего значения аргумента, если перемены знака не произошло, то шаг прибавляется.

Иными словами, в описанной схеме счета значение функции во вновь образованной точке сравнивается с последним из тех значений, которое имеет одинаковый знак со своим предыдущим.

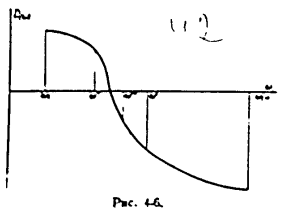


Рис. 4-6.

Для составления схемы программы можно ввести следующие операторы:

T — пересылка начальной точки ω_0 в рабочую ячейку;

R — восстановление шага;

P_i — подготовка счетчика I (подсчет числа точек, в которых вычисляется функция D);

P_j — подготовка счетчика J (подсчет числа делений);

P_r — подготовка положительного значения управляющего параметра r (при $r>0$ шаг не делится, при $r\leqslant 0$ делится);

$f(r)$ — параметру r придается отрицательное значение;

$f(D)$ — выработка управляющего параметра t (произведение двух значений D): при $t>0$ шаг прибавляется, при $t<0$ — вычитается;

D — вычисление функции $D(\omega)$;

T' — пересылка вычисленного значения функции D в рабочую ячейку;

I — деление текущего шага пополам;

B — прибавление текущего шага;

B' — вычитание текущего шага;

Π — печать результата;

$f(i)$ $f(j)$ — увеличение индексов i и j на единицу.

Теперь можно составить схему программы (рис. 4-7).

При составлении программы разбивать оператор D на отдельные команды не нужно, так как конкретный вид функции $D(\omega)$ сейчас не существенен. Предполагается, что программа вычисления значения функции D в некоторой точке хранится в группе ячеек, начиная с ячейки под номером p. Последняя команда этой программы возвращает счет к команде $k+10$.

Ввиду сложности программы следует сделать несколько замечаний.

Программа решения уравнения $D(\omega)=0$

$T\{k+1$	ПЧ	$\cdot\omega_0 - h^*$	—	β
$R\{k+2$	ПЧ	$\cdot h^*$	—	δ
$P_i\{k+3$	ПЧ	$\cdot \frac{1}{2}^*$	—	γ_i
$P_j\{k+4$	ПЧ	$\cdot \frac{1}{2}^*$	—	γ_j
$P_r\{k+5$	ПЧ	$\cdot 1^*$	—	γ_r
$\lambda_r\{k+6$	$<$	γ_r	$\cdot 0^*$	$k+15$
$T'\{k+7$	ПЧ	α	—	α'
$B'\{k+8$	$+$	β	δ	β
$k+9$	ПУ	ρ	—	—
$f(i)\{k+10$	$+$	$\cdot 1^*$	γ_i	γ_i
$\lambda_i\{k+11$	$<$	γ_i	$\cdot 2^*$	$k+7$
$f(j)\{k+12$	$\cdot$	ε	α'	γ_j
$\lambda_j\{k+13$	$<$	$\cdot 0^*$	γ_j	$k+6$
$f(r)\{k+14$	$-$ПЧ	$\cdot 1^*$	—	γ_r
$L\{k+15$		δ	$\cdot 2^*$	δ
$f(D)\{k+16$	$+$	$\cdot 1^*$	γ_j	γ_j
$\lambda_j\{k+17$	$<$	$\cdot n^*$	γ_j	$k+21$
$\lambda_t\{k+18$	$\swarrow$	$\cdot 0^*$	γ_t	$k+7$
$B'\{k+19$	$-$	δ	δ	β
$k+20$	ПУ	ρ	—	—
$\Pi\{k+21$	Печать	β	—	—
$k+22$	Стоп			

Передача управления к вычислению D

α — образование нового значения D;
α' — сохранение старого значения D;
β — образование текущего аргумента ω; в конце счета в ячейке β образуется корень;
δ — образование текущего шага;
γ_i — счетчик индекса i;
γ_j — счетчик индекса j;
γ_r — ячейка для хранения параметра r;
γ_t — ячейка для хранения параметра t.

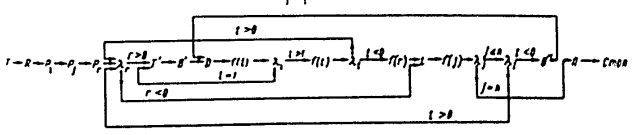


Рис. 4-7.

Во-первых, команда $k+1$ передает в ячейку β не начальное значение аргумента ω_0, а $\omega_0 - h$, так как оператор B' выполняется первый раз до оператора D. Во-вторых, в счетчики γ_j и γ_i передаются не нули, а половинки. Делается это для того, чтобы, например, в команде $k+11$ сравнивались не числа 2 и 2, а сначала числа $1\frac{1}{2}$ и 2,

а затем $2\frac{1}{2}$ и 2. В-третьих, при переходе к каждому новому значению аргумента в счетчик γ_i прибавляется единица, хотя нас интересует выделение лишь первой точки x_0. Таким образом, сравнение $k-11$ нужно только 1 раз, но во всяком случае дальнейшей работе программы оно не мешает. Условимся рассматривать всю приведенную программу за исключением команды $k+1$ как один оператор C. Этот оператор будет использован в следующей главе при составлении программы расчета критических скоростей роторов турбогенераторов.

4-5. Хранение функций в машине

Очень часто для решения задач техники и естествознания бывают нужны значения специальных функций таких, как тригонометрические, гиперболические функции Бесселя и т. п. При ручном счете значения этих функций берутся вычислителем из таблиц, пользование которыми при решении задач на автоматических машинах исключено. Поэтому в программу решения задачи приходится включать подпрограммы вычисления этих функций по каким-либо формулам. Например, в § 3-3 были составлены программы вычисления функций e^x и $\sin x$ по их разложениям в ряды по степеням x. В § 3-6 были составлены программы для вычисления функций $\mathrm{sh}x$, $\mathrm{ch}x$. Как указывалось в § 3-8, пользоваться такими формулами надо весьма осторожно, все время заботясь о предельном сокращении времени вычислений. Это особенно относится к таким задачам, решение которых связано с многократным использованием подпрограмм для специальных функций. Для сокращения времени вычислений во многих случаях весьма выгодно заменить одно значение аргумента другим, пользуясь каким-нибудь аналитическими соотношениями, как, например,

$$\sin(2k\pi + x) = \sin x,$$

$$\lg e^p x = p + \lg x$$

и т. п. (см. § 3-9).

Весьма выгодными в этом смысле являются формулы, представляющие специальные функции рядами по полиномам Чебышева, с успехом применяемые при составлении подпрограмм для машины БЭСМ. Иногда для вычисления значений функций при разных значениях аргумента выгодно пользоваться различными формулами.

Однако очень часто значения функций бывают заданы не в виде аналитических формул, а в виде таблиц или графиков, полученных экспериментальным путем. В последнем случае на основании этого графика необходимо составить таблицу, ибо машины могут оперировать только с числовыми данными. Чтобы на основании таблиц, задающих значения функций лишь при некоторых фиксированных значениях аргумента, определить значения функции при произвольном значении аргумента, необходимо пользоваться интерполяционными формулами. Последнее можно выполнить двумя различными способами. Если из условия задачи известно, что аргумент может изменяться в небольших пределах, или если известно, что функция изменяется достаточно плавно, то можно вычислять ее значения по одному и тому же интерполяционному полиному. В этом случае бывает выгодно заранее подсчитать коэффициенты интерполяционного полинома и, введя их в машину, вычислять значения полинома, пользуясь схемой Горнера, описанной в § 3-7. Если сделать это нельзя, то в машину вводится таблица значений функций и уже сама машина строит интерполяционный полином, по которому затем вычисляет значения функции. Основную трудность в этом случае представляет отыскание входа в таблицу по заданному значению аргумента x. Пусть в $n+1$ равноотстоящих точках x_0, $x_1 = x_0 + h$, $x_2 = x_0 + 2h$, ..., $x_n = x_0 + nh$ заданы значения функции y_0, y_1, ..., y_n. Они помещены соответственно в ячейки $l+0$, $l+1$, $l+2$, ..., $l+n$. Предположим, что используется интерполяционная формула k-того порядка, при которой значение функции в некоторой точке x выражается через табличные значения y_i, y_{i+1}, ..., y_{i+k}, где значение y_i соответствует точке x_i, удовлетворяющей неравенствам

$$x_i \leqslant x \leqslant x_{i+1}.$$

Стоит задача отыскания указанного значения y_i по заданному значению x, где $x_0 \leqslant x \leqslant x_{n-k}$. Рассмотрим разность $x - x_0$, равную длине отрезка (x_0, x). Целая часть отношения

$$\frac{x - x_0}{h}$$

равна числу точек $x_1, \ldots, x_i$, лежащих слева от точки x (рис. 4-8). Если к этому числу прибавить число l — номер ячейки, в которой хранится x_0, то число

$$\left[\frac{x - x_0}{h}\right] + l$$

будет равно номеру той ячейки $l+i$, в которой хранится y_i. Для составления программы, выбирающей и передающей число y_i в некоторую рабочую ячейку a, можно воспользоваться операцией взятия антье, которая помещает в младшие разряды третьего адреса целую часть числа.

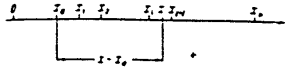


Рис. 4-8.

$k+1$	—	α^*	α_0^*	β
$k+2$		β	α^*	β
$k+3$	[ ]	β	γ	δ
$k+4$	—	δ	24	δ
$k+5$	СК	$k+s$	δ	$k+6$
$k+6$	ПЧ	$l+i$	—	a
$k+7$	ПЧ	l	—	a

В младшие разряды ячейки δ помещено $\left[\frac{x - x_0}{h}\right]$

Число $\left[\frac{x - x_0}{h}\right]$ помещено в младшие разряды 1-го адреса ячейки δ

В ячейке $k+s$ хранится команда, передающая содержимое ячейки l в ячейку a. При помощи операции сложения команд, в первый адрес ячейки $k+s$ прибавляется число i и результат помещается в ячейку $k+6$. Тем самым в ячейке $k+6$ формируется команда, передающая искомое число y_i в рабочую ячейку a. Вместе с тем первый адрес этой команды дает номер ячейки, в которой хранится это число y_i. Отыскание значений $y_{i+1}, y_{i+2}, \ldots, y_{i+k}$, нужных для построения интерполяционной формулы, уже не представляет труда. Также и в этом случае программа вычисления интерполяционного полинома может исходить из представления этого полинома как при помощи конечных разностей, которые строит сама машина, так и непосредственно через значения функции y_i, $y_{i+1}, \ldots, y_{i+k}$. Обычно последний способ несколько экономнее с точки зрения числа команд.

Пусть ту же задачу нужно выполнить на машине с фиксированной запятой и пусть все числа хранятся в запоминающем устройстве машины с некоторым масштабом $\frac{1}{M}$. Если образовать вновь отношение

$$\frac{1}{M} \cdot \frac{x - x_0}{h},$$

умножить это число на $\frac{1}{2^p}$ и разделить на $\frac{1}{M}$, то получается число

$$x' = \frac{1}{2^p} \cdot \frac{x - x_0}{h},$$

где p выбрано так, что $\frac{1}{2^{p+1}} < \frac{1}{M} \leqslant \frac{1}{2^p}$. Заменой масштаба $\frac{1}{M}$ на $\frac{1}{2^p}$ достигнуто то, теперь целая часть

числа x' хранится в p старших разрядах ячейки. Путем выделения ее при помощи операции логического умножения на число

$$\underbrace{11\ldots100\ldots0}_{p}$$

и сдвига на нужное число разрядов целая часть числа x' помещается в младшие разряды первого адреса. Дальнейшее построение программы будет совершенно аналогично предыдущему случаю.

Табличный способ задания используется иногда для функций, для которых аналитические формулы достаточно громоздки или требуют большого числа действий. Его особенно удобно применять в случае, когда не требуется высокая точность вычислений или известно, что аргумент функции изменяется в небольших пределах.

Пусть, наконец, значения функции заданы в $n+1$ неравноотстоящих точках. Чтобы найти в этом случае вход в таблицу, нужно поместить в запоминающее устройство машины и значения функции и значения аргумента. Пусть точки $x_0, x_1, \ldots, x_n$ помещены в ячейки $l, l+1, \ldots, l+n$, а соответствующие значения функции — в ячейки $l+n+1, l+n+2, \ldots, l+2n+1$. Чтобы найти по заданному x вход в такую таблицу, нужно последовательно сравнивать x с точками $x_0, x_1, x_2, \ldots$, пока не будет найдена та первая точка x_i, для которой $x_i \leqslant x$. Выделением адреса этой точки и прибавлением к нему $n+1$ единиц соответствующего адреса определяется номер ячейки, в которой хранится y_i.

ГЛАВА ПЯТАЯ

ОПРЕДЕЛЕНИЕ КРИТИЧЕСКИХ СКОРОСТЕЙ РОТОРОВ ТУРБОГЕНЕРАТОРОВ

5-1 Постановка задачи

Как известно, при быстром вращении вала имеет место следующее явление: при увеличении угловой скорости ω она достигает такого значения $\omega_{кр}$, при котором вал не сохраняет прямолинейной формы и начинает гнуться, так что дальнейшем увеличении ω это явление прекращается, а затем опять возникает при $\omega = \omega_{кр}$ и т. д. Скорости $\omega_{кр}, \omega_{кр}, \omega_{кр}, \ldots$ называются критическими и при расчете различных вращающихся валов важно предусмотреть такие их размеры, при которых рабочая скорость вращения отличается от критических на 20—30%. В настоящей главе излагается расчет на АЦВМ критических скоростей роторов мощных турбогенераторов в зависимости от частоты собственных колебаний опор, определяемых экспериментальным путем [Л. 21].

Биение вала объясняется тем, что практически невозможно достигнуть полного совпадения геометрической оси вала с одной из главных осей инерции, в результате чего при вращении вала возникают возмущающие силы. Когда частота этих сил, численно равная числу оборотов вала в секунду, становится равной собственной частоте боковых колебаний вала, тогда наступает явление резонанса и вал начинает колебаться. Итак, критические скорости вала — это такие скорости, при которых число оборотов вала в секунду равно собственной частоте поперечных колебаний вала.

Можно рассматривать ротор турбогенератора как стержень, состоящий из n прямых круговых цилиндров различных диаметров с общей осью (рис. 5-1). Направим ось x по оси стержня и через $y_i(x, t)$ обозначим отклонение оси i-того цилиндра в точке x в момент времени t.

Пусть опоры вала помещены в точках $x=0$ и $x=l$, граница между i-тым и $i+1$-м участками находится в точке $x=l_i$ и $s=\frac{x}{l}$. Если B_i — жесткость и m_i — масса i-того цилиндра

[1] Задача была поставлена сотрудником НИИЭП Госплана СССР канд. физ.-мат. наук И. Е. Сазаровым.

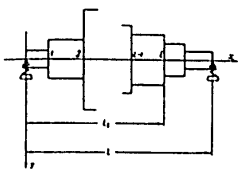


Рис. 5-1

то уравнение его поперечных колебаний имеет вид[1]

$$a_i^4 \frac{\partial^4 y_i}{\partial x^4} + \frac{\partial^2 y_i}{\partial t^2} = 0, \quad a_i^4 = \frac{B_i}{l^4 m_i}. \quad (5\text{-}1)$$

Собственные колебания системы ротор—опоры происходят по закону

$$y = S(s)\sin\omega t, \quad (5\text{-}2)$$

$$y_л = S_л \sin\omega t, \quad y_{нр} = S_{нр}\sin\omega t, \quad (5\text{-}3)$$

где ω — собственная частота колебаний системы ротор—опоры, а $y_л$ и $y_{нр}$ — перемещения центров тяжести опор. Уравнение (5-1) после подстановки в него (5-2) получает вид

$$S_i^{IV} - \frac{\omega^2}{a_i^4} S_i = 0.$$

Будем искать решение этого уравнения в виде

$$S_i(s) = A_i \operatorname{ch}\frac{s}{a_i}\sqrt{\omega} + B_i \operatorname{sh}\frac{s}{a_i}\sqrt{\omega} + C_i \cos\frac{s}{a_i}\sqrt{\omega} + D_i \sin\frac{s}{a_i}\sqrt{\omega},$$

$$s_{i-1} \leqslant s \leqslant s_i;\ s_i = \frac{l_i}{l}. \quad (5\text{-}4)$$

Константы $s_л$, $s_{нр}$, A_i, B_i, C_i, D_i можно определить из граничных условий, которые для границы s_i между i-тым и $i+1$-м участками выражаются в равенстве прогибов, углов наклона касательных к оси, изгибающих моментов и перерезывающих сил:

$$S_i = S_{i+1};$$

$$S_i' = S_{i+1}';$$

$$B_i S_i'' = B_{i+1} S_{i+1}'';$$

$$B_i S_i''' = B_{i+1} S_{i+1}'''. \quad (5\text{-}5)$$

Граничные условия на опорах выражаются в равенстве прогибов ротора и перемещений опор, отсутствии изгибающих моментов на концах ротора и равенства перерезывающих сил по концам ротора силам, действующим на ротор со стороны опор. При этом опоры рассматриваются как колебательные системы с одной степенью свободы. Граничные условия имеют вид:

на левой опоре ($s=0$)

$$S_1 = S_л;$$

$$S_1'' = 0;$$

$$-B_1 S_0''' = a_л S_л \left(1 - \frac{\omega^2}{\omega_л^2}\right); \quad (5\text{-}6)$$

на правой опоре

$$S_n = S_{нр};$$

$$S_n'' = 0;$$

$$B_n S_n''' = a_{нр} S_{нр}\left(1 - \frac{\omega^2}{\omega_{нр}^2}\right), \quad (5\text{-}7)$$

где $a_л$, $a_{нр}$ — константы, зависящие от физических свойств опор и ротора, а $\omega_л$ и $\omega_{нр}$ — собственные частоты колебаний опор ($\omega_л = \sqrt{1/\delta_л M_л}$, $\omega_{нр} = \sqrt{1/\delta_{нр} M_{нр}}$, δ и M — соответственно податливости и массы опор). При подстановке формул (5-3) и (5-4) в граничные условия (5-5), (5-6) и (5-7), получаются $4n+2$ однородных уравнения для определения $4n+2$ неизвестных $S_л$, $S_{нр}$, A_i, B_i, C_i, D_i, $i=1, 2, \ldots, n$. Для существования нетривиального решения этой системы ее определитель, рассматриваемый как функция частоты ω, должен равняться нулю.

При расчете критических скоростей ротора турбогенератора мощностью 150 тыс. *квт* критические ско-

[1] См., например, Смирнов, Курс высшей математики, т. II, § 189.

рости ротора определялись как корни уравнения

$$D(\omega, \omega_x, a_x, \omega_{xp}, a_{xp}) = 0, \quad (5\text{-}8)$$

являющегося определителем 46-го порядка (табл 5-1 см. вклейку в конце книги). Здесь ω_x, a_x, ω_{xp}, a_{xp} — константы, зависящие при данном роторе от динамических податливостей опор. Придавая ω_x, a_x, ω_{xp}, a_{xp} различные значения, можно найти зависимость критических скоростей ротора от физических свойств опор.

5-2. Решение задачи на машине БЭСМ

При проведении расчетного исследования рассматривались 7 различных опор, что приводило к 28 различным вариантам задачи т е были заданы 2^8 групп численных значений параметров

$$\omega_x^i, \quad a_x^i, \quad \omega_{xp}^i, \quad a_{xp}^i,$$

для которых требовалось решить уравнение

$$D(\omega, \omega_x^i, a_x^i, \omega_{xp}^i, a_{xp}^i) = 0. \quad (5\text{-}9)$$

Приближенное отыскание корней такого уравнения является, вообще говоря, весьма трудной задачей. В данном случае, однако, задача сильно упрощается. Из физических соображений известно, что, во-первых, наименьшая собственная частота превосходит 50 $сек^{-1}$ и, во-вторых, что две последовательные собственные частоты отстоят друг от друга на расстоянии, большем, чем 25 $сек^{-1}$. Кроме того, практический интерес представляют лишь первые четыре собственные частоты. Благодаря этому можно решить уравнение $D(\omega)=0$ методом, описанным в § 4-4, положив шаг $h = 25$. Допустимая в расчетах ошибка составляет 1% истинного значения, т. е. превышает 0,5, так как минимальный корень больше 50. Значит после семикратного деления шага h пополам ошибка станет равной $\frac{h}{2^7} = \frac{25}{128}$, т е достигнет требуемой точности Начиная вычисления с $\omega_0 = 50$, определяем четыре последовательных корня для одного варианта задачи, после чего переходим к следующему, пока не будут просчитаны все 28 вариантов

Схема счета будет иметь вид

$$\prod_{i=1}^{28} T_i \prod_{k=1}^{4} C_k,$$

где C_k — оператор, отыскивающий k-тый корень уравнения $D(\omega) = 0$,

T_i — оператор, передающий константы ω_x^i, a_x^i, ω_{xp}^i, a_{xp}^i в рабочие ячейки.

При составлении программы можно воспользоваться оператором C, введенным в конце § 4-4. Если в схеме этого оператора (рис. 4-4) заменить команду *стоп* командой, передающей управление к оператору R, то по такой программе можно отыскивать корни уравнения $D(\omega) = 0$. Оператору C придан индекс k, хотя его команды от этого индекса не зависят. Схема программы, представленная на рис. 5-2, содержит следующие операторы.

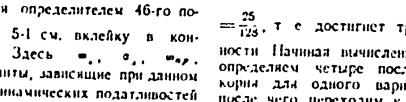


Рис 5-2.

R_i — восстановление оператора T_i;

T_i — передача ω_x^i, a_x^i, ω_{xp}^i, a_{xp}^i в рабочие ячейки;

P_k — подготовка счетчика k (число вычисленных корней),

T — восстановление начальной точки ω_0,

$I(k)$ — прибавление 1 в счетчик k;

λ_k — сравнение k с 4;

F_i — преобразование оператора T_i;

λ_i — сравнение i с 28 (сравнение переменной команды со стандартной).

Схема оператора C_k содержит оператор D, вычисляющий значение функции $D(\omega)$ в любой заданной точке. В § 4-4 вид этой функции не был конкретизирован и поэтому оператор D не расписывался на отдельные команды. Но в настоящем случае наиболее трудной частью всей задачи является именно программа вычисления значения функции $D(\omega)$ *.

Вообще говоря, вычисление значения определителя сводится к следующему под каждый элемент определителя отводится отдельная ячейка запоминающего устройства, затем вычисляется значение этих элементов при заданном ω, полученные значения вписываются в соответствующие ячейки и определитель вычисляется одним из известных способов Удобно представить оператор D в виде двух операторов D' и D'' D' вычисляет элементы определителя и расписывает их по соответствующим ячейкам памяти, а D'' вычисляет сам определитель, начала приводится описание оператора D''.

Как указывалось выше, настоящая задача была решена на машине БЭСМ, оперативная память которой содержит 1 024 ячейки Между тем число элементов определителя равняется $46^2 = 2\,116$. Если отвести под каждый элемент определителя отдельную ячейку памяти, то для вычисления значения определителя в одной точке пришлось бы многократно обращаться к внешнему запоминающему устройству, что, во-первых, усложнило бы программу и, во-вторых, сильно увеличило бы время решения задачи. Для вычисления значения определителя был использован метод Гаусса, описанный в § 4-3 Так как в определителе из 2 116 элементов отличными от нуля являются лишь 350, то отпала необходимость отводить под каждый элемент отдельную ячейку. Вместе с тем равенство нулю большей части элементов определителя обеспечивает быструю реализацию метода Гаусса. Фактически под элементы определителя было отведено 586 ячеек, указанных в табл. 5-2. Такое распределение ячеек объясняется следующим. Чтобы избежать деления на числа, близкие нулю, описанный в § 4-3 метод был несколько изменен. Перед тем как делить элементы l-той строки на a_{ll}, среди них выбирался наибольший по модулю элемент a_{il} и i-тый столбец переставлялся с k-тым (такая перестановка изменяет знак определителя на обратный). В силу такой перестановки в каждом столбце надо было предусмотреть семь ячеек для элементов, лежащих ниже диагонального. Действительно, если, например, в четвертой строке самым большим является элемент, стоящий в 45-й клетке, то четвертый столбец переставляется с девятым и в четвертом столбце под диагональным элементом образуются семь отличных от нуля элементов. Но, начиная с 40-го столбца, подте же элементы достаточно отвести $46 - l$ ячеек, где l — номер столбца. Итак, в l-том столбце надо обратить в нуль число элементов, равное $\min\{7, 46 - l\}$. Далее: вычитание из элементов j-той строки соответствующих элементов l-той строки имеет смысл выполнять лишь для тех из них, которые отличны от нуля. В каждой строке справа от диагонального элемента может лежать не более пяти отличных от нуля элементов. Именно такое число ячеек отведено под эти элементы в принятом распределении памяти. Чтобы понять назначение ячеек, лежащих вне самого определителя, нам нужно обратиться к программе вычисления определителя, приведенной в § 4-3.

В ней все преобразования команд совершаются при помощи констант, содержащих n (или $n+1$) единиц того или иного адреса, где n — порядок определителя. Это связано с тем, что два любых последовательных диагональных элемента определителя находятся в ячейках, номера которых отличаются на $n+1$ единиц. В рассматриваемом определителе адреса двух последовательных диагональных элементов, лежащих в столбцах от 7-го до 42-го, отличаются на 13 единиц. Чтобы сохранить это число постоянным и тем самым упростить всю систему переадресации, под каждую строку опре-

* В составлении программы принимал участие мл. научный сотрудник Института математики и механики АН Арм. ССР Г. Геворкян.

зателя отведено по 13 ячеек (кроме первой и последней, где ячейки, лежащие вне определителя, не нужны). В ячейках, лежащих вне определителя, могут, очевидно, находиться любые числа, так как с ними мы вообще никаких действий не производим.

После всего сказанного ясно, чем отличается программа вычисления определителя в настоящей задаче от общей схемы, разобранной в § 4.3. Во-первых, как было указано, в каждой строке выбирается наибольший элемент, затем определяется его адрес (такая программа приводилась в § 4.5), после чего столбец, в котором находится этот элемент, переставляется со столбцом, содержащим диагональный элемент. Во-вторых, во всех командах пересылки вместо числа n (в данном случае $n=46$) фигурирует число 13. Наконец, в третьих, обращать надо в нуль лишь число элементов, равное $\min\{7\cdot 46-i, 46\}$, где i — номер столбца, а вычитать из каждой строки пять элементов (последовательно, переходя от одного столбца к другому, нужно каждый раз подсчитывать число $\min\{7\cdot 46-i, 46\}$ и использовать его в качестве константы сравнения). Точно так же другой константой сравнения будет не n, как в § 4.4, а 5. При этом обнулять надо каждый раз очищать. Можно, конечно, и расписывать по командам схему этого метода вычисления определителя, так как она ничем принципиально не отличается от приведенной в § 4.3.

Остается составить схему оператора D, вычисляющего значения самих элементов определителя и расставляющего их по соответствующим ячейкам памяти. Сначала оператор G_1 (рис. 5-3) перетащит во все 586 ячеек нуль. Тем самым очищаются лишние ячейки, но зато такую очистку можно выполнить в четыре команды. Между тем программа, передающая нули лишь в нужные ячейки, была бы очень громоздка и сложна. Далее, группы по 16 элементов, находящиеся в табл. 5-2 в квадратах I, III, ..., XIX, отличаются друг от друга лишь коэффициентами s_i/a_i, $i=1,2,\ldots,10$, стоящими под знаками функций sin, cos, sh и ch. Точно так же группы по 16 элементов, стоящие в квадратах II, IV, ..., XX, отличаются только коэффициентами, стоящими под знаками этих функций и перед ними. Вычисление и расстановка элементов этих 20 квадратов производится следующим способом. Константы $\frac{s_1}{a_1}\cdot\frac{s_2}{a_2}\cdot\frac{s_3}{a_3}\cdot\frac{s_4}{a_4}\cdot\ldots\cdot\frac{s_{10}}{a_{10}}\cdot\frac{s_{11}}{a_{11}}$ помещаются в указанном здесь порядке в 21 последовательную ячейку памяти. Точно так же в 30 последовательных ячейках памяти помещаются константы $\frac{a_1}{a_2}\cdot\left(\frac{a_2}{a_3}\right)^2\frac{B_2}{B_1}\cdot\left(\frac{a_1}{a_3}\right)^3\frac{B_3}{B_1}\cdot\ldots\cdot\frac{a_{10}}{a_{11}}\cdot\left(\frac{a_{10}}{a_{11}}\right)^9\frac{B_{10}}{B_{10}}\cdot\left(\frac{a_{10}}{a_{11}}\right)^{10}\frac{B_{11}}{B_{10}}$.

Текущее значение аргумента ω передается в подпрограмму вычисления корня, которую мы обозначим Γ^{ω}. Далее, оператор T_l умножает либо $\frac{s_l}{a_l}$, либо $\frac{s_{l-1}}{a_{l-1}}$ на $1-\omega$ и управление передается к оператору Γ^{ω}, вычисляющему значения функций sin, cos, sh, ch.

Через A_l, $l=1,2,\ldots,10$ обозначен оператор, передающий элементы квадратов I, III, ..., XIX в соответствующие ячейки, а через A''_l, $l=1,2,\ldots,10$ — оператор, передающий элементы квадратов II, IV, ..., XX. После оператора Γ^{ω} поставлен управляющий параметр r. В зависимости от знака этого параметра счет направляется либо к оператору A_l, либо к оператору K''_l. В последнем случае перед оператором K''_l стоит оператор T_{11}, передающий константы $\frac{c_l}{a_{l+1}}\cdot\left(\frac{a_l}{a_{l+1}}\right)^2\frac{B_{l-1}}{B_l}\cdot\left(\frac{a_l}{a_{l+1}}\right)^3\times\frac{B_{l-1}}{B_l}$ в рабочие ячейки.

Следует подробнее описать работу оператора K'_l, его преобразование и восстановление. Пусть оператор Γ^{ω} помещает значения функций sin, cos, sh, ch соответственно в ячейки p_1, p_2, p_3, p_4. Оператор K'_l состоит из 16 команд, первоначальный вид которых выписан в ячейках $n+1$, $n+2$, ..., $n+16$ (в третьем адресе их стоят номера ячеек, соответствующих элементам определителя согласно табл. 5-2).

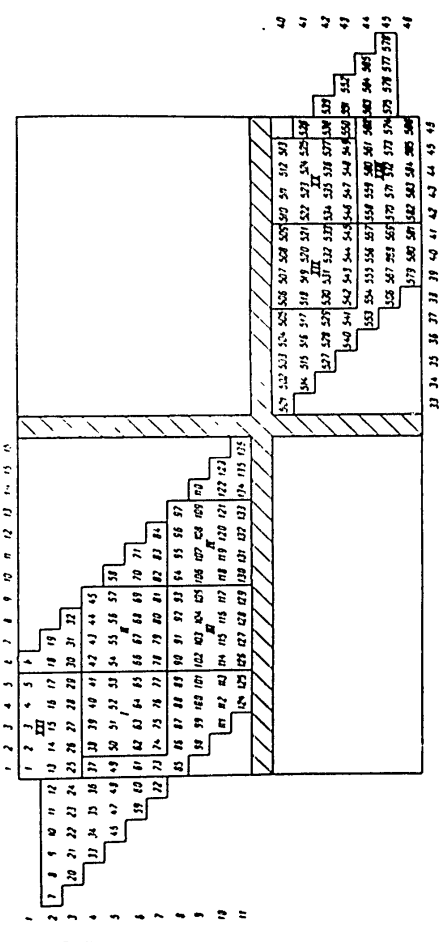


Таблица 5-2

$n+1$	ПЧ	p_1	—	38
$n+2$	ПЧ	p_1		39
. . .	. . .	. . .	. . .	. . .
$n+16$	ПЧ	p_1	—	77

Схема D' начинается с оператора R, передающего первоначальный вид этих команд на их рабочие места в программе, которые обозначены через $k+1$, $k+2$, . . . $k+16$. Этот оператор содержит пять команд (см. рис. 5-3)

$l+1$	ПЧ	0	—	a
$l+2$	ПЧ	$n+1$	—	$k+1$
$l+3$	СК	$l+2$	.11	///A• $l+2$
$l+4$	+	.1•	2	a
$l+5$	<	2	16	$l+2$

Таким образом, в $k+1 \div k+16$ будет помещен оператор K_1', который расшлет по своим ячейкам элементы квадрата I. Теперь оператор K_1' надо

Схема содержит следующие операторы

R — восстановление операторов G, T_j, K_i', K_i'' и параметра r,

G_i — очистка ячеек,

Γ' — вычисление корня,

T_j — передача $\frac{s_j}{a_j}$ или $\frac{s_j}{a_{j+1}}$ в рабочую ячейку,

Γ'' — вычисление sin, cos, sh, ch,

$I(r)$ — перемена знака управляющего параметра на обратный (вначале $r<0$)

K_1' — роспись элементов, входящих в квадраты I, III, . . XIX,

K_1' — роспись элементов, входящих в квадраты II, IV, . . XX

L — роспись элементов, входящих в прямоугольники XXI и $XXII$,

I — преобразование соответствующих операторов

λ — проверка числа выполненных циклов,

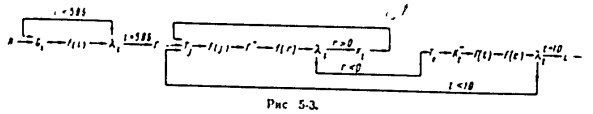


Рис 5-3.

преобразовать к виду K_3', при котором он будет расписывать элементы квадрата III. Для этого к третьему адресу ячеек $k+1$, $k+2$, . . . $k+16$ надо прибавить 52 единицы ибо тогда, например, команда $k+1$ примет вид

ПЧ $p_4 - 90$.

Это преобразование выполняет оператор $F(t)$, который может быть составлен так же, как и оператор R. Аналогично составляется и оператор K_i'', причем оператору R поручается одновременно восстановление K_i' и всех остальных команд, входящих в оператор D', а оператору $F(t)$ — преобразование K_i''. Остается поместить на свои места элементы, входящие в прямоугольники, помеченные на табл. 5-2 номерами XXI и $XXII$. Это выполняет оператор L, программирование которого не представляет никакого труда, а включение его в программу становится ясным из приводимой на рис. 5-3 схемы оператора D'.

Составлением схем операторов D' и D'' заканчивается составление всей программы. Как видно из содержания этого параграфа, составление схемы программы настоящей задачи выполнялось в несколько этапов. Сначала составлялась укрупненная схема, в которую входили такие операторы, как оператор D вычисления значения определителя. Этот оператор был представлен в виде двух операторов D' и D'', для каждого из которых, в свою очередь, составлялась схема программы. Точно так же и саму программу можно разбить на ряд отдельных программ, каждая из которых описывает один оператор и составлена в условных адресах и ячейках. После этого производится распределение памяти и из программ отдельных операторов составляется или, как говорят, монтируется общая программа всей задачи, в которой уже вместо условных адресов фигурируют конкретные ячейки памяти.

Остается рассказать, как фактически была распределена память в этой задаче. Первые пять ячеек в запоминающем устройстве машины БЭСМ, занятые под оперативные ячейки стандартных подпрограмм, не могли быть использованы. Элементы определителя были помещены в ячейки 11 – 596. Сама программа содержала 251 команду и размещалась в ячейках 601—851. Стандартные команды и константы, используемые в программе (константы сравнения, переадресации и т. д.), занимали 81 ячейку с 861 по 941. Константы $\frac{a_i}{a_{i+1}}$, $\frac{s_i}{a_i}$, $\frac{s_i}{a_{i+1}}$, $\left(\frac{a_i}{a_{i+1}}\right)^2$, $\frac{B_{i+1}}{B_i}$, $\left(\frac{a_i}{a_{i+1}}\right)^2 \frac{B_{i+1}}{B_i}$ занимали 51 ячейку с 950 до 1 000.

Из оставшихся 50 ячеек 31 была использована под рабочие, 112 констант $\omega_{x_i}^i$, $a_{x_i}^i$, $\omega_{n_i p}^i$, $a_{n_i p}^i$ были размещены во внешнем запоминающем устройстве, к которому, следовательно, пришлось обращаться всего 28 раз (оператор T_j в схеме программы на рис. 5-2). Расчет каждого варианта на БЭСМ продолжался около 2 *мин*. За это время машина выполняла около миллиона операций.

ГЛАВА ШЕСТАЯ

РАСЧЕТ УСТОЙЧИВОСТИ СИСТЕМ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ НА ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

6-1. Основные сведения

Одним из основных вопросов, возникающих при проектировании систем автоматического регулирования и управления, является обеспечение их устойчивости. Расчет и анализ устойчивости сложных динамических систем требует выполнения большого объема вычислительной работы. Для расчета и исследования устойчивости могут быть эффективно использованы автоматические цифровые вычислительные машины.

Впервые методика расчета границ областей устойчивости на автоматической вычислительной машине была разработана в АН УССР [Л. 24]. В настоящей главе описывается программа, употребляемая в вычислительной лаборатории ИИИЭП Госплана при расчетах устойчивости систем автоматического регулирования на машине М-3 [Л. 10]. При составлении этой программы использован ряд характерных особенностей методики АН УССР. В то же время описываемая ниже программа имеет и некоторые существенные отличия.

Основой современных методов исследования устойчивости динамических систем является теория устойчивости движения Ляпунова [Л. 19]. Движение динамической системы может быть описано системой дифференциальных уравнений:

$$\left.\begin{aligned} \frac{dx_1}{dt} &= f_1(x_1, x_2, \ldots, x_n); \\ \frac{dx_2}{dt} &= f_2(x_1, x_2, \ldots, x_n); \\ &\ldots\ldots\ldots\ldots \\ \frac{dx_n}{dt} &= f_n(x_1, x_2, \ldots, x_n). \end{aligned}\right\} \quad (6\text{-}1)$$

где $x_1, x_2, \ldots, x_n$ — отклонения переменных (обобщенных координат — углов, токов, напряжений и т. д.) от их значений в установившемся режиме; $f_1, f_2, \ldots, f_n$ — известные, в общем случае нелинейные, функции отклонений переменных $x_1, x_2, \ldots, x_n$. Уравнения (6-1) названы Ляпуновым уравнениями возмущенного движения.

В установившемся режиме отклонения переменных и их производные равны нулю, т. е.

$$x_1 = 0;\ x_2 = 0;\ \ldots;\ x_n = 0 \quad (6\text{-}2)$$

и

$$\left.\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0; \\ f_2(x_1, x_2, \ldots, x_n) &= 0; \\ &\ldots\ldots\ldots \\ f_n(x_1, x_2, \ldots, x_n) &= 0. \end{aligned}\right\} \quad (6\text{-}3)$$

Нулевое решение (6-2) системы дифференциальных уравнений (6-1) в теории Ляпунова называется невозмущенным движением. Невозмущенное движение (6-1) называется устойчивым по отношению к величинам $x_1, x_2, \ldots, x_n$, если при любом произвольно задаваемом положительном числе ε, как бы мало оно ни было, может быть выбрано другое положительное число $\gamma(\varepsilon)$ так, чтобы при всяких возмущениях $x_{10}, \ldots, x_{n0}$, удовлетворяющих условиям

$$|x_{i0}| < \gamma, \quad (6\text{-}4)$$

для возмущенного движения при любом $t > 0$ выполнялось неравенство

$$|x_i(t)| < \varepsilon \quad (6\text{-}5)$$

В противном случае невозмущенное движение (6-2) неустойчиво.

Невозмущенное движение называется асимптотически устойчивым, если наряду с неравенствами (6-4) и (6-5) справедливо также равенство

$$\lim_{t \to \infty} x_i = 0 \quad (i = 1, \ldots, n) \quad (6\text{-}6)$$

Существуют методы, позволяющие анализировать устойчивость динамической системы, не требуя при этом интегрирования дифференциальных уравнений (6-1). Наиболее мощным из них является так называемый прямой метод Ляпунова (или вторая метода Ляпунова), основанный на свойствах некоторых функций переменных $x_1, \ldots, x_n$ («функций Ляпунова»). Прямой метод Ляпунова позволяет исследовать устойчивость нелинейных динамических систем при любых больших отклонениях переменных, если только при этом сохраняют силу исходные дифференциальные уравнения системы (6-1).

Однако во многих задачах отыскание функций Ляпунова, необходимых для анализа устойчивости прямым методом, связано с большими трудностями. Поэтому часто ограничиваются рассмотрением устойчивости системы [устойчивости невозмущенного движения (6-2)] при малых отклонениях переменных, рассматривая линеаризованные дифференциальные уравнения системы («уравнения первого приближения»).

Во многих практически важных случаях функции в правых частях уравнений динамической системы (6-1) могут быть разложены в ряды по целым положительным степеням отклонений переменных $x_1, \ldots, x_n$, если эти отклонения достаточно малы.

Тогда уравнения (6-1) принимают вид

$$\begin{aligned}
\frac{dx_1}{dt} &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + N_1(x_1, \ldots, x_n),\\
\frac{dx_2}{dt} &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + N_2(x_1, \ldots, x_n),\\
&\ldots\\
\frac{dx_n}{dt} &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + N_n(x_1, \ldots, x_n),
\end{aligned} \quad (6\text{-}7)$$

где $N_i(x_1, \ldots, x_n)$ — функции $x_1, x_2, \ldots, x_n$, не содержащие членов ниже второго порядка малости;

$a_{11}, \ldots, a_{nn}$ — постоянные, причем

$$a_{ik} = \left(\frac{\partial f_i}{\partial x_k}\right)_{x_1 = x_2 = \ldots = x_n = 0} \quad (6\text{-}8)$$

Линеаризованные уравнения системы или уравнения первого приближения

$$\begin{aligned}
\frac{dx_1}{dt} &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n,\\
\frac{dx_2}{dt} &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n,\\
&\ldots\\
\frac{dx_n}{dt} &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n
\end{aligned} \quad (6\text{-}9)$$

получаются из (6-7) отбрасыванием в правых частях членов выше первого порядка малости.

Системе дифференциальных уравнений (6-9) соответствует характеристическое уравнение

$$\lambda^n + p_1\lambda^{n-1} + \ldots + p_{n-1}\lambda + p_n = 0, \quad (6\text{-}10)$$

коэффициенты которого могут быть получены путем раскрытия характеристического определителя

$$\begin{vmatrix}
a_{11} - \lambda & a_{12} & a_{13} & \ldots & a_{1n}\\
a_{21} & a_{22} - \lambda & a_{23} & & a_{2n}\\
a_{31} & a_{32} & a_{33} - \lambda & & a_{3n}\\
\ldots & \ldots & \ldots & \ldots & \ldots\\
a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} - \lambda
\end{vmatrix}, \quad (6\text{-}11)$$

составленного из коэффициентов системы уравнений (6-9).

А. М. Ляпунов впервые строго определил условия, при которых допустимо исследование устойчивости невозмущенного движения (6-1) по уравнениям первого приближения. Согласно Ляпунову, если линеаризованные дифференциальные уравнения (уравнения первого приближения) системы имеют характеристическое уравнение, вещественные части корней которого отрицательны, то невозмущенное движение $x_1 = 0; x_2 = 0, \ldots, x_n = 0$ асимптотически устойчиво независимо от членов $N_i(x_1, x_2, \ldots, x_n)$ выше первого порядка малости в уравнениях (6-7). Если же хотя бы один корень имеет положительную вещественную часть, то невозмущенное движение неустойчиво независимо от членов $N_i(x_1, x_2, \ldots, x_n)$ выше первого порядка малости.

Возможен случай, когда вещественная часть у одного или нескольких корней равна нулю, а у остальных — отрицательна. Тогда об устойчивости системы нельзя судить по уравнениям первого приближения и требуется учитывать нелинейные члены $N_i(x_1, x_2, \ldots, x_n)$.

Существуют критерии, позволяющие определить отрицательность вещественных частей всех корней характеристического уравнения без решения уравнения. Широко известны необходимые и достаточные критерии отрицательности вещественных частей корней уравнения вида (6-10): алгебраические критерии Гурвица, Рауса, частотные критерии Найквиста, Михайлова и др.

На цифровой вычислительной машине удобнее расчет устойчивости производить по алгебраическим критериям, так как при этом выполняются операции лишь с вещественными числами, а суждение об устойчивости выносится в зависимости от знака чисел, получаемых при расчете.

Из алгебраических критериев для уравнений выше четвертой степени удобнее пользоваться критерием Рауса из-за единообразия вычислений. Следует привести формулировку этого критерия, так как в дальнейшем он будет использован.

Пусть линеаризованная система имеет характеристическое уравнение (6-10) n-ной степени.

Если выполнить вычисления для $n+1$ строк табл. 6-1, то согласно критерию Рауса, для того чтобы все корни уравнения (6-10) имели отрицательные вещественные части, необходимо и достаточно, чтобы были положительными все числа, стоящие в первом столбце таблицы ($p_1, b_1, c_1, \ldots$).

Таблица 6-1

$p_0 = 1$	p_2	p_4	p_6	. . . . .
p_1	p_3	p_5	p_7	. . . . .
$b_1 = \frac{p_1p_2 - p_0p_3}{p_1}$	$b_2 = \frac{p_1p_4 - p_0p_5}{p_1}$	$b_3 = \frac{p_1p_6 - p_0p_7}{p_1}$	$b_4 = \frac{p_1p_8 - p_0p_9}{p_1}$	. . . . .
$c_1 = \frac{b_1p_3 - p_1b_2}{b_1}$	$c_2 = \frac{b_1p_5 - p_1b_3}{b_1}$	$c_3 = \frac{b_1p_7 - p_1b_4}{b_1}$	$c_4 = \frac{b_1p_9 - p_1b_5}{b_1}$	. . . . .
$d_1 = \frac{c_1b_2 - b_1c_2}{c_1}$	$d_2 = \frac{c_1b_3 - b_1c_3}{c_1}$	$d_3 = \frac{c_1b_4 - b_1c_4}{c_1}$	$d_4 = \frac{c_1b_5 - b_1c_5}{c_1}$	. . . . .
. . . .	. . . .	. . . . .	. . . . .	. . . . .

Обычно при исследовании устойчивости динамической системы нужно бывает найти пределы изменения величины одного или нескольких конструктивных параметров (коэффициенты усиления, постоянные времени и др), при которых система сохраняет устойчивость. В этих случаях возникает задача определения границы области устойчивости в плоскости двух параметров, от которых зависят (в том числе и нелинейно) коэффициенты дифференциальных уравнений (6.9) и коэффициенты характеристического уравнения

Пусть, например, кривая *1* на рис. 6.1 есть граница области устойчивости в плоскости двух параметров δ, μ и всем точкам, расположенным внутри этой области, соответствуют значения δ, μ, при которых система устойчива

Значения δ и μ следует выбирать так, чтобы получить некоторый запас по устойчивости и быстрое затухание переходного процесса в системе

Я З Цыпкин и П В Бромберг ввели понятие о степени устойчивости, которой называется величина наименьшей из вещественных частей корней характеристического уравнения или, иначе говоря, расстояние h от мнимой оси в плоскости корней (рис 6.2) до ближайшего к ней корня характеристического уравнения [Л 25] В ряде случаев степень устойчивости может приблизительно характеризовать быстроту затухания переходного процесса

Правильный выбор конструктивных параметров облегчается, если внутри области устойчивости построить линии, соответствующие значениям δ и μ, при которых степень устойчивости имеет величины $h=h_1$, h_2, На границе об-

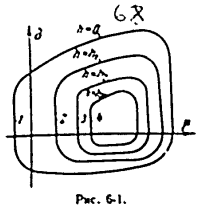


Рис. 6-1.

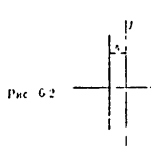


Рис 6.2

ласти устойчивости степень устойчивости $h=0$ (рис 6.1)

Линия, соединяющая точки с равной степенью устойчивости h, может определяться по тем же критериям что и границы области устойчивости если за исходные принять уравнения

$$\begin{aligned}
\frac{dx_1}{dt} &= (a_{11}-h)x_1 + a_{12}x_2 + \ldots + a_{1n}x_n,\\
\frac{dx_2}{dt} &= a_{21}x_1 + (a_{22}+h)x_2 + \ldots + a_{2n}x_n,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
\frac{dx_n}{dt} &= a_{n1}x_1 + a_{n2}x_2 + \ldots + (a_{nn}+h)x_n,
\end{aligned} \qquad (6\text{-}12)$$

отличающиеся от уравнений (6-9) лишь тем, что к каждому коэффициенту a_{ii} прибавлено положительное число h.

6-2. Схема общей программы расчета областей устойчивости и линий равных степеней устойчивости в плоскости двух параметров

Описываемая ниже программа расчета позволяет свести к минимуму подготовительную работу, которая должна быть проделана перед постановкой расчета на машину Работа исследователя сводится лишь к составлению линеаризованной системы дифференциальных уравнений первого порядка в нормальной форме

$$\begin{aligned}
\frac{dx_1}{dt} &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n,\\
\frac{dx_2}{dt} &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n;\\
&\ldots\ldots\ldots\ldots\ldots\\
\frac{dx_n}{dt} &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n
\end{aligned} \qquad (6\text{-}13)$$

и определению численных значений коэффициентов этих уравнений или зависимости (которая может быть нелинейной) коэффициентов от двух параметров δ и μ, в плоскости которых строится граница области устойчивости Всю остальную работу по определению коэффициентов характеристического уравнения, границы области устойчивости и линий равных степеней устойчивости машина выполняет автоматически

При работе на машине с фиксированной запятой в выражения для коэффициентов уравнений (6-13) следует ввести соответствующие масштабные множители с тем, чтобы в интересующем диапазоне изменений параметров приведенные величины δ и μ не выходили из области значений

$$-1 \leqslant \delta \leqslant +1 \quad -1 \leqslant \mu \leqslant +1 \qquad (6\text{-}14)$$

Область (6-14) плоскости параметров δ, μ разбивается на равные квадраты со сторонами $\Delta\delta=\Delta\mu=\Delta$, которые обычно выбираются равными 0,05 или 0,1 (рис. 6-3)

Расчет границы области производится следующим образом В определенном порядке обходятся узловые точки изображенной на рис. 6-3 сетки и для каждой данной точки при соответствующих ей значениях параметров δ и μ проверяется устойчивость системы. Процесс вычисления границы области состоит из двух этапов, автоматически выполняемых машиной:

а) поиск первой точки (вершины квадрата), расположенной внутри области, б) обход границы и определение координат δ, μ вершин квадратов, прилегающих к границе области изнутри. Граница области строится по этим вершинам или, при желании получить большую точность, по точкам, расположенным от них на расстоянии половины стороны квадрата, как это показано на рис. 6-3. Граница области проходит между соседними вершинами квадратов, расположенными в области и вне области.

При поиске первой точки, расположенной в области, выполняется движение снизу вверх по вертикальным линиям сетки (рис. 6-3) и проверяется устойчивость последовательно для каж-

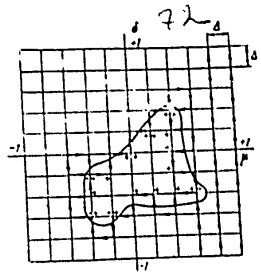


Рис 6-3.

дой узловой точки сетки. Обычно расчет начинается с крайней правой нижней точки ($\delta=-1$, $\mu=-1$). Затем, если эта точка лежит вне области, параметру δ дается приращение Δ и снова проверяется устойчивость и т. д., пока δ не станет равным $+1$. После этого начинается расчет для точек ближайшей слева вертикальной прямой, начиная с нижней точки, и т. д.

После того как найдена первая точка, расположенная в области, начинается обход границы области. Обход производится по вершинам квадратов следующим образом. При каждом шаге обхода один параметр δ или μ получает приращение $\pm\Delta$ и проверяется устойчивость системы. Если на данном шаге изменения δ или μ пересекается граница области и точка попадает внутрь области, то следующие шаги делаются так, чтобы стороны квадрата обходились в направлении против часовой стрелки (учитывая направление шага, при котором пересечена граница области). Если на данном шаге пересечена граница области и точка вышла из нее, то параметры δ и μ изменяются так, чтобы стороны квадрата обходились в направлении по часовой стрелке. Таким образом, при каждом следующем пересечении границы области меняется направление обхода по элементарному квадрату. При каждом пересечении границы (при входе в область и выходе из нее) печатаются значения параметров δ, μ для пограничной точки, расположенной внутри области.

На рис 6-3 показан весь путь поиска и обхода границы области. Крестики указывают точки, координаты которых печатаются Машина автоматически прекращает обход, если все граничные точки определены

Если область устойчивости является незамкнутой в пределах сетки, изображенной на рис 6-3, то при обходе границы области произойдет выход на внешний контур сетки в точке, в которой система устойчива В этом случае

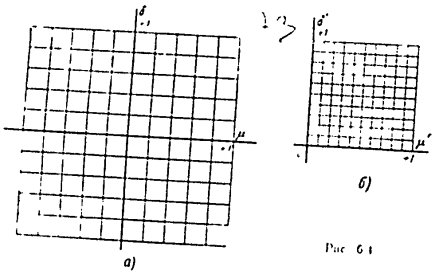


Рис 6-4

обход продолжается по внешнему контуру сетки (а не по вершинам элементарных квадратов) по направлению часовой стрелки до тех пор, пока не будет пересечена граница области При движении по точкам внешнего контура координаты этих точек не печатаются

Возможны более простые приемы обхода границы области. Однако принятый способ обхода является универсальным, так как обеспечивает определение границы при сложных формах области, которые, например, могут иметь место, если коэффициенты характеристического уравнения зависят нелинейно от параметров δ и μ.

Расчет устойчивости для каждой данной точки на этапах поиска и обхода границы выполняется в следующей последовательности: а) определение численных величин коэффициентов уравнений (6-13) при значениях $\delta = \delta_i$, $\mu = \mu_i$, соответствующих рассматриваемой точке, б) определение коэффициентов характеристического уравнения путем раскрытия характеристического определителя (6-11), в) проверка выполнения критерия устойчивости Рауса.

Построение логической схемы программы для осуществления автоматического поиска и обхода границы области упрощается, если от координат δ, μ перейти к таким координатам δ', μ', чтобы интересующей нас области зна

чений δ, μ ($-1 \leq \delta \leq +1$, $-1 \leq \mu \leq +1$) (рис. 6-1,*а*) соответствовал в плоскости δ', μ' квадрат $0 \leq \delta' \leq +1$, $0 \leq \mu' \leq +1$, целиком расположенный в первом квадранте (рис. 6-4,*б*). Координаты δ, μ и δ', μ' связаны между собой следующей зависимостью.

$$\left.\begin{aligned} \delta &= 2\delta' - 1, \\ \mu &= 2\mu' - 1. \end{aligned}\right\} \qquad (6\text{-}15)$$

Шагу Δ в плоскости δ, μ соответствует шаг $\frac{\Delta}{2}$ в плоскости δ', μ'.

Координаты δ' и μ' используются в операторах логической схемы, управляющих поиском и обходом границы области, т. е. выбором следующей расчетной точки области. Затем, когда координаты δ', μ' новой точки выбраны, они преобразуются в соответствующие координаты δ, μ и производятся

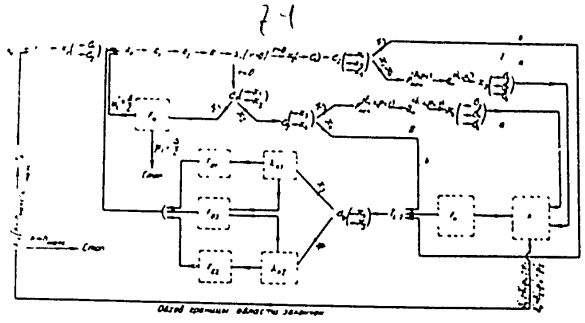


Рис. 6-5.

Расчет коэффициентов характеристического уравнения и проверка критерия устойчивости. Граница области устойчивости печатается в исходных координатах δ, μ.

Упрощенная логическая схема программы изображена на рис. 6-5[1]. Логические условия, соответствующие операциям условного перехода, обозначены через $\lambda[\varphi]$, где φ — условие, определяющее разветвление процесса вычислений. В схеме пунктиром обведены операторы, содержащие логические условия, существенные для управления поиском и обходом границы области, но для упрощения схемы отдельно не показанные.

Ряд разветвлений в процессе вычислений осуществляются операторами безусловного перехода σ_j. Вспомогательные операторы x_j, x_{j+1}, ... ставят в оператор σ_j соответствующую команду безусловного перехода, и управление передается в нужном направлении. Обозначения

$$x_n(\rightarrow \sigma_j) \text{ и } \sigma_j\left(\begin{matrix}\leftarrow x_n \\ \leftarrow x_m\end{matrix}\right)\begin{matrix}x_n \\ x_m\end{matrix}$$

означают, что вспомогательный оператор x_n воздействует на оператор разветвлений σ_j, а оператор σ_j передает управление по тому или иному направлению в зависимости от того, какой из его вспомогательных операторов работал последним.

Логическая схема содержит ряд операторов. Вспомогательные операторы H_0, T, x_j подготовляют схему к началу поиска границы области устойчивости. H_0 ставит в рабочую ячейку величину степени устойчивости $h = 0$ (границе области устойчивости соответствует степень устойчивости $h = 0$). T передает координаты δ, μ первой точки, с которой начинается поиск, в рабочие ячейки и образует соответствующие значения δ', μ'. A_1 осуществляет преобразование координат δ'_j, μ'_j в соответствующие координаты δ_j, μ_j для текущей точки плоскости [по формулам (6-15)]. A_2 вычисляет коэффициенты дифференциальных уравнений (6-13) при текущих значениях δ_j, μ_j.

Оператор A_3 вычисляет коэффициенты характеристического уравнения при текущих значениях δ_j, μ_j, а операторы R и λ выполняют необходимые для критерия Рауса вычисления (в соответствии с табл. 6-1) и проверку совпадения знаков всех элементов левого столбца таблицы.

Если при данных δ_j, μ_j критерий Рауса не выполняется, то оператор σ_j, под-

[1] В разработке программы участвовал Г. С. Гекчан.

готовленный перед началом поиска вспомогательным оператором x_1, передает управление оператору поиска F_x. I_x выбирает следующую точку поиска, делая шаг Δ вверх по ординате, если только при этом точка не выходит за пределы квадрата на рис. 6-16. В противном случае следующей берется нижняя точка на ближайшей слева вертикальной линии сетки. После выбора следующей точки поиска и ее передачи ее координат в ячейки δ'_1, μ'_1 оператор F_x передает управление снова оператору A_x.

После того как найдена первая точка области устойчивости оператор A_x меняет содержание оператора α_1 так, что оператор поиска I_x выключается и включается цепь передачи управления $z_1 \to z_2$. Затем начинается обход границы области.

Если текущая точка при обходе границы лежит внутри области, управление передается цепи *I* логической схемы. Если же текущая точка находится вне области, управление передается цепи *II*. Каждая из указанных цепей схемы состоит из двух цепей *a* и *b*. Непосредственно после пересечения границы области управление передается цепи *Ia* (текущая точка в области) или *IIa* (текущая точка вне области), а затем после того как будет сделан следующий шаг обхода, переходит соответственно к цепи *Ib* или *IIb* и осуществляется этой цепью до нового пересечения границы.

Цепь *a* ветви *I* содержит оператор $P_{xxx}^{\delta'_1 \mu'_1}$, печатающий координаты δ'_1, μ'_1 текущей точки (граничной точки), и оператор $Q_H^{\delta'_1 \mu'_1}$, передающий преобразованные координаты текущей точки в рабочие ячейки оператора K, который контролирует конец обхода области. Кроме того, в этой ветви находится вспомогательный оператор z_3. Оператор z_3 подготовляет цепи управления так, чтобы после нового шага счет пошел либо по цепи *Ib* если новая текущая точка, как и предыдущая, лежит внутри области (выключается цепь *Ia*), либо по цепи *IIa*, если новая точка не находится в области. Оператор z_3, воздействуя на оператор α_3, подготовляет передачу управления оператору выбора шага при обходе против часовой стрелки $F_{\alpha 1}$.

Цепь *IIa* содержит аналогичные операторы с тем различием, что печатаются и передаются в оперативные ячейки оператора K координаты не текущей, а предыдущей $(i-1)$-й точки обхода. Вспомогательный оператор z_4 подготовляет передачу управления оператору выбора шага при обходе по часовой стрелке $F_{\alpha 2}$.

От групп операторов *Ia* (или *IIa*) управление переходит к оператору контроля конца обхода границы области K и оператору направления F_n. Оператор K, контролирующий конец обхода границы области, представляет собой логическую схему, работающую следующим образом. В начале обхода запоминаются координаты δ'_1, μ'_1 и δ'_2, μ'_2 первых двух неодинаковых пограничных точек. Обход границы области заканчивается, когда эти точки снова повторяются в той же последовательности. Если координаты двух последовательных граничных точек не совпадают поочередно с координатами первых двух неодинаковых граничных точек обхода, т. е. если одновременно не соблюдаются равенства $\delta'_k = \delta'_2$, $\mu'_k = \mu'_2$, $\delta'_{k-1} = \delta'_1$, $\mu'_{k-1} = \mu'_1$, то обход не закончен и управление от оператора K передается к оператору направления F_n. Оператор F_n определяет, в каком направлении был сделан шаг обхода, при котором произошло пересечение границы области. Это направление определяется логической схемой, сравнивающей координаты δ'_i, μ'_i текущей точки обхода с координатами δ'_{i-1}, μ'_{i-1} предыдущей точки. В зависимости от направления пересечения границы области („снизу-вверх", „сверху-вниз", „справа-налево", „слева-направо") оператор подготовляет операторы обхода $F_{\alpha 1}$ и $F_{\alpha 2}$, которые обеспечат нужное изменение координат δ', μ'.

После оператора F_n работает оператор T_{i-1}, передающий координаты текущей точки δ'_i, μ'_i в рабочие ячейки координат предыдущей точки. Таким образом, в памяти все время сохраняются координаты предыдущей точки обхода. Эти координаты используются оператором F_n для определения направления пересечения границы области. Если граница не пересекается, то эти операторы выключаются управление от α_3 (или α_4) непосредственно передается оператору T_{i-1} по цепи *Ib* (или *IIb*). Если обход производится внутри области, то оператор α_4 передает управление операторам $\lambda_{\alpha 1}$ и $F_{\alpha 1}$. Если же обход совершается вне области, то включаются операторы $\lambda_{\alpha 2}$, $F_{\alpha 2}$.

Операторы $F_{\alpha 1}$ и $F_{\alpha 2}$ изменяют координаты δ', μ' таким образом, чтобы обход по вершинам квадратов проходил соответственно против или по часовой стрелке. При этом операторы $\lambda_{\alpha 1}$, $\lambda_{\alpha 2}$ проверяют, можно ли сделать нужный шаг, не выходя за пределы квадрата, изображенного на рис. 6-4,*б*. Если этого сделать нельзя, то управление переходит к оператору $F_{\alpha 3}$, который изменяет координаты δ', μ', совершая обход внешнего контура квадрата по часовой стрелке.

Оператор $F_{\alpha 3}$ изменяет координаты δ', μ' до тех пор, пока снова не будет пересечена граница области.

После операторов $F_{\alpha 1}$, $F_{\alpha 2}$ или $F_{\alpha 3}$ управление переходит к A_x и для новой точки весь цикл счета повторяется.

Когда обход границы области закончен, оператор K передает управление оператору T_{Δ}, который увеличивает степень устойчивости на величину Δh. Оператор A_{λ}, вычисляющий коэффициенты исходной системы дифференциальных уравнений, учитывает степень устойчивости h, прибавляя ее к диагональным элементам матрицы коэффициентов системы дифференциальных уравнений.

От оператора T_{Δ} управление переходит к оператору T и производится определение границы с новой степенью устойчивости h.

Оператор λ_x проверяет, не достигло ли текущее значение степени устойчивости h заданной максимальной величины $h_{макс}$. При $h = h_{макс}$ расчет окончен и вычислительная машина останавливается.

Вычисление коэффициентов характеристического уравнения по определителю (6-11) (оператор A_x) выполняется по методу Данилевского [Л. 22], который основан на том, что подобные матрицы имеют одинаковые характеристические полиномы, и состоит в приведении путем последовательности подобных преобразований исходной матрицы коэффициентов системы

$$\begin{pmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & & & & a_{2n} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ a_{n1} & a_{n2} & & & & a_{nn} \end{pmatrix} \quad (6\text{-}16)$$

к подобной матрице, соответствующей „нормальному виду" Фробениуса:

$$\begin{pmatrix} -p_1 & -p_2 & \cdot & -p_{n-1} & -p_n \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \cdot & \cdot & & \cdot & \cdot \\ 0 & 0 & & 1 & 0 \end{pmatrix} \quad (6\text{-}17)$$

Элементы первой строки этой матрицы, как легко непосредственно убедиться, суть коэффициенты характеристического полинома

$$N(\lambda) = -(-1)^n(\lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \ldots + p_n) \quad (6\text{-}18)$$

матрицы (6-17), а следовательно, и подобной ей матрицы (6-16).

При вычислении на машине коэффициентов характеристического полинома правильность результата контролируется проверкой равенства коэффициента p_1 следу матрицы, т. е.

$$-p_1 = a_{11} + a_{22} + \ldots + a_{nn}. \quad (6\text{-}19)$$

При работе на машине с фиксированной запятой вычислению коэффициентов характеристического уравнения и таблицы Рауса ведется с использованием подпрограммы счета с искусственной плавающей запятой.

В вычислительных машинах без внутреннего быстродействующего запоминающего устройства (М-3, «Урал») скорость работы составляет несколько тысяч операций в минуту. При такой скорости в случае расчета устойчивости динамических систем высокого порядка потребуется много времени на раскрытие характеристического определителя во всех точках обхода границы области.

В этом случае целесообразно разделить описанную программу на две части:

а) программу вычисления коэффициентов характеристического уравнения путем раскрытия характеристического определителя (оператор A_3), если известен в алгебраической форме вид зависимости коэффициентов характеристического уравнения от параметров p, Δ (линейная, квадратичная и т. п.) то эта часть программы может быть использована для получения численных значений коэффициентов, определяющих эту зависимость[1].

б) программу вычисления путем обхода и использования критерия Рауса границы области устойчивости (или линий равных степеней устойчивости) в случае, когда в численной форме заданы коэффициенты характеристического уравнения и их зависимость от параметров Δ, p.

6-3. Пример расчета статической устойчивости дальней электропередачи

В качестве простого примера будет рассмотрена схема электропередачи, представленная на рис. 6-6 [Л. 8]. Генератор $Г$ работает на шины приемной системы $ПС$ бесконечной мощности ($U = \text{const}$).

Регулятор $Р$ осуществляет регулирование возбуждения генератора в функции тока I генератора и его первой и второй производных.

Под термином статическая устойчивость электропередачи понимается устойчивость при малых отклонениях от состояния равновесия (устойчивость невозмущенного движения).

Составим дифференциальные уравнения рассматриваемой системы, следуя при этом работе С. А. Лебедева [Л. 18]. Движение ротора синхронного генератора описывается дифференциальным уравнением

$$M\frac{d^2\delta}{dt^2}+P_д\frac{d\delta}{dt}=P^{пот}-P^{эл}, \qquad (6\text{-}20)$$

[1] Например, в случае линейной зависимости коэффициенты характеристического уравнения имеют вид $F_i=A_i+B_i\phi+C_i p$. Раскрывая характеристический определитель для случаев: 1) $p=\phi=0$, 2) $p=0$, $\phi=1$, 3) $p=1$, $\phi=0$, можно найти коэффициенты A_i, B_i, C_i.

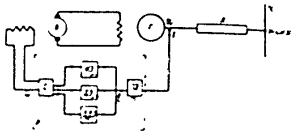


Рис. 6-6.

$Г$ — генератор, $Л$ — линия, $ПС$ — приемная система, $Р$ — регулятор возбуждения, $ПЭ$ — преобразовательный элемент, $ИЧ$ — измерительный элемент, $ДЭ$ — дифференцирующий элемент, $ДД$ — двойной дифференцирующий элемент, $С$ — сумматор.

где M — постоянная инерции ротора генератора,

$P_д$ — успокоительный момент генератора,

$P^{пот}$, $P^{эл}$ — мощность, развиваемая турбиной, и мощность, отдаваемая генератором в сеть,

δ — угол между вектором э. д. с. холостого хода генератора и вектором напряжения U приемной системы.

Для цепи обмотки возбуждения генератора имеем дифференциальное уравнение

$$T_в\frac{dE'_д}{dt}+E_д=E_{дв}, \qquad (6\text{-}21)$$

где $E_д$ — продольная синхронная э. д. с.,

$E'_д$ — продольная составляющая переходной э. д. с. генератора,

$E_{дв}$ — э. д. с., соответствующая напряжению на зажимах возбудителя,

$T_в$ — постоянная времени обмотки возбуждения генератора при разомкнутой цепи статора.

Мощность $P^{эл}$ отдаваемая генератором в систему, зависит от угла δ и э. д. с. $E_д$ (или $E'_д$). Эта зависимость определяется следующими выражениями:

$$P^{эл}_{E_д}=\frac{E_д U}{x_д+x_э}\sin\delta+$$
$$+\frac{U^2(x_д-x'_д)}{2(x'_д+x_э)(x_д+x_э)}\sin 2\delta; \qquad (6\text{-}22)$$

$$P^{эл}_{E'_д}=\frac{E'_д U}{x'_д+x_э}\sin\delta+$$
$$+\frac{U^2(x'_д-x_д)}{2(x'_д+x_э)(x_д+x_э)}\sin 2\delta. \qquad (6\text{-}23)$$

Процессы в электромашинном возбудителе описываются дифференциальным уравнением

$$T_в\frac{dE_{дв}}{dt}+E_{дв}=U_р, \qquad (6\text{-}24)$$

где $T_в$ — постоянная времени цепи возбуждения возбудителя,

$U_р$ — выходное напряжение регулятора.

В регуляторе процессы описываются следующими уравнениями.

Преобразующий элемент $ПЭ$ создает на своем выходе постоянное напряжение, по величине пропорциональное току генератора. Дифференциальное уравнение этого элемента имеет вид:

$$T_р\frac{de}{dt}+e=I, \qquad (6\text{-}25)$$

где I — ток генератора, $T_р$ — постоянная времени преобразующего элемента. Если параметром регулирования является не ток, а напряжение генератора $U_г$ или угол δ, то в правую часть уравнения (6-25) вместо I следует подставить соответственно $U_г$ или δ.

Уравнение чувствительного органа регулятора, реагирующего на отклонение величины тока и его первую и вторую производные, имеет вид:

$$K_0 e+K_1\frac{de}{dt}+K_2\frac{d^2e}{dt^2}=U_р. \qquad (6\text{-}26)$$

В уравнении (6-26) не учтены постоянные времени дифференцирующих контуров. При проектировании регулятора принимаются меры, чтобы эти постоянные времени были малы.

Уравнения (6-20)—(6-26) необходимо дополнить уравнением, связывающим ток статора I с другими переменными.

Из векторной диаграммы легко установить зависимость

$$I=\sqrt{I_д^2+I_q^2}=f(E_д, \delta),$$

причем

$$I_д=\frac{E_д-U\cos\delta}{x_д+x_э}; \quad I_q=\frac{U\sin\delta}{x_q+x_э};$$

$$I=\sqrt{\left(\frac{E_д-U\cos\delta}{x_д+x_э}\right)^2+\left(\frac{U}{x_q+x_э}\sin\delta\right)^2} \qquad (6\text{-}27)$$

В уравнениях (6-26)—(6-27) нужно перейти от самих переменных к их отклонениям от установившихся значений. В результате получается система уравнений

$$M\frac{d^2\Delta\delta}{dt^2}+P_д\frac{d\Delta\delta}{dt}=-\Delta P;$$

$$\Delta P=\left(\frac{\partial P}{\partial\delta}\right)_{E_д=\text{const}}\Delta\delta+\left(\frac{\partial P}{\partial E_д}\right)_{\delta=\text{const}}\Delta E_д;$$

$$\Delta P=\left(\frac{\partial P}{\partial\delta}\right)_{E'_д=\text{const}}\Delta\delta+\left(\frac{\partial P}{\partial E'_д}\right)_{\delta=\text{const}}\Delta E'_д; \qquad (6\text{-}28)$$

$$T_в\frac{d\Delta E'_д}{dt}=\Delta E_{дв}-\Delta E_д;$$

$$T_в\frac{d\Delta E_{дв}}{dt}=\Delta U-\Delta E_{дв};$$

$$T_р\frac{d\Delta e}{dt}=\Delta I-\Delta e;$$

$$K_0\Delta e+K_1\frac{d\Delta e}{dt}+K_2\frac{d^2\Delta e}{dt^2}=\Delta U_р;$$

$$(\Delta I)=\left(\frac{\partial I}{\partial\delta}\right)_{E_д=\text{const}}\Delta\delta+\left(\frac{\partial I}{\partial E_д}\right)_{\delta=\text{const}}\Delta E_д.$$

В соответствии с изложенным в § 6-1 частные производные в (6-28) при малых отклонениях рассматриваются как постоянные коэффициенты. Дифференцируя выражения (6-22), (6-23) и (6-27), можно найти значения частных производных

$$P^{E_д}_\delta=\left(\frac{\partial P}{\partial\delta}\right)_{E_д=\text{const}}=\frac{E_д U}{x_д+x_э}\cos\delta+$$
$$+\frac{U^2(x_д-x_q)}{(x_q+x_э)(x_д+x_э)}\cos 2\delta;$$

$$P^{E_д}_{E_д}=\left(\frac{\partial P}{\partial E_д}\right)_{\delta=\text{const}}=\frac{U}{x_д+x_э}\sin\delta,$$

$$P^{E'_д}_\delta=\left(\frac{\partial P}{\partial\delta}\right)_{E'_д=\text{const}}=\frac{E'_д U}{x'_д+x_э}\cos\delta+$$
$$+\frac{U^2(x'_д-x_q)}{(x'_д+x_э)(x_q+x_э)}\cos 2\delta; \qquad (6\text{-}29)$$

$$P^{E'_д}_{E'_д}=\left(\frac{\partial P}{\partial E'_д}\right)_{\delta=\text{const}}=\frac{U}{x'_д+x_э}\sin\delta;$$

$$I_d^{E_d'} = \left(\frac{\partial I}{\partial \delta}\right)_{E_d'=\text{const}} = \frac{U}{x_d + x_d'}\frac{I_d}{I}\cos\delta + \frac{U}{x_d + x_d'}\frac{I_d}{I}\sin\delta,$$

$$I_{E_d'}^{\delta} = \left(\frac{\partial I}{\partial E_d'}\right)_{\delta=\text{const}} = \frac{I_d}{(x_d + x_d')I},$$

введя обозначения

$$x_1 = \Delta\delta,\ x_2 = \frac{d\Delta\delta}{dt},\ x_3 = \Delta E_d',$$

$$x_4 = \Delta E_{fd},\ x_5 = \Delta e,$$

можно представить систему уравнений 6-28) в виде

$$\begin{aligned}
\frac{dx_1}{dt} &= a_{12}x_2,\\
\frac{dx_2}{dt} &= a_{21}x_1 + a_{22}x_2 + a_{23}x_3,\\
\frac{dx_3}{dt} &= a_{32}x_2 + a_{33}x_3 + a_{34}x_4,\\
\frac{dx_4}{dt} &= a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 + a_{45}x_5,\\
\frac{dx_5}{dt} &= a_{51}x_1 + a_{53}x_3 + a_{54}x_4,
\end{aligned} \qquad (6\text{-}30)$$

где

$$a_{12} = 1,\ a_{21} = -\frac{P_d^{E_d'}}{M},\ a_{23} = -\frac{P_d}{M},$$

$$a_{22} = -\frac{P_{E_d'}^{\delta}}{M},\ a_{23} = -\frac{P_d^{E_d'} - P_{\delta}^{E_d}}{T_d P_{E_d}^{\delta}},$$

$$a_{32} = -\frac{P_{E_d'}^{\delta}}{T_d P_{E_d}^{\delta}},\ a_{34} = \frac{1}{T_d},$$

$$a_{41} = \left[-\frac{P_{E_d'}^{\delta}}{P_{E_d}^{\delta}}\left(P_{\delta}^{E_d'} - P_{\delta}^{E_d}\right) + I_{\delta}^{E_d}\right]\frac{1}{T_j},$$

$$a_{43} = \frac{P_{E_d'}^{\delta}}{P_{E_d}^{\delta}}P_{E_d}^{\delta}\frac{1}{T_j},\ a_{45} = -\frac{1}{T_j},$$

$$a_{51} = -\frac{1}{T_e}\left[a_{51}\left(K_5 - \frac{K_3}{T_p}\right) + K_5 a_{51}a_{13}\right],$$

$$a_{54} = \frac{K_5 a_{54}}{T_e};$$

$$a_{11} = \frac{a_{51}}{T_e}\left(K_5 - \frac{K_3}{T_p}P_{E_d}^{\delta} - P_{E_d}^{\delta} - \frac{K_3}{T_p}\right),$$

$$a_{44} = \frac{1}{T_e}\left(K_4\frac{a_{54}}{T_e} - 1\right),$$

$$a_{55} = \frac{1}{T_e}\left(K_5 - \frac{K_3}{T_p} + \frac{K_3}{T_p^2}\right).$$

Определим границы области устойчивости и линии равных степеней устойчивости в плоскости коэффициентов регулирования по первой и второй производным K_1, K_2 для электропередачи Куйбышевская ГЭС—Москва. Для случая, когда отключена емкостная компенсация и шунтирующие дроссели, параметры системы имеют следующие значения (в относительных единицах)[1]: $U_c = 1$, $U = 2{,}42$, $x_{dT} = 0{,}511$, $x'_{dT} = 0{,}253$, $x_{cT} = 0{,}38$, $x_1 = 2{,}36$, $P_d = 0$; $T_e = 5$ сек; $T_a = 0{,}12$ сек; $M_{cm} = 17$ сек (постоянная инерции, отнесенная к номинальной активной мощности).

На рис. 6-7 построены вычисленные на машине М-3 области устойчивости и линии равных степеней устойчивости для режимов передачи $\delta = 30°$; 60°; 90°, 120° (при $K_4 = 2{,}3$).

[1] Приведены параметры электродинамической модели передачи Куйбышевская ГЭС—Москва, построенной в МЭИ. Приняты базисные величины: $P_E = 11{,}5$ квт; $U_E = 360$ в; $z_E = 11{,}3$ ом (гл. 7).

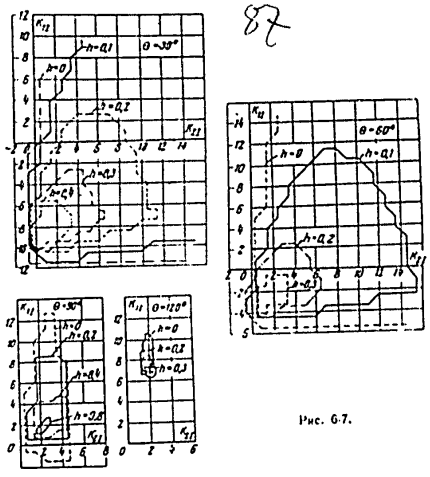


Рис. 6-7.

ГЛАВА СЕДЬМАЯ

РАСЧЕТ И ИССЛЕДОВАНИЕ ПЕРЕХОДНЫХ ПРОЦЕССОВ

7-1. Предварительные замечания

Важное место при проектировании и исследовании новой техники и, в частности, различных электротехнических устройств (электропередач, электроприводов, систем автоматического управления и регулирования и др.) занимают расчеты переходных процессов. Во многих случаях выбор конструктивных параметров определяется требованиями к переходным процессам в проектируемой системе.

Расчет переходных процессов состоит в интегрировании дифференциальных уравнений, описывающих поведение системы при различных характерных начальных условиях и возмущающих воздействиях.

Реальные динамические системы могут содержать элементы с нелинейными характеристиками, с переменными параметрами, а также с распределенными параметрами. В настоящей главе рассматривается расчет переходных процессов в системах, описываемых обыкновенными дифференциальными уравнениями.

В подавляющем большинстве случаев решение дифференциальных уравнений переходных процессов не может быть получено аналитическим методом. Для решения таких задач математика располагает мощным аппаратом методов численного решения дифференциальных уравнений. Среди них широкую известность получили методы Эйлера, Рунге—Кутта, Адамса и др.

Расчеты переходных процессов в сложных нелинейных динамических системах при ручном счете требуют огромного труда. Для таких расчетов во многих случаях весьма успешно могут быть использованы вычислительные машины непрерывного действия. В особенности это относится к динамическим системам, содержащим «типовые» нелинейности — люфт, релейное звено, зону нечувствительности и др.

Использование быстродействующих цифровых вычислительных машин для расчета переходных режимов рационально, если система имеет высокий порядок и содержит элементы, характеризующиеся сложными нелинейными зависимостями.

Особенно эффективно применение этих машин тогда, когда основные расчеты целесообразно сводятся к решению логических задач, связанных с выбором оптимальных конструктивных параметров системы управления. К цифровым машинам приходится прибегать также в тех случаях, когда требуется высокая точность счета нестационарного процесса.

Время, необходимое на подготовительные работы (программирование, отладка программы), существенно сокращается при наличии уже отработанных программ типовых задач.

При интегрировании обыкновенных дифференциальных уравнений на МЦВМ широко используется метод Рунге—Кутта, обеспечивающий достаточную для большинства технических задач точность при большом единообразии вычислений. При повышенных требованиях к точности интегрирования или при большем интервале интегрирования используют метод Адамса, который в вычислительном отношении более сложен, чем метод Рунге—Кутта, так как имеет более сложную схему счета и требует применения какого-нибудь дополнительного метода для определения значений переменных в нескольких первых точках интервала интегрирования.

В гл. 4 была приведена программа интегрирования дифференциального уравнения первого порядка методом Рунге—Кутта. В настоящей главе будут описаны логические схемы программ интегрирования систем обыкновенных дифференциальных уравнений методом Рунге—Кутта с постоянным шагом и с автоматическим выбором шага. В качестве примера будет рассмотрен расчет динамической устойчивости дальней электропередачи.

7-2. Логическая схема программы интегрирования системы обыкновенных дифференциальных уравнений методом Рунге—Кутта с постоянным шагом

Систему дифференциальных уравнений описывающих переходный процесс в динамической системе, можно введением вспомогательных переменных привести к нормальной форме Коши, т. е. разрешить относительно производных первого порядка. Пусть переходный процесс описывается системой дифференциальных уравнений в нормальной форме

$$\left.\begin{aligned}\frac{dy_1}{dt}&=f_1(t,\ y_1,\ y_2,\ \ldots,\ y_m)\\ \frac{dy_2}{dt}&=f_2(t,\ y_1,\ y_2,\ \ldots,\ y_m)\\ &\cdots\\ \frac{dy_m}{dt}&=f_m(t,\ y_1,\ y_2,\ \ldots,\ y_m)\end{aligned}\right\}\quad(7\text{-}1)$$

при начальных условиях $t=t_0,\ y_1=y_{10},\ y_2=y_{20},\ \ldots,\ y_m=y_{m0}$, $f_1, f_2, \ldots, f_m$ — известные, в общем случае нелинейные, функции $t, y_1, \ldots, y_m$.

Для большего единообразия в схеме счета удобно в правых частях уравнений (7-1) время t рассматривать как зависимую переменную

$$y_{m+1}=t \qquad (7\text{-}2)$$

и добавить в системе (7-1) дифференциальное уравнение

$$\frac{dy_{m+1}}{dt}=1 \qquad (7\text{-}3)$$

С учетом (7-2) и (7-3) система (7-1) примет вид

$$\left.\begin{aligned}\frac{dy_1}{dt}&=f_1(y_1,\ y_2,\ \ldots,\ y_n),\\ \frac{dy_2}{dt}&=f_2(y_1,\ y_2,\ \ldots,\ y_n),\\ &\cdots\\ \frac{dy_n}{dt}&=f_n(y_1,\ y_2,\ \ldots,\ y_n),\end{aligned}\right\}\quad(7\text{-}4)$$

где $n=m+1$, $y_n=t$ и $f_n(y_1, y_2, \ldots, y_n)=1$.

Согласно методу Рунге—Кутта, если задана система дифференциальных уравнений в нормальной форме (7-4), то значения функций $y_{i,r+1}$ в конце $r+1$ шага интегрирования находятся по значениям функций $y_{1r}, y_{2r}, \ldots, y_{nr}$ в начале шага интегрирования по формулам

$$y_{i,r+1}=y_{ir}+\Delta y_{ir}\ (i=1,\ \ldots,\ n), \qquad (7\text{-}5)$$

$$\Delta y_{ir}=\frac{1}{6}(K_{i1}+2K_{i2}+2K_{i3}+K_{i4}). \qquad (7\text{-}6)$$

Для системы уравнений (7-4) коэффициенты Рунге-Кутта K_{i1}, K_{i2}, K_{i3}, K_{i4} определяются следующими выражениями

$$\left.\begin{aligned}K_{i1}&=h\ f_i(y_{1r},\ y_{2r},\ \ldots,\ y_{nr}),\\ K_{i2}&=h\ f_i\left(y_{1r}+\frac{K_{11}}{2},\right.\\ &\left.y_{2r}+\frac{K_{21}}{2},\ \ldots,\ y_{nr}+\frac{K_{n1}}{2}\right),\\ K_{i3}&=h\cdot f_i\left(y_{1r}+\frac{K_{12}}{2},\right.\\ &\left.y_{2r}+\frac{K_{22}}{2},\ \ldots,\ y_{nr}+\frac{K_{n2}}{2}\right);\\ K_{i4}&=h\cdot f_i(y_{1r}+K_{13},\\ &y_{2r}+K_{23},\ \ldots,\ y_{nr}+K_{n3}),\end{aligned}\right\}\quad(7\text{-}7)$$

где h — шаг интегрирования.

На каждом шаге интегрирования для определения коэффициентов Рунге—Кутта необходимо 4 раза вычислять значения функций $f_1, f_2, \ldots, f_n$ в правых частях уравнений (7-4) каждый раз при иных значениях их аргументов. Таким образом, естественно процесс счета на каждом шаге интегрирования разбить на четыре такта, соответствующие вычислению первых, вторых, третьих и четвертых коэффициентов Рунге—Кутта — K_{i1}, K_{i2}, K_{i3}, K_{i4} ($i=1, \ldots, n$).

После того как все коэффициенты Рунге—Кутта определены, можно найти приращения и новые значения переменных $y_{1r+1}, y_{2r+1}, \ldots, y_{nr+1}$ по формулам (7-6) и (7-5). Однако программа получается более компактной, если приращение Δy_{ir} разбить на четыре частичных приращения, соответствующих коэффициентам Рунге—Кутта, взятым с определенными весами:

$$\Delta y_{ir}=\frac{1}{6}K_{i1}+\frac{1}{3}K_{i2}+\frac{1}{3}K_{i3}+\frac{1}{6}K_{i4}\ (i=1,\ \ldots,\ n),$$

и в каждом такте счета получаемые частичные приращения прибавлять к значениям переменных в начале шага интегрирования. В табл. 7-1 показан процесс вычислений для одного шага интегрирования, расчлененный на отдельные элементы. Процесс вычислений состоит из четырех единообразных тактов счета и нулевого такта, в котором из значений переменных в конце шага интегрирования y_{ir+1} образуются (путем передачи значений y_{ir+1} в соответствующую группу ячеек памяти) аргументы, по которым определяются f_i в первом такте счета. Величины коэффициентов веса γ и σ циклически изменяются в зависимости от номера s такта счета. Так, величина γ последовательно принимает значения 1, 1, 2, 2, 1, 1, 2, 2 и т. д. Величины коэффициентов веса γ и σ связаны простым соотношением

$$\sigma_s=\frac{1}{3}\gamma_{s+1}\ (s=1,2,3,4). \qquad (7\text{-}8)$$

В последней строчке табл. 7-1 в скобках приведены значения независимой переменной t для всех тактов счета.

На рис. 7-1 приведена логическая схема программы интегрирования системы из n обыкновенных дифференциальных уравнений первого порядка, составленная в соответствии с табл. 7-1. Пользуясь табл. 7-1, можно легко проследить работу логической схемы программы.

В запоминающем устройстве выделяются пять групп по n ячеек памяти — группы ячеек A_0, A_1, A_2, C, D. В группу ячеек A_0 помещаются начальные условия задачи y_{i0}. В группе ячеек A_1 образуются новые значения переменных y_{ir+1} путем прибавления частичных приращений. На каждом шаге интегрирования полученные в группе ячеек A_1

Таблица 7-1

	Исходный такт	Первый такт	Второй такт	Третий такт	Четвертый такт
Коэффициент веса γ_s		$\gamma_1 = 1$	$\gamma_2 = 1$	$\gamma_3 = 2$	$\gamma_4 = 2$
Коэффициент веса δ_s		$\delta_1 = \frac{1}{3}\gamma_2$	$\delta_2 = \frac{1}{3}\gamma_3$	$\delta_3 = \frac{1}{3}\gamma_4$	$\delta_4 = \frac{1}{3}\gamma_1$
Вычисление правых частей f_i $(i=1, \ldots n)$ по значениям аргументов, полученным в предыдущем такте		$f_i(y_{1r} \ldots y_{ir} \ldots y_{nr})$	$f_i\left(y_{1r} + \gamma_1 \frac{K_{11}}{2} \ldots y_{nr} + \gamma_1 \frac{K_{n1}}{2}\right)$	$f_i\left(y_{1r} + \gamma_2 \frac{K_{12}}{2} \ldots y_{nr} + \gamma_2 \frac{K_{n2}}{2}\right)$	$f_i\left(y_{1r} + \gamma_3 \frac{K_{13}}{2} \ldots y_{nr} + \gamma_3 \frac{K_{n3}}{2}\right)$
Вычисление коэффициентов Рунге—Кутта K_{is} $(i=1, \ldots n, S=1,2,3,4)$		$\frac{K_{i1}}{2} = \frac{h}{2} f_i$	$\frac{K_{i2}}{2} = \frac{h}{2} f_i$	$\frac{K_{i3}}{2} = \frac{h}{2} f_i$	$\frac{K_{i4}}{2} = \frac{h}{2} f_i$
Вычисление аргументов, по которым в следующем такте определяются f_i	y_{ir}	$y_{ir} + \gamma_1 \frac{K_{i1}}{2}$	$y_{ir} + \gamma_2 \frac{K_{i2}}{2}$	$y_{ir} + \gamma_3 \frac{K_{i3}}{2}$	—
Вычисление частичных приращений		$\delta_1 \frac{K_{i1}}{2}$	$\delta_2 \frac{K_{i2}}{2}$	$\delta_3 \frac{K_{i3}}{2}$	$\delta_4 \frac{K_{i4}}{2}$
Суммирование частичных приращений и значения переменных в начале шага интегрирования y_{ir}		$y_{ir} + \delta_1 \frac{K_{i1}}{2}$ $\left(t_r + \frac{1}{6}h\right)$	$y_{ir} + \delta_1 \frac{K_{i1}}{2} + \delta_2 \frac{K_{i2}}{2}$ $\left(t_r + \frac{3}{6}h\right)$	$y_{ir} + \delta_1 \frac{K_{i1}}{2} + \delta_2 \frac{K_{i2}}{2} + \delta_3 \frac{K_{i3}}{2}$ $\left(t_r + \frac{5}{6}h\right)$	$y_{ir} + \delta_1 \frac{K_{i1}}{2} + \delta_2 \frac{K_{i2}}{2} + \delta_3 \frac{K_{i3}}{2} + \delta_4 \frac{K_{i4}}{2} = y_{ir+1}$ $(t_r + h)$

...значения переменных, переданные в группу ячеек A, и находящиеся сохраненные в ячейках ..., используются при вычислении аргументов для счета правых частей уравнений. Новые аргументы $y_{ir} + \gamma_s \frac{K_{is}}{2}$ для вычисления правых частей образуются в группе ячеек C. В группе ячеек D в каждом такте сначала помещаются значения правых частей f_i, а затем коэффициенты Рунге—Кутта $\frac{K_{is}}{2}$.

Передача значений переменных из одной группы ячеек в другую, а также образование новых аргументов, коэффициентов Рунге—Кутта, прибавление частичных приращений происходит в n тактов. Управление этими циклическими операциями производится при помощи счетчика числа тактов (ячейка памяти p).

Предварительно перед началом каждого такого цикла в ячейку p ставится нуль, а после каждого такта (например, после каждой передачи y_{ir} из ячейки A_{1i} в ячейку A_{2i}) в ячейку p добавляется единица. Число, образующееся в ячейке p, используется для преобразования команд данного цикла с тем, чтобы на следующем такте операции проводились над величинами с индексом $i+1$. Контроль за окончанием цикла операций выполняется при помощи сравнения величин n и p (операция условного перехода).

Оператор образования циклически изменяющихся коэффициентов веса γ может быть реализован, например, посредством следующей группы команд[1]:

$m+1 \;\; - \;\; .1^* \;\; \beta \;\; \beta$

$m+2 \;\; + \;\; \gamma \;\; \beta \;\; \gamma$

$m+3 \;\; < \;\; \gamma \;\; .3^* \;\; m+5$

$m+4 \;\; - \;\; \gamma \;\; .2^* \;\; \gamma$

$m+5.$

В начале программы в ячейки γ и β передается единица. Читатель легко может проследить, что оператор циклического изменения коэффициента веса последовательно образует в ячейке γ значения 1, 1, 2, 2, 1, 1, 2, 2 и т. д.

[1] Такой прием используется в программе, разработанной в ИТМ и ВТ АН СССР.

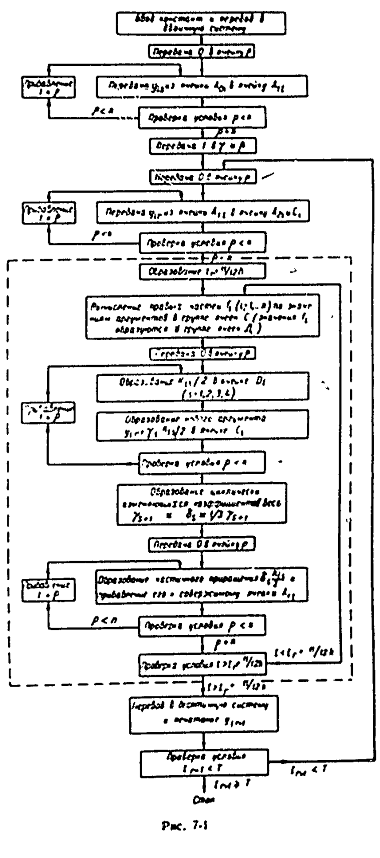


Рис. 7-1

В начале каждого шага интегрирования образуется величина $t_r + \frac{11}{12} h$, которая используется для контроля окончания вычислений на данном шаге, т. е. окончания четырех тактов счета, приведенных на табл. 7-1. Для этого величина $t_r + \frac{11}{12} h$ сравнивается со значениями независимой переменной в конце каждого такта (см. последнюю строчку в табл. 7-1). После четвертого такта $t = t_r + h$, и управление переходит к оператору печати результатов расчета.

Затем проверяется условие $t_{r+1} < T$, где T — заданный интервал. Если $t_{r+1} < T$, то выполняется следующий шаг интегрирования. Для этого все подготовлено, так как в ячейках γ и β стоят единицы, а новые значения переменных $y_{i,r}$ получены в ячейках памяти, в которых в начале предыдущего шага находились величины $y_{i,r}$. Новые значения t_r передаются в ячейки аргументов текущего счета правых частей, и снова повторяется цикл вычислений для одного шага интегрирования.

Если $t_{r+1} > T$, то вычисления закончены и машина останавливается.

Программа, соответствующая схеме на рис. 7-1, является стандартной программой, пригодной для интегрирования системы дифференциальных уравнений (7-1) при любом порядке n и произвольном виде функций f_i. Для каждой конкретной задачи необходимо заново составить лишь кусок программы для вычисления правых частей уравнений.

7-3. Логическая схема программы интегрирования системы обыкновенных дифференциальных уравнений методом Рунге — Кутта с автоматическим выбором шага

При выборе величины шага интегрирования дифференциальных уравнений необходимо, с одной стороны, учитывать требования к точности искомого решения, а с другой стороны, стремиться уменьшить время, затрачиваемое для решения задачи на машине.

Уменьшая в определенных пределах шаг интегрирования, мы увеличиваем точность решения. Однако возрастает общее число шагов, которое должно быть сделано в заданном интервале интегрирования, и пропорционально возрастает машинное время. Для выполнения интегрирования в минимальное время с заданной точностью следует на каждом участке процесса производить вычисления с максимально возможным по условиям получения заданной точности шагом. При этом на участках резкого изменения зависимых переменных шаг должен выбираться небольшим, а на участках, где процесс протекает плавно, шаг можно значительно увеличивать без снижения точности вычислений[1]. При интегрировании дифференциальных уравнений на автоматических цифровых вычислительных машинах широко используются программы, обеспечивающие автоматический выбор величины шага интегрирования, исходя из заданной оценки точности вычислений.

Автоматический выбор шага может осуществляться следующим образом. На каждом шаге интегрирования значения переменных $y^h_{i,r+1}$ вычисляются шагом h (целый шаг), сравниваются со значениями переменных $y^{h/2}_{i,r+1}$, полученными после двух шагов интегрирования половинным шагом $h/2$. Если

$$|y^h_{i,r+1} - y^{h/2}_{i,r+1}| < \varepsilon \quad (i = 1, \ldots, n), \qquad (7\text{-}9)$$

где ε — величина, характеризующая точность вычислений, то за конечный результат принимаются значения $y^{h/2}_{i,r+1}$, полученные половинным шагом, и затем выполняется следующий шаг интегрирования.

Если условие (7-9) не соблюдается, то полученный результат машина отбрасывает и возвращает счет на шаг h назад. Половинный шаг $h/2$ принимается за целый и результат, полученный после одного шага интегрирования шагом $h/2$, сравнивается с результатом, достигнутым после двух шагов интегрирования шагом $h/4$ и т. д. Поэтому необходимо при каждом шаге интегрирования сохранять в запоминающем устройстве значения переменных в начале текущего целого шага и результат первого шага интегрирования половинным шагом. Если на предыдущем результативном шаге интегрирования шаг не делился пополам, то следующий шаг интегрирования увеличивается вдвое. В противном случае величина его не меняется.

Упрощенная логическая схема программы с автоматическим выбором шага изображена на рис. 7-2. В этой схеме не показаны вспомогательные операторы, управляющие при помощи счетчика p n-тактными циклами (передача переменных и др.). Обведенная пунктирной линией часть схемы на рис. 7-2 полностью совпадает с соответствующей частью логической схемы, изображенной более подробно на рис. 7-1.

При автоматическом выборе шага кроме групп ячеек A_0, A_1, A_4, C и D, используемых для тех же целей, что и в программе с постоянным шагом (рис. 7-1), дополнительно в запоминающем устройстве выделяются три группы по n ячеек: B_1, B_2, B_3. В группе

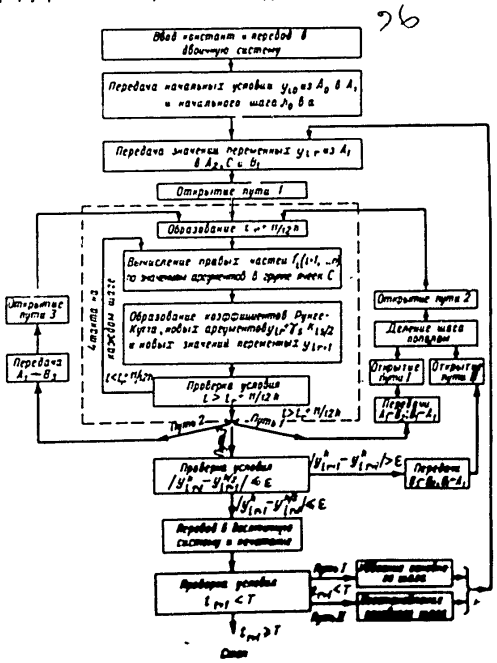


Рис. 7-2.

[1] Более подробно этот вопрос рассмотрен в гл. 9.

ячеек B_i запасаются значения переменных для начала нового целого шага. Эти ячейки используются для восстановления начальных значений переменных при уменьшении шага, когда половинный шаг принимается за целый и необходимо сделать два шага интегрирования шагом $h/1$. Для большего единообразия схемы счета из этой же группы ячеек передаются значения y_{i_0} для начала целого шага при переходе от целого шага к половинному, хотя эти значения можно было бы передать из группы ячеек A_i. В группе ячеек B_i хранятся значения переменных, полученных целым шагом. Ячейки группы B_i используются для хранения полученных после одного половинного шага значений переменных, которые в случае уменьшения шага принимаются за результат, полученный целым шагом.

Цикл вычислений для каждого нового целого шага интегрирования начинается с передачи полученных в группе ячеек A_i на предыдущем результирующем шаге новых значений переменных y_i в группы ячеек A_i, C и B_i. Затем посылкой соответствующей команды 6 условного перехода в конце обведенной пунктиром группы операторов открывается путь 1 передачи управления в программе.

Производятся вычисления, соответствующие одному шагу интегрирования (целым шагом). При этом обведенная пунктиром группа операторов (операторы собственно метода Рунге — Кутта) работает 4 раза согласно табл. 7-1. После окончания этих вычислений происходит передача полученных в группе ячеек A_i значений $y^h_{i,1}$ в группу ячеек B_i для хранения, а в группу ячеек A_i передаются запасенные в группе B_i значения y_{i_0} — начальные значения для интегрирования половинным шагом. Затем происходит открытие пути 1 (подготовка удвоения шага в следующем цикле счета), деление текущего шага пополам и открытие пути 2. Выполняется первый шаг интегрирования половинным шагом. Управление по пути 2 переходит к оператору, передающему полученные в группе ячеек A_i значения $y^{h/2}_{i,1}$ на хранение в группу ячеек B_i. Открывается путь 3 передачи управления и затем выполняется второй шаг интегрирования половинным шагом, после чего в группе ячеек A_i образуются новые значения переменных $y^{h/2}_{i,2}$.

Если условие (7-9) выполняется, то после печатания и проверки, что заданный интервал интегрирования еще не пройден, происходит удвоение основного шага и затем весь цикл счета для нового шага повторяется снова. Если же условие (7-9) не выполняется, значения переменных $y^{h/2}_{i,2}$, полученные последнего (половинного) шага интегрирования и хранившиеся в группе ячеек B_i, передаются в группу ячеек B_i, так как они принимаются за результат, полученный целым шагом. В группе ячеек A_i восстанавливаются начальные условия для рассматриваемого шага интегрирования y_{i_0}. Открывается путь 11 передачи управления (выключается оператор удвоения основного шага), текущий шаг снова делится пополам, открывается путь 2. Затем выполняются два шага интегрирования шагом $h/4$ и т. д., пока не будет удовлетворяться условие (7-9).

Выполнение интегрирования с автоматическим выбором шага усложняет программу счета, но позволяет повысить точность и вести расчет на каждом участке с максимально возможным по условиям точности шагом, что способствует ускорению вычислений при заданной точности.

При численном интегрировании дифференциальных уравнений следует учитывать, что неточность в определении значений переменных $y_{i1}, y_{i2}, y_{i3}, \ldots$ $(i=1,\ldots,n)$ может создать систематически увеличивающуюся погрешность на последующих шагах интегрирования. Поэтому, если даже вычисленное решение сравнительно близко совпадает с точным при небольшом числе шагов интегрирования, то оба эти решения могут сильно разойтись при значительном увеличении интервала интегрирования и соответственно числа шагов.

Если используется метод Рунге — Кутта с постоянным шагом без предварительного исследования влияния шага на точность вычислений для данной задачи, то обычно считается, что

числу шагов интегрирования не должно превышать 200 — 300. При расчетах с автоматическим выбором шага интегрирования общее число шагов может быть увеличено до 600 — 800.

Для выбора величины ε, ограничивающей величину разности переменных, полученных целым и половинным шагом [условие (7-9)], можно пользоваться формулой:

$$\varepsilon = 15\frac{\lambda}{N}, \qquad (7\text{-}10)$$

где λ — допустимая погрешность расчета; N — число шагов.

Приведенные на рис. 7-1 и 7-2 схемы программ отличаются наглядностью, но они не являются оптимальными с точки зрения экономии ячеек памяти и уменьшения числа операций при вычислениях. Так, например, в программе с автоматическим выбором шага можно в запоминающем устройстве сохранять значения f_i правых частей уравнений, полученные по значениям переменных y_i, при определении K_{i1} для целого шага, и использовать эти значения при вычислениях K_{i1} на первом шаге интегрирования половинным шагом.

7-4. Расчет динамической устойчивости дальних электропередач

Использование цифровых быстродействующих вычислительных машин для расчета переходных процессов можно показать на примере расчета динамической устойчивости дальних электропередач.

В случае нелинейных динамических систем, к которым, в частности, относится система электропередачи, рассмотренная в предыдущей главе расчет устойчивости по уравнениям первого приближения дает ответ об устойчивости лишь при малых отклонениях переменных от их значений в установившемся режиме. Между тем в процессе эксплуатации электропередач могут иметь место внезапные резкие изменения режимов работы (короткие замыкания, отключения линий, генераторов и др.). О динамической устойчивости электропередачи, т. е. об отсутствии выпадания из синхронизма при резких (но конечных) внезапных

Рис. 7-3.

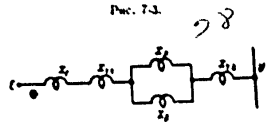


Рис. 7-4.

нарушениях режима ее работы, судят по переходным процессам в системе, протекающим после некоторых характерных (в отношении величины и места приложения) возмущениях.

Рассмотрим, следуя П. С. Жданову [Л. 9], упрощенную качественную картину переходного процесса электропередачи при отключении одной из двух параллельных линий. На рис. 7-3 изображена упрощенная схема электропередачи. На схеме обозначено: $Г$ — генератор, $Т_1$ и $Т_2$ — трансформаторы, $Л_1$ и $Л_2$ — две параллельные (трехфазные) линии электропередачи, $ПС$ — шины приемной системы.

Пусть мощность приемной системы $ПС$ во много раз превышает мощность электропередачи. В этом случае величина и фаза вектора напряжения U приемной системы остаются постоянными при всех режимах работы электропередачи. Если пренебречь активными сопротивлениями и емкостями, то для рассматриваемой электропередачи может быть составлена схема замещения, изображенная на рис. 7-4, в которой генератор, трансформаторы, линии представлены соответствующими реактивными сопротивлениями. Со стороны передающего конца к схеме замещения приложена э. д. с. генератора E. В схеме замещения величины E, U и все реактивности приведены к одной ступени напряжения.

Общая реактивность электропередачи

$$x_r = x_{T1} + x_{Л1} + \frac{x_{Л2}}{2} + x_{T2}. \qquad (7\text{-}11a)$$

Если же одна из параллельных линий ($Л_1$ или $Л_2$) отключается, то реак-

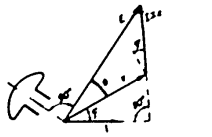


Рис. 7-5.

тивность системы увеличивается, принимая значение

$$x_{\Sigma} = x_1 + x_{T1} + x_2 + x_{T2} \qquad (7\text{-}11б)$$

Из векторной диаграммы электропередачи (рис. 7-5) легко получить выражение для активной мощности P, передаваемой по линии передачи в приемную систему

$$P = UI\cos\varphi.$$

Учитывая, что

$$I_{x_{\Sigma}} = \frac{E}{\sin\delta} = \frac{U}{\sin(90^\circ + \varphi)},$$

получим

$$P = \frac{EU}{x_{\Sigma}}\sin\delta \qquad (7\text{-}12)$$

В системе относительных единиц выражение (7-12) дает мощность для трехфазной системы.

При постоянстве э. д. с. генератора E и напряжения приемной системы U активная мощность P, передаваемая в приемную систему по линии передачи, зависит лишь от угла δ между вектором э. д. с. генератора и напряжения приемного конца. Угловая характеристика мощности (рис. 7-6) в соответствии с (7-12) имеет вид синусоиды.

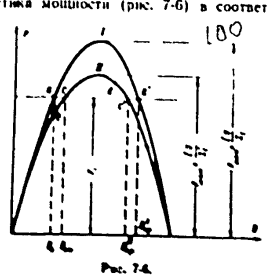


Рис. 7-6.

При возрастании угла δ передаваемая в приемную систему мощность сначала увеличивается, а затем уменьшается. Рассмотрим характеристику I на рис. 7-6. Пусть турбина развивает мощность P_0, причем $P_0 < P_{max}$. Тогда на характеристике I имеются две точки a и a', в которых момент турбины уравновешивается моментом генератора. Отсюда только в точке a равновесие может быть устойчивым. Точка a' на падающей ветви характеристики соответствует неустойчивому равновесию.

Понятие динамической устойчивости можно пояснить на примере отключения одной из параллельных линий в электропередаче, изображенной на рис. 7-3. До отключения линии характеристика системы определяется кривой I на рис. 7-6. Если турбина развивает мощность P_0, то режим работы определяется точкой a на этой кривой.

При отключении одной из линий увеличивается реактивность системы и электропередача переходит на новую характеристику (кривая II) с меньшей амплитудой.

В первый момент после отключения угол δ сохраняет свою величину и рабочая точка смещается в точку b на кривой II.

При этом уменьшается мощность генератора и избыточный момент турбины ускоряет ротор генератора[1]. Угол δ возрастает и рабочая точка перемещается по кривой II вверх от точки b. При этом система по инерции проходит точку c, соответствующую углу $\delta = \delta_{max}$, при котором моменты турбины и генератора уравновешиваются. В точке c избыточный момент изменяет знак, так как мощность генератора становится больше мощности турбины, и ротор начинает тормозиться. Система динамически устойчива, если процесс торможения закончится, не дойдя до точки c', например в точке d. После этого угол δ снова начнет уменьшаться и рабочая точка будет перемещаться от d к c. Колебания ротора постепенно затухнут и установится новое значение угла $\delta = \delta_{уст}$ (рис. 7-7,*а*).

[1] Влияние на рассматриваемый процесс регулятора скорости турбины обычно невелико из-за его инерционности.

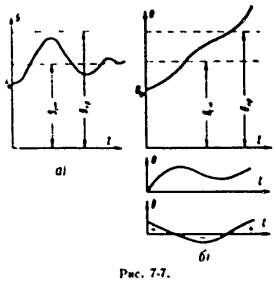


Рис. 7-7.

Если же в процессе торможения угол δ превысит критическое значение $\delta = \delta_{кр}$ (точка c'), то снова появится избыточный момент турбины, угол δ будет непрерывно возрастать и машина выпадет из синхронизма (рис. 7-7,*б*). Этот случай соответствует нарушению динамической устойчивости. По виду переходного процесса при определенном возмущении можно судить о динамической устойчивости системы.

Выражение (7-12) и характеристика на рис. 7-6 справедливы при определенных допущениях. При практических расчетах динамической устойчивости необходимо учитывать непостоянство э. д. с. генератора, активные сопротивления в элементах электропередачи, наличие явных полюсов ротора и другие факторы.

В связи со строительством в нашей стране крупных гидроэлектростанций и дальних линий передач особое значение имеет обеспечение статической и динамической устойчивости синхронных генераторов большой мощности, работающих через дальние линии. Статическая и динамическая устойчивость дальних электропередач обеспечивается быстродействующими системами возбуждения, допускающими значительную форсировку напряжения на кольцах ротора, и регуляторами возбуждения с изодромными органами, реагирующими не только на отклонение напряжения генераторов (в некоторых случаях и на производные напряжения), но и на величину тока генератора и его производные, угол δ и его производные.

Расчет статической устойчивости электропередачи был рассмотрен в предыдущей главе. Расчеты динамической устойчивости электропередачи позволяют правильно сформулировать требования к синхронным генераторам, параметрам линии, систем возбуждения и к аппаратуре, определить необходимые быстродействия защиты и выключателей, необходимые величины скорости нарастания и «потолочного» значения напряжения возбуждения, уточнить необходимый закон регулирования возбуждения.

Широкое внедрение в практику систем возбуждения и регуляторов, обеспечивающих интенсивную форсировку возбуждения, создало условия, при которых стала возможна самосинхронизация (ресинхронизация) выпавших из синхронизма генераторов после отключения короткого замыкания. Поэтому в настоящее время расчет динамической устойчивости должен давать не только ответ на вопрос, произойдет ли нарушение синхронизма, но и определять характер установившегося режима после динамического перехода (восстановление синхронизма или установление асинхронного хода). Расчеты такого рода должны производиться с учетом действия успокоительных обмоток генераторов, ограниченной мощности приемной системы, а также нелинейностей и ограничений в регуляторах возбуждения и скорости и насыщений в системе возбуждения и самом генераторе. Решение такой задачи может быть получено при использовании быстродействующих вычислительных машин.

Для наглядности ниже в качестве примера рассмотрена значительно более простая задача расчета динамической устойчивости, когда об устойчивости судят по поведению системы в первом цикле колебаний после короткого замыкания. Рассмотрим динамическую устойчивость при работе синхронного генератора через дальнюю линию передачи на шины бесконечной мощности.

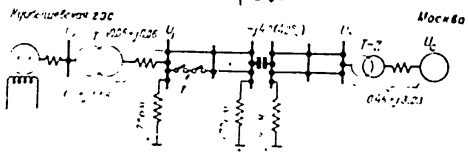


Рис. 7.8

На рис. 7.8 в качестве примера представлена принципиальная схема электропередачи Куйбышевская ГЭС — Москва, выполненная с использованием комплексной индуктивности (4.2), с шунтирующими реакторами [Л. 7]. Линия передачи состоит из двух параллельных цепей, состоящих каждая из четырех участков. С точки зрения динамической устойчивости наиболее опасным является трехфазное короткое замыкание в начале линии с последующим отключением этого участка (рис. 7.8). На рис. 7.9 представлены схемы замещения при включении всех участков электропередачи (*а*) и при отключении первого участка в одной из цепей (*б*). Там же приведены параметры, соответствующие электродинамической модели электропередачи Куйбышевская ГЭС — Москва, построенной в МЭИ [Л. 7].

Задача ставится следующим об-

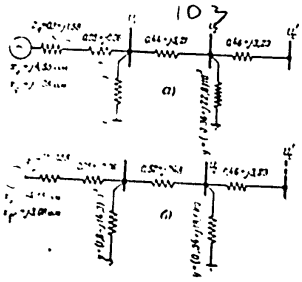


Рис. 7-9

разом. Электропередача работает в установившемся режиме. В момент времени, принимаемый за $t=0$, произошло короткое замыкание на первом участке линии. Через время $t_{к.з}$ короткое замыкание отключается с одновременным отключением первого участка. Спустя время t_a после начала короткого замыкания на обмотке возбуждения генератора скачком изменяется величина напряжения возбуждения в соответствии с его «потолочным значением» $E_{п.т}$. Форсировка возбуждения после этого не отключается.

Целью рассматриваемого упрощенного расчета является определение влияния основных параметров системы: форсирования возбуждения, времени запаздывания включения форсировки t_a, длительности короткого замыкания $t_{к.з}$, на динамическую устойчивость электропередачи.

Изложенная постановка задачи такова, как в работе [Л. 7], где приведены результаты исследований, выполненных на электродинамической модели электропередачи МЭИ и на механическом интеграторе ЭНИН АН СССР. Рассмотрим особенности решения аналогичной задачи на АЦВМ.

Динамическая устойчивость электропередачи при прочих равных условиях может быть оценена максимальным значением угла $\theta_{кр}$ (или передаваемой мощности в доаварийном режиме), при котором еще не происходит выпадения генератора из синхронизма при заданном возмущении. Можно получить вычислительной машине отыскивать два таких граничных значения угла $\theta_{кр}$, которые, отличаясь друг от друга менее чем, например на 5°, удовлетворяют условию, что при

меньшем граничном значении угла система сохраняет устойчивость, а при большем выпадает из синхронизма. Динамическую устойчивость удобно характеризовать полусуммой значений этих углов.

Вычислительная машина начинает расчет с некоторого значения угла θ_0 в доаварийного режима. Если при этом значении угла электропередача оказывается динамически устойчивой, то машина увеличивает угол для исходного режима на 5° и снова проверяет устойчивость и т. д., пока не произойдет выпадения из синхронизма или при первом значении угла динамическая устойчивость нарушается, машина уменьшает исходный угол на 5° и действует таким образом до тех пор, пока не прекратятся выпадения из синхронизма.

Для определения динамической устойчивости при каждом исходном значении угла θ_0 решаются соответствующие дифференциальные уравнения (например, методом Рунге — Кутта) и вычисляется относительное движение ротора синхронного генератора в переходном процессе после короткого замыкания, а заключение об устойчивости может быть сделано по поведению системы в первом цикле колебаний на основе следующих правил (см. пояснения к рис. 7-7). Система устойчива, если в переходном процессе угол θ вначале увеличивается, а затем начинает уменьшаться, т. е. если для каких-нибудь двух последующих моментов времени t_i и t_{i+1} ($t_i < t_{i+1}$) справедливо неравенство $\theta_i > \theta_{i+1}$.

Генератор выпадает из синхронизма, если ускорение ротора в переходном процессе меняет знак с отрицательного на положительный (рис. 7-7,*б*)[1].

Вычислительная машина может сама проверять выполнение указанных условий и прекращать интегрирование дифференциальных уравнений переходного процесса, как только одно из этих условий окажется выполненным. Затем она вычисляет новое значение исходного угла θ_0 и начинает новый расчет. При этом для экономии времени можно совсем не печатать результаты интегрирования дифференциальных уравнений. Машина напечатает лишь искомое значение угла $\theta_{кр}$.

На рис. 7-10 изображена логическая схема оператора, который осуществляет следующие функции:

1) Проверяет выполнение условий устойчивости и выпадения из синхронизма; 2) прекращает процесс интегрирования дифференциальных уравнений при данном значении исходного угла θ_{0_n}, если одно из этих условий выполняется, 3) вычисляет следующее значение исходного угла $\theta_{0_{n+1}} = \theta_{0_n} + 5°$, выбирая знак в зависимости от устойчивости системы при значении угла θ_{0_n}, с которого начался весь расчет; 4) вычисляет и печатает величину $\theta_{кр}$. В начало общей программы, в которую входит этот оператор, ставятся команды, открывающие пути *I*, *II* и *III* в схеме на рис. 7-10.

Для расчета относительного движения ротора синхронного генератора, работающего через дальнюю линию передачи на шины бесконечной мощности, воспользуемся дифференциальными уравнениями [Л.7]

1) $\frac{d\theta}{dt} = \omega$,

2) $\frac{d\omega}{dt} = \frac{M_т - M_э}{M_а}$;

3) $\frac{d\Psi_q}{dt} = E_q - I_d r_a$,

4) $\frac{dt}{dt} = 1$,

5) $M_э = [E + I_d(x_d - x_{р.г})]I_q$; (7-13)

6) $i_q = \frac{x_{р.г}}{r^2 + x_q x_{р.г}} U \sin\theta + \frac{r}{r^2 + x_q x_{р.г}}(E - U\cos\theta)$,

7) $I_q = \frac{\Psi_q - E}{x_{р.г}}$;

8) $I_d = \frac{x_q}{r^2 + x_q x_{р.г}}(E - U\cos\theta) - \frac{r}{r^2 + x_q x_{р.г}} U\sin\theta$;

9) $E = F(I_f - I_d)$,

где θ — угол между векторами эквивалентного напряжения U приемной системы и э. д. с. холостого хода гене-

[1] Строго говоря, если не учитывается апериодическая составляющая тока статора.

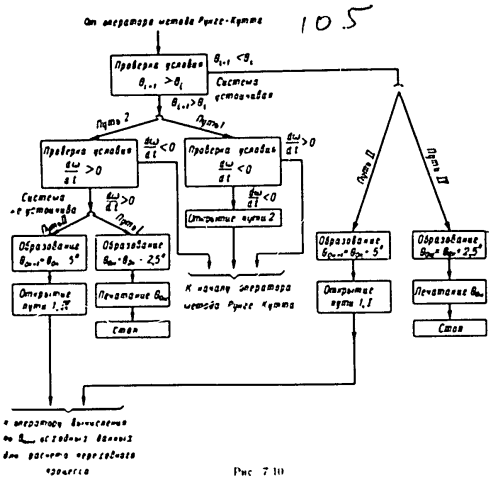


Рис 7-10

ратора ω - скорость относительного движения ротора Ψ_q - потокосцепление обмотки ротора M_t вращающий момент турбины M_e — электромагнитный момент генератора M_0 - постоянная энергий, i_d и i_q — продольная и поперечная составляющие тока статора, i_e — ток возбуждения E_q — напряжение на обмотке возбуждения генератора r - активное сопротивление линии, трансформатора и генератора с учетом добавочных потерь, x_{pr} - сопротивление рассеяния ротора, x_{pc} - сопротивление рассеяния статора генератора с учетом индуктивного сопротивления линии $x_{pc} = x_{pcr} + x_л$, x_e - полное индуктивное сопротивление генератора по поперечной оси с учетом индуктивного сопротивления линии $x_q = x_{qr} + x_л$; E — продольная составляющая э. д. с., соответствующая продольной составляющей магнитного потока в зазоре, r_e — активное сопротивление обмотки возбуждения. Векторная диаграмма для рассматриваемой системы приведена на рис. 7-11.

В уравнениях (7-13) все величины в том числе и время, выражены в относительных единицах (угол θ, время t и M_t выражены в радианах). При выводе уравнений (7-13) сделано предположение, что можно пренебречь трансформаторными э. д. с., наводимыми в переходных процессах в обмотке якоря по осям d и q. Кроме того, не учитывается вращающий момент от токов успокоительной обмотки ротора. Зависимость $E = F(i_e - i_d)$, совпадающая с кривой холостого хода генератора, учитывает насыщение магнитной цепи машины в предположении, что насыщение влияет только на величину сопротивления взаимоиндукции по продольной оси.

Для расчета переходного процесса $\theta = f(t)$ необходимо знать приведенные

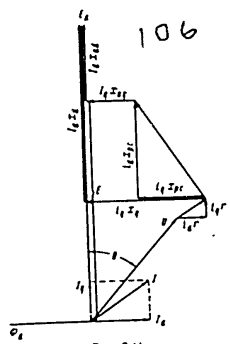


Рис 7-11

к базовым величинам значения параметров блока генератор—трансформатор и параметров передачи для трех режимов работы: а) для исходного нормального режима; б) для режима короткого замыкания; в) для режима после снятия короткого замыкания и отключения первого участка в одной из параллельных цепей.

Для каждого исходного нормального режима ($\theta_0 = \theta_{0s}$, $\omega_0 = 0$), для которого производится проверка динамической устойчивости, необходимо определить начальное условие для $\Psi_q = \Psi_{q0}$ и соответствующие значения M_{t0} и E_{q0}. Эти значения определяются при параметрах $(r_1)_1$, $(x_q)_1$, $(x_{pc})_1$, $(U)_1$ для исходного нормального режима.

Вычисление переходного процесса в рассматриваемой задаче состоит из трех этапов:

1. Интегрирование системы дифференциальных уравнений (7-13) на интервале $0 \leqslant t \leqslant t_1$ при начальных условиях $t = 0$, $\omega = 0$, $\theta = \theta_0$, $\Psi_q = \Psi_{q0}$ и параметрах $(r_1)_1$, $(x_q)_1$, $(x_{pc})_1$, $(U)_1 = 0$ для режима короткого замыкания.

2. В момент времени $t = t_1$ напряжение возбуждения скачком принимает значение $E_q = E_{q\phi}$. Параметры схемы остаются без изменений, т. е. те же, что и в п. 1. Система интегрируется на интервале $t_1 \leqslant t \leqslant t_{1,2}$, причем начальные условия $t = t_1$, $(\theta)_{t-t_1}$, $(\omega)_{t-t_1}$, $(\Psi_q)_{t-t_1}$ определяются из решения для предыдущего участка.

3. В момент времени $t = t_{1,2}$ короткое замыкание отключается. При этом отключается участок параллельной линии. Начиная с $t = t_{1,2}$, система характеризуется новыми значениями параметров:

$$(r_1)_2,\ (x_q)_2,\ (x_{pc})_2,\ (U)_2.$$

Напряжение возбуждения сохраняется равным $E_q = E_{q\phi}$. Начальные условия для расчета движения системы после отключения короткого замыкания определяются из решения на предыдущем участке.

Интегрирование системы уравнений (7-13) производится методом Рунге — Кутта с автоматическим выбором шага и использованием стандартной программы, соответствующей логической схеме на рис. 7-2. Для рассматриваемого расчета $n = 4$.

Функции правых частей дифференциальных уравнений имеют вид:

$$f_1(y_1, y_2, y_3, y_4) = y_3;$$

$$f_2(y_1, y_2, y_3, y_4) = \frac{M_t - M_e}{M_0};$$

$$f_3(y_1, y_2, y_3, y_4) = E_q - i_e r_e;$$

$$f_4(y_1, y_2, y_3, y_4) = 1,$$

где $y_1 = \theta$, $y_2 = \omega$, $y_3 = \Psi_q$, $y_4 = t$, а i_e и M_e определяются на каждом шаге интегрирования по уравнениям (5-9) системы (7-13).

Логическая схема оператора вычисления правых частей уравнения для рассматриваемой задачи показана на рис. 7-12. Схема не требует специальных пояснений. Необходимо лишь остановиться на способе вычисления э. д. с. E и токов i_q, i_d, которые определяются решением системы трех уравнений 7, 8, 9 из (7-13)*.

* Известный ныне способ предложен Т. М. Тер-Микаэляном.

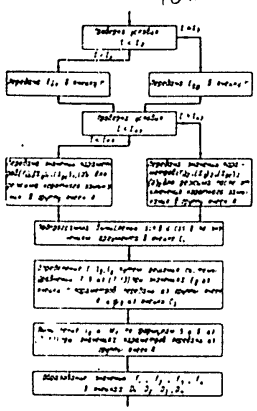


Рис. 7-12

Введем переменную $z = i_\delta - i_d$. После вычитания восьмого уравнения системы (7-14) из седьмого, получается система уравнений

$$\left.\begin{aligned} E &= b - az, \\ E &= F(z), \end{aligned}\right\} \qquad (7\text{-}14)$$

где

$$a = \frac{1}{\dfrac{x_d}{x_{p\,p}} + \dfrac{x^2 + x_\delta x_{p\,c}}{x_{p\,p}}},$$

$$b = \left[\frac{\Psi_0}{x_{p\,p}} + \frac{U}{x^2 + x_\delta x_{p\,c}} \cdot \right.$$

$$\left. \cdot (x_c \cos\delta + r \sin\delta)\right] a.$$

Первое уравнение (7-14) есть уравнение прямой, а второе задано таблицей значений кривой холостого хода генератора (рис. 7-13).

В память машины помещается $N+1$ значение функции $F(z)$ для аргументов

$$z_k = k \cdot \Delta z \quad (k = 0, 1 \ldots N)$$

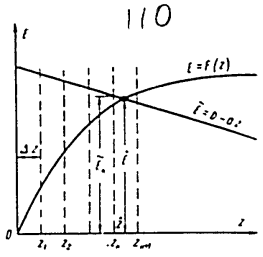


Рис. 7-13.

и в интервале z_k, z_{k+1} функция F заменяется прямой, проходящей через точки кривой, соответствующие абсциссам z_k и z_{k+1}. Легко показать, что в этом случае решение E^*, z^* системы (7-14) может быть получено из выражений

$$E^* = \frac{\tilde{E}_{k+1}(E_k - E_{k+1}) - a\,\Delta z\,E_{k+1}}{(E_k - E_{k+1}) - a \cdot \Delta z},$$

$$z^* = \frac{b - E^*}{a}, \qquad (7\text{-}15)$$

где $\tilde{E}_{k+1}$ и E_k, E_{k+1} — значения ординат прямой и кривой для соответствующих значений z (рис. 7-13).

Чтобы воспользоваться выражениями (7-15), надо найти номер k интервала z_k, z_{k+1}, в котором находится искомое решение. Машина путем последовательных проб находит минимальное k, при котором

$$\tilde{E}_{k+1} - E_{k+1} < 0, \qquad (7\text{-}16)$$

а затем выбирает из памяти E_k, E_{k+1}, вычисляет

$$\tilde{E}_{k+1} = b - (k+1)\,\Delta z \cdot a$$

и определяет E^* и z^* по формулам (7-15). После этого вычисляются:

$$i_\delta = \frac{\Psi_\delta - E^*}{x_{p\,p}} \text{ и } i_d = i_\delta + z.$$

ГЛАВА ВОСЬМАЯ

ПРИМЕНЕНИЕ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ ДЛЯ РАСЧЕТА ЭЛЕКТРИЧЕСКИХ МАШИН

8-1. Общие замечания

Большой интерес представляет использование малогабаритных электронных цифровых вычислительных машин для расчета серий электрических машин, трансформаторов и отдельных специальных машин, если необходимо рассчитать большое число вариантов для выбора оптимального решения.

Вычислительные машины также успешно могут быть применены для получения данных, необходимых для разработки и уточнения инженерной методики расчета некоторых параметров и характеристик электрических машин (параметры успокоительных обмоток синхронных машин и др.).

При проектировании новых серий электрических машин должна быть выполнена большая вычислительная работа по определению оптимальных соотношений конструктивных параметров в разрабатываемых машинах и по расчету всех электрических машин, входящих в серию.

До последнего времени при проектировании серии электрических машин из-за огромной трудоемкости не удавалось проводить полное расчетное определение оптимальных вариантов выполнения машин. Конструктивные параметры назначались после проведения ограниченного числа вариантных расчетов, при этом большую роль играли опыт и интуиция проектантов.

С появлением АЦВМ возникли совершенно новые возможности для выполнения расчетов серии электрических машин. С 1956 г. в НИИЭП ведутся на вычислительной машине М-3 расчеты серий электрических машин. При выполнении поверочных расчетов электрических машин на АЦВМ лишь частично используются ее вычислительные возможности, так как из-за характерного для расчета электрических машин большого числа исходных и конечных данных ввод исходных данных в АЦВМ и печатание результатов расчета занимают во много раз больше времени, чем сам расчет.

Полное использование вычислительных возможностей АЦВМ при расчете серий электрических машин предполагает такую постановку задачи расчета, при которой АЦВМ используется не просто для поверочного расчета, а для расчетного определения оптимальных геометрии и обмоточных данных электрических машин. Поверочный расчет электрической машины на М-3 занимает около 1 *мин*. Поэтому АЦВМ, выполняя по определенной программе большое число расчетов для различных вариантов параметров электрической машины, может автоматически найти и «запомнить» оптимальный вариант геометрии и обмоточных данных машины и затем отпечатать ее расчетный формуляр.

При применении АЦВМ необходимо, чтобы расчетные формуляры электрических машин учитывали особенности производства вычислений на математической машине. Кривые намагничивания должны задаваться в форме полиномов. При использовании АЦВМ с фиксированной запятой большое удобство создает применение относительных единиц, что облегчает выбор масштабных коэффициентов.

8-2. Постановка задачи расчета

В настоящем параграфе рассмотрена с некоторыми упрощениями постановка задачи, которая была принята при расчетах на вычислительной машине М-3 серии асинхронных двигателей от 0,6 до 100 *квт* в 1957 г.*

В качестве критерия оптимальности двигателя принимается минимум сум-

* Постановка задачи расчета серии асинхронных двигателей на АЦВМ и формуляр расчета, приспособленный для вычислительной машины, разработаны под руководством проф., доктора техн. наук Т. Г. Сорокера. Программа, осуществляющая автоматический поиск оптимального расчетного варианта, разработана под руководством автора главы. В работе участвовали Мордвинов Ю. В., Пламодьяло Е. В., Бурмистрова В. Т.

мы C стоимости изготовления двигателя и затрат на электроэнергию за принятый срок его эксплуатации. При таком критерии оптимума расчет в ряде случаев может приводить к варианту двигателя с чрезмерно большим расходом меди. Поэтому для контроля и сопоставления вводится второй критерий оптимума — минимум стоимости изготовления двигателя $C_д$, так как двигатель, соответствующий этому последнему критерию, будет наиболее экономичным по расходу меди.

Для каждого двигателя проектируемой серии вычислительная машина должна найти и отпечатать данные (в том числе величины C и $C_д$) оптимального варианта по критерию C и оптимального варианта по критерию $C_д$.

Проектируемые двигатели должны удовлетворять определенным требованиям в отношении основных показателей, таких как коэффициент мощности $\cos\varphi$, к. п. д., превышение температуры обмотки статора и некоторые другие. Эти требования ограничивают свободу выбора параметров двигателя.

При расчете серии асинхронных двигателей накладывались ограничения на следующие величины: 1) кратность максимального момента k_M, 2) превышение температуры обмотки статора θ, 3) коэффициент мощности $\cos\varphi$.

Величина к. п. д. специально не ограничивалась, так как в двигателе с минимумом суммарных затрат C автоматически достигается высокое значение к. п. д., а в двигателе, выбранном по критерию $C_д$, величина к. п. д. ограничивается допустимым превышением температуры. Ограничение величины коэффициента мощности необходимо лишь для расчетов по критерию $C_д$, так как критерий суммарных затрат обеспечивает одновременно высокое значение $\cos\varphi$. Отдельного ограничения на расход меди не вводится из-за отсутствия обоснований для выбора предельных значений этой величины. Расход меди контролируется путем сопоставления с двигателем, выбранным по критерию $C_д$, являющимся предельным по минимуму расхода меди.

Вычислительная машина должна определить параметры двигателей, при которых суммарные затраты C (а в другом случае — стоимость изготовления $C_д$) имеет минимально возможную величину, при условии, что кратность максимального момента k_M и коэффициент мощности не ниже, а превышение температуры статора θ не выше заданных предельных величин, т. е. выполняются ограничивающие условия

$$\left.\begin{aligned} k_M &\geqslant k_M^{пр}, \\ \theta &\leqslant \theta^{пр}, \\ \cos\varphi &\geqslant (\cos\varphi)^{пр}, \end{aligned}\right\} \quad (8\text{-}1)$$

где $k_M^{пр}$, $\theta^{пр}$, $(\cos\varphi)^{пр}$ заданные предельные значения этих величин.

Для расчетов асинхронных двигателей на АЦВМ использовался специально разработанный для этой цели расчетный формуляр (в относительных единицах), который однозначно определяет все основные размеры (длина активной части, размеры пазов и д.), обмоточные данные и характеристики двигателя как функции четырех величин: 1) внешнего диаметра статора D_a, 2) внутреннего диаметра статора D_i, 3) индукции в зазоре B_δ, 4) числа эффективных проводов в пазу s.

Таким образом, поиск оптимального варианта машины состоит в определении соответствующих значений D_a, D_i, B_δ, s.

При проектировании серии машин производится унификация диаметров и штампов. Поэтому расчет оптимальных диаметров статора — внешнего и внутреннего, оптимальных пазовой геометрии и обмоточных данных, иначе говоря, определение оптимальных значений всех четырех независимых величин D_a, D_i, B_δ, s, производится в каждом габарите машин лишь для наиболее массового в производстве двигателя.

Полученный в результате такого расчета внешний диаметр принимается для всех двигателей данного габарита.

Затем для двигателей вторых длин каждого габарита и скорости вращения рассчитываются при заданном внешнем диаметре оптимальные внутренний диаметр, пазовая геометрия и обмоточные данные. Полученные в результате этих расчетов внутренний диаметр и пазовая геометрия принимаются для двигателей первых длин.

Расчет двигателей первых длин состоит в определении оптимальных обмоточных данных при выбранных в предыдущих расчетах внешнем и внутреннем диаметрах и пазовой геометрии.

8-3. Математическая трактовка задачи. Замечание о линейном и нелинейном программировании

Рассмотрим наиболее общий случай расчета, когда все четыре величины — внешний диаметр статора D_a, внутренний диаметр статора D_i, индукция в зазоре B_δ, число эффективных проводов s — являются искомыми величинами и должны быть определены из условий минимизации критериальных величин C или $C_д$ при учете ограничений (8-1).

Математически эту задачу можно сформулировать следующим образом. Необходимо найти значения переменных D_a, D_i, B_δ и s, при которых функция этих переменных

$$C = F(D_a, D_i, B_\delta, s) \quad (8\text{-}2а)$$

или

$$C_д = F_D(D_a, D_i, B_\delta, s) \quad (8\text{-}2б)$$

принимает минимально возможное значение при условии, что выполняются ограничения, наложенные на значения некоторых функций этих переменных:

$$k_M = f_1(D_a, D_i, B_\delta, s) \geqslant k_M^{пр}; \quad (8\text{-}3)$$

$$\theta = f_2(D_a, D_i, B_\delta, s) \leqslant \theta^{пр}; \quad (8\text{-}4)$$

$$\cos\varphi = f_3(D_a, D_i, B_\delta, s) \geqslant \cos\varphi^{пр}. \quad (8\text{-}5)$$

Учитывая, что по физическому смыслу переменные D_a, D_i, B_δ и s могут принимать лишь положительные значения, необходимо к условиям ограничения (8-3), (8-4) и (8-5) добавить еще четыре следующих ограничения:

$$D_a > 0; \quad (8\text{-}6)$$

$$D_i > 0; \quad (8\text{-}7)$$

$$B_\delta > 0; \quad (8\text{-}8)$$

$$s > 0, \quad (8\text{-}9)$$

В математической литературе задача минимизации (или максимализации) функции

$$z = F(x_1, x_2, \ldots, x_n) \quad (8\text{-}10)$$

при условии, что переменные могут принимать лишь положительные значения

$$x_l \geqslant 0 \ (l = 1, 2, \ldots, n) \quad (8\text{-}11)$$

и ограничены значения некоторых функций этих переменных

$$f_k(x_1, x_2, \ldots, x_n) \leqslant b_k \ (k = 1, \ldots, m), \quad (8\text{-}12)$$

получила название задачи «программирования». В настоящее время этот класс задач привлекает к себе большое внимание, в частности, в связи с расчетами на АЦВМ оптимальных решений в области экономики.

Неравенства (8-11) и (8-12) выделяют в первом квадранте n-мерного пространства область допустимых значений переменных $x_1, x_2, \ldots x_n$.

Если функция F и функции ограничения f_k линейны, то задача называется «линейным программированием». В этом случае подлежат определению значения $x_1, x_2, \ldots, x_n$, при которых линейная функция

$$z = \sum_{l=1}^{n} a_l x_l \quad (8\text{-}13)$$

имеет максимально (или минимально) возможное значение при условии, что

$$x_l \geqslant 0 \ (l = 1, \ldots, n) \quad (8\text{-}11)$$

и удовлетворяются линейные неравенства (ограничения)

$$\sum_{l=1}^{n} c_{lk} x_l \leqslant b_k \ (k = 1, 2, \ldots, m). \quad (8\text{-}14)$$

Проблема линейного программирования особенно часто встречается в экономических исследованиях.

В задаче линейного программирования границы области допустимых значений переменных образуются плоскостями в соответствии с линейными неравенствами (8-11) и (8-14). В этом случае точка, соответствующая макси-

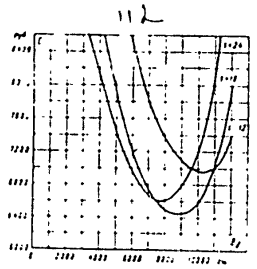


Рис. 8-1

мально (или минимально) возможному значению линейной функции, всегда находится в одной из вершин многогранника, образованного плоскостями ограничений[1].

На знании этого факта основаны методы решения задач линейного программирования.

В случае задачи нелинейного программирования заранее ничего нельзя сказать о расположении точки, доставляющей функции z максимально (или минимально) возможное значение. Она может находиться в любой точке поверхности сферического многогранника, образованного поверхностями ограничений или внутри него.

В настоящее время общие методы решения задачи нелинейного программирования находятся лишь в стадии разработки.

В сформулированной выше задаче определения оптимальных параметров асинхронного двигателя минимизируемые функции и функции ограничения нелинейны. В качестве примера на рис. 8-1 приведены кривые зависимости суммарных затрат C от индукции в зазоре при различных числах эффективных проводов в пазу для двигателя А94-2. Таким образом, наша задача представляет собой задачу нелинейного программирования.

[1] В частном случае функция z (8-13) может принимать наибольшее (или наименьшее) возможное значение на ребре или грани многогранника.

8-4. Способ поиска оптимального варианта двигателя на АЦВМ. Логическая схема программы

Учитывая, что в рассматриваемой задаче имеется сравнительно небольшое число — всего четыре переменных, а расчет двигателя при заданных значениях D_a, D_i, B_δ, s занимает на машине М-3 менее 1 *мин*, определение оптимальных параметров D_a, D_i, B_δ, s двигателя может быть произведено следующим образом.

Первый квадрант четырехмерного пространства переменных D_a, D_i, B_δ, s разбивается параллельными плоскостями, $D_a = \text{const}$, $D_i = \text{const}$, с надлежащим шагом по D_a и D_i.

Каждое полученное сечение представляет собой плоскость переменных B_δ, s при соответствующих значениях $D_a = \text{const}$ и $D_i = \text{const}$. На эти плоскости (в пределах первого квадранта) наносится сетка с шагами ΔB и Δs соответственно по осям B_δ и S (рис. 8-2).

Можно заставить АЦВМ обходить одну за другой узловые точки сетки плоскости B_δ, s для некоторых фиксированных значений D_a и D_i и в каждой точке при соответствующих значениях B_δ, s, D_a и D_i производить расчет двигателя. При этом вычислительная машина должна отбрасывать все варианты, для которых не удовлетворяются заданные ограничения (8-1) для значений k_M, b_z и $\cos\varphi$, а из вариантов для которых условия (8-1) выполняются, выбирать и запоминать лишь

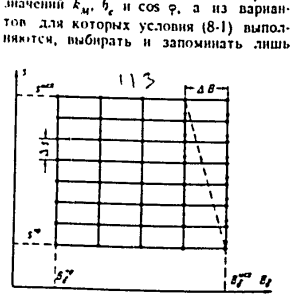


Рис. 8-2.

лучший вариант по критерию C и лучший вариант по критерию C_a. После окончания поиска лучших вариантов в одной плоскости B_δ, s машина переходит к поиску в другой плоскости B_δ, s, соответствующей другим значениям D_a и D_i.

Такой способ поиска позволяет определить с точностью до шагов ΔD_a, ΔD_i, ΔB_δ, Δs, оптимальные значения параметров D_a, D_i, B_δ, s, при которых критерий C (или C_a) имеет с учетом ограничения (8-1) минимально возможную величину.

Таким образом, поиск оптимального решения задачи состоит из поисков лучших вариантов в отдельных плоскостях B_δ, s, соответствующих различным, но фиксированным значениям D_a, D_i.

Выбор в качестве плоскости поиска плоскости B_δ, s удобен тем, что во всех вариантах расчета двигателей, в том числе при расчетах с заданным внешним диаметром статора или с заданными D_a и D_i, сохраняется необходимость в поиске наилучшего решения в плоскости B_δ, s.

На первый взгляд кажется, что число точек, которые АЦВМ должна обойти при поиске, недопустимо велико. На самом же деле для каждого двигателя всегда можно установить разумные пределы изменений B_δ, s, а также D_a и D_i при поиске оптимального варианта.

Поиск оптимального варианта в плоскости B_δ, s (рис. 8-2) начинается с крайней правой верхней точки сетки. Начиная с этой точки, машина автоматически обходит сверху вниз точки сетки, расположенные на одной вертикали. После того как число эффективных проводов в пазу (линейная нагрузка) достигнет предельного минимального значения, поиск переходит на ближайшую слева вертикальную линию сетки.

Для ускорения поиска расчет для данной точки сетки прекращается, как только обнаруживается, что хотя бы одно из условий (8-1) не выполняется, и машина переходит к расчету другой точки поиска.

С этой же целью поиск по данной вертикали прекращается и переходит на следующую вертикаль:

1) если для какой-либо точки данной вертикали было справедливо условие $\cos\varphi \geq (\cos\varphi)^{зад}$, а затем при движении вниз это условие перестало удовлетворяться, или

2) если для какой-либо точки данной вертикали выполнялись все условия ограничения (8-1), а затем при переходе на следующую точку этой же вертикали перестает удовлетворяться хотя бы одно условие ограничения.

Иначе говоря, из физических представлений предположено, что область плоскости B_δ, s, в которой удовлетворяются условия (8-1), односвязна и выпукла. То же, очевидно, относится и к четырехмерной области (D_a, D_i, B_δ, s).

В процессе поиска вычислительная машина производит сравнение двигателей, для которых удовлетворяются ограничения (8-1), по суммарной стоимости C и стоимости изготовления двигателя C_a и запоминает параметры D_a, D_i, B_δ, s для двигателей с минимальной стоимостью C и минимальной стоимостью C_a.

Обозначим C_i и C_{ai} — суммарную стоимость и стоимость двигателя в текущей точке поиска, C_{i-1} и C_{ai-1} — то же для предыдущей точки поиска.

При одновременном выполнении неравенств

$$C_i > C_{i-1},$$

$$C_{ai} \gtrsim C_{ai-1}$$

поиск по данной вертикали сетки прекращается и переходит на следующую вертикаль.

При расчете асинхронного двигателя в машину вводятся следующие исходные величины: 1) мощность; 2) число полюсов; 3) число параллельных ветвей; 4) число пазов статора и ротора; 5) обмоточный коэффициент статора; 6) величина воздушного зазора; 7) шаг обмотки статора; 8) ширина шлица паза статора; 9) таблица диаметров изолированных проводов; 10) кривые намагничивания в форме полиномов для отдельных участков магнитной цепи.

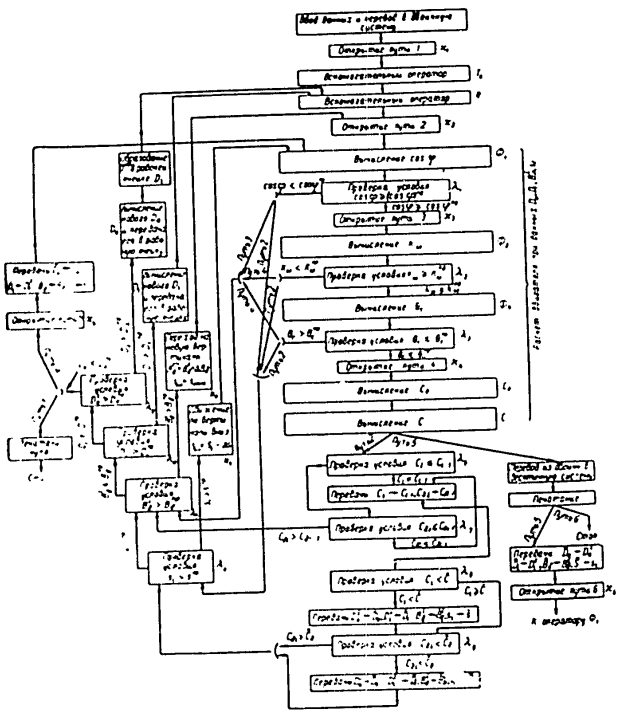


Рис. 8-3.

Кроме того, задаются предельные значения основных показателей машины коэффициента мощности $(\cos\varphi)^{пр}$, кратности максимального вращающего момента $k_M^{пр}$ и превышения температуры статора $\vartheta^{пр}$, а также исходные и предельные значения линейной нагрузки, индукции в зазоре и внешнего диаметра статора.

На рис. 8-3 представлена логическая схема программы поиска оптимальных параметров асинхронного двигателя.

В процессе поиска в ячейки запоминающего устройства $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_\delta$, $\bar{s}$, $\bar{C}$ помещаются соответствующие величины для варианта двигателя с наименьшей стоимостью C, а в ячейки D_a, $\bar{D}_i$, $\bar{B}_\delta$, $\bar{s}$, $\bar{C}_\phi$ — для варианта с наименьшей стоимостью C_ϕ.

Оператор x_1, открывая путь *1*, подготовляет печатание нуля — условного знака, обозначающего отсутствие во всей заданной области поиска точки, удовлетворяющей условиям (8-1)

Вспомогательный оператор T_1 для каждого рассчитываемого двигателя определяет границы области поиска, вычисляя по исходному значению внешнего диаметра $D_a^{исх}$ и номинальным данным двигателя исходное значение внутреннего диаметра статора $D_i^{исх}$ и предельные значения диаметров $D_a^{пр}$ и $D_i^{пр}$ и передает исходные значения $D_a^{исх}$ и $D_i^{исх}$ в рабочие ячейки для D_a и D_i. Кроме того, оператор T_1 засылает единицу в ячейки $\bar{C}$ и $\bar{C}_\phi$ (исходное состояние программы).

Вспомогательный оператор R подготовляет поиск в плоскости B_δ, s при данных величинах D_a и D_i. Этот оператор вычисляет $s^{исх}$, $s^{пр}$ и шаг сетки ΔB и Δs. При этом для числа эффективных проводов в пазу и шага Δs берутся ближайшие бо́льшие четные числа.

Оператор R засылает единицу в ячейки $\bar{C}_{i-1}$ и $\bar{C}_{\phi i-1}$, где должны храниться значения стоимостей для предыдущей точки поиска, и передает координаты $B_\delta^{исх}$ и $s^{исх}$ первой точки поиска в данной плоскости в соответствующие рабочие ячейки для B_δ^i и s_i.

Формулы расчета асинхронного двигателя (для данных D_a, D_i, B_δ, s) разбиты на несколько операторов: Φ_1, Φ_2, Φ_3, C, C_ϕ. Каждый из операторов представляет законченный этап расчета.

Связь между первыми тремя операторами осуществляется посредством логических операций — сравнения. Оператор Φ_1 определяет в конечном счете коэффициент мощности $\cos\varphi$. Оператор Φ_2 производит расчет величины вращающего момента и определяет коэффициент k_M. Оператор Φ_3 осуществляет тепловой расчет и определяет величину превышения температуры статора ϑ.

Операторы C_ϕ и C заключают в себе формулы экономического расчета: они находят стоимости C_ϕ и C.

Операторы λ_1, λ_2, λ_3 проверяют выполнение условий ограничения (8-1). Если $\cos\varphi < \cos\varphi^{пр}$, то при условии, что еще ни разу при движении по данной вертикали сетки не выполнялось неравенство $\cos\varphi \geqslant (\cos\varphi)^{пр}$, расчет для данной точки поиска прекращается и оператор λ_1 передает управление оператору λ_4. Поиск перейдет в новую точку с меньшим s.

При выполнении неравенства $\cos\varphi \geqslant (\cos\varphi)^{пр}$ управление переходит к оператору x_3, который, открывая путь *3*, выключает передачу управления к λ_4; если в дальнейшем при уменьшении s снова перестанет выполняться условие $\cos\varphi \geqslant (\cos\varphi)^{пр}$.

В этом случае управление будет передано оператору λ_4 и поиск перейдет на новую вертикаль.

Если коэффициент мощности находится на требуемом уровне, то работает оператор Φ_2. Оператор λ_2 проверяет выполнение условия $k_M \geqslant k_M^{пр}$. При $k_M < k_M^{пр}$ управление переходит к оператору λ_4. Если условие выполнится, начинает работать оператор Φ_3.

Если превышение температуры обмотки статора меньше заданного, управление передается к оператору x_4, если больше, — то к оператору λ_4.

Оператор λ_4 сравнивает s_i в текущей точке обхода с $s^{пр}$, и если число s_i не достигло своего предельного значения, то производится уменьшение числа проводов в пазу (движение вниз по вертикали сетки). В противном случае управление переходит к оператору λ_5, который проверяет, что еще не достигнуто предельное значение индукции $B_\delta^{пр}$ и в этом случае переводит поиск на новую вертикаль сетки, начиная с самой верхней точки $s = s_{max}$.

Переход к оператору x_5 означает, что текущая точка находится в обла-

сти допустимых значений переменных по условиям (8.1). Оператор λ_9 выполняет передачу управления по пути 1 от оператора λ_{11}, подготовляет печатание данных варианта двигателя, оказавшегося в результате поиска наилучшим. Как только АЦВМ находит на данной вертикали первую точку, в которой условия (8.1) удовлетворяются, открываются пути 4. В результате, если при дальнейшем движении по данной вертикали в сторону меньших s произойдет выход из области допустимых значений B_i, s [по условиям (8.1)] то поиск перейдет на другую вертикаль.

Затем АЦВМ переходит к определению стоимостей C_{i-1} и C. Варианты, в которых все условия ограничения выполняются, сравниваются по стоимости. Операторы λ_7, λ_8 сравнивают обе стоимости C_{i-1} и C_i в данной точке с C_{i-2} и C_{i-1} в предыдущей точке поиска по данной вертикали.

Если операциями сравнения установлено, что обе стоимости в данной точке больше чем в предыдущей, то оператор λ_7 передает управление оператору λ_8 и тем самым машина автоматически переводит расчет на новую вертикаль.

Операторы λ_7, λ_8 осуществляют поиск точек с наименьшей стоимостью. Оператор λ_7 производит сравнение общей стоимости C в данной точке с минимальной стоимостью, которая была достигнута в предыдущих точках поиска. Эта величина находится в ячейке C. При условии, что стоимость в текущей точке C_i меньше величины в ячейке $\bar{C}$, происходит передача координат текущей точки (D_a, D_i, B_i, s) в ячейки $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_i$, $\bar{s}$ и самого значения C в ячейку $\bar{C}$.

Оператор λ_8 сравнивает стоимость двигателя C_{i-1} в данной точке с минимальной величиной стоимости двигателя (хранится в ячейке $\bar{C}_a$), которая была получена в предыдущих точках поиска.

Если $C_{a\,i} < \bar{C}_a$, то происходит передача координат текущей точки и величины $C_{a\,i}$ в ячейки $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_i$, $\bar{s}$, $\bar{C}_a$.

Если $C_i > C$ и $C_{a\,i} > \bar{C}_a$, вытеснения старых координат не происходит.

Затем управление переходит к оператору λ_9 и совершается, если это возможно, движение вниз по вертикали и производится расчет двигателя при новом значении s и т. д.

Если поиск в данной плоскости B_i, s закончен, т. е. достигнуто предельное значение индукции B_i, оператор λ_9 передает управление оператору, меняющему с заданным шагом величину внутреннего диаметра D_i, после чего начинается поиск в новой плоскости B_i, s при измененном значении D_i. При этом оператор λ_{10} проверяет, что величина внутреннего диаметра D_i еще не достигла предельного значения. В противном случае управление переходит к оператору D_a, меняющему с заданным шагом величину внешнего диаметра, и оператору D_i, образующему в соответствующей рабочей ячейке исходное значение диаметра D_i. После этого начинается поиск поочередно в группе плоскостей B_i, s при новом значении D_a и значениях D_i, изменяющихся с заданным шагом в пределах от $D_i^{нач}$ до $D_i^{пр}$.

Весь процесс поиска оптимального варианта двигателя заканчивается, как только оператор λ_{11} обнаруживает, что внешний диаметр достиг предельного значения $D_a^{пр}$. К моменту окончания поиска в ячейках памяти $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_i$, $\bar{s}$ и $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_i$, $\bar{s}$ находятся координаты точек для наилучших вариантов двигателя, соответствующих наименьшим стоимостям C и $\bar{C}_a$.

Оператор λ_{11} по окончании поиска передает управление группе операторов, осуществляющих печатание расчетных данных оптимальных вариантов. Прежде всего происходит открытие пути 5 (подготовка цепей операторов печатания данных). Запасенные в памяти значения $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_i$, $\bar{s}$ для варианта с минимальной стоимостью C передаются в соответствующие рабочие ячейки и управление переходит к оператору Φ_1.

Производится снова расчет двигателя с оптимальными D_a, D_i, B_i, s (по критерию C) и печатание данных его расчетного формуляра.

Затем в рабочие ячейки передаются значения $\bar{D}_a$, $\bar{D}_i$, $\bar{B}_i$, $\bar{s}$ для варианта двигателя, оптимального по критерию C_a, производятся снова расчет двигателя и печатание данных его расчетного формуляра, после чего АЦВМ останавливается.

Описанная программа занимает 1 100 ячеек памяти. При расчете каждого нового типа двигателя в программе меняются 22 константы.

Поверочный расчет двигателя при заданных D_a, D_i, B_i, s на машине М-3 занимает 45—55'. Вывод 40 данных расчетного формуляра (перевод в десятичную систему и печатание) занимает 4 *мин*.

Поиск наилучшего варианта двигателя в плоскости B_i, s при фиксированных значениях D_a и D_i продолжается в среднем около часа. При этом вычислительная машина производит 60 — 80 расчетов вариантов двигателя.

При поиске оптимальных значений D_a, D_i, B_i, s диаметры D_a и D_i принимали каждый 6 различных значений с шагом 2,5%. Таким образом, полный расчет оптимального варианта состоял из поисков наилучшего варианта в 36 плоскостях B_i, s, соответствующих различным значениям D_a и D_i. Такой поиск занимал на машине М-3 около 36 ч.

ГЛАВА ДЕВЯТАЯ

НЕКОТОРЫЕ СВЕДЕНИЯ О ПРИБЛИЖЕННЫХ ВЫЧИСЛЕНИЯХ

9-1. Теория погрешностей

Во всех практических расчетах оперируют с числами, которые получаются в результате измерений различных величин, как, например, расстояния, времени, веса и т. п. Ввиду ограниченных возможностей измерительных средств никогда нельзя получить абсолютно точное численное значение искомых величин. Напротив, результат всегда содержит некоторую погрешность, которая обнаруживается тем, что при повторном измерении получается число, отличное от предыдущего. Лишь в очень редких случаях известно точное значение величин, входящих в формулу. В основном это относится к математическим константам, таким как π, e и т. д. Но и в этих случаях нельзя выразить эти числа точно, так как они содержат бесконечное число десятичных знаков.

Даже при пользовании только точными формулами погрешности исходных величин пройдут через все вычисления и мы неизбежно получим приближенное значение ответа. Кроме того, многие математические задачи можно решить, лишь описав некоторый бесконечный процесс, предел которого и есть искомое решение. Такими бесконечными процессами являются, например, взятие интегралов, вычисление производных и т. п. Так как завершить бесконечный процесс невозможно, то необходимо остановиться на каком-то конечном шаге, что неизбежно внесет погрешность в ответ.

Вместе с тем в инженерных задачах искомые величины определяются для того, чтобы быть использованными на практике. Здесь вновь возникают неизбежные погрешности изготовления или измерения интересующих нас изделий и величин. Следовательно, получение точного значения искомой величины лишено практической ценности и вполне можно удовлетвориться получением приближенного ответа. При этом возникают две задачи: по известным погрешностям исходных величин определить погрешность ответа, и, наоборот, определить, с какой точностью надо взять исходные величины, чтобы результат обладал заданной точностью. Чтобы ответить на эти вопросы, следует определить понятие точности какого-нибудь результата.

Абсолютная и относительная погрешности. Если некоторое число a есть приближенное значение числа A, то модуль разности

$$|A-a|$$

называется истинной абсолютной погрешностью приближенного числа a.

В большинстве случаев истинная абсолютная погрешность неизвестна, так как для ее определения необходимо знать истинное значение A. Но, как правило, при проведении вычислений можно ручаться, что величина допущенной ошибки не превосходит такой-то величины. Эту величину и берут за меру точности измерений и называют абсолютной погрешностью приближенного числа a. Таким образом, **абсолютной погрешностью** приближенного числа a называется возможно меньшее число Δ, удовлетворяющее неравенству

$$A-a\leq\Delta,$$

т. е. абсолютная погрешность больше или равна истинной абсолютной погрешности.

Ясно, что абсолютная погрешность плохо определяет качество измерений. Например, абсолютная погрешность в $\frac{1}{2}$ кГ при взвешивании железнодорожных вагонов и кирпичей свидетельствует о совершенно различной точности измерений. Кроме того, абсолютная погрешность, как правило, есть величина именованная и потому ее значение изменяется с изменением единицы измерения. Поэтому для определения точности приближенных чисел введено понятие относительной погрешности.

Относительной погрешностью приближенного числа a называется отношение абсолютной погрешности Δ к числу a:

$$s=\frac{\Delta}{a}.$$

Относительная погрешность есть величина безразмерная, независящая от единиц измерения. Обычно относительную погрешность выражают либо в процентах ($1\%=0{,}01$), либо в промилле ($1‰=0{,}001$).

Число верных знаков. Практика выработала весьма простой способ определения абсолютной и относительной погрешности путем счета числа верных знаков. Как известно, в десятичной системе счисления каждое число представляется в виде суммы различных степеней 10, умноженных на одну из цифр 0, 1, 2,..., 9. Например, $1\,023=1\cdot10^3+0\cdot10^2+2\cdot10^1+3\cdot10^0$. Если через p обозначен показатель наивысшей степени 10, входящей в число a, то десятичная запись числа a может быть представлена в виде:

$$a=a_p\cdot10^p+a_{p-1}\cdot10^{p-1}+\ldots+$$
$$+a_{p-n+1}\cdot10^{p-n+1}+a_{p-n}\cdot10^{p-n}.$$

Первая слева отличная от нуля цифра a_p называется первой или старшей значащей цифрой. Например, в числах 1 023 и 0,0023 первой значащей цифрой будут соответственно 1 и 2, а p в этих числах равно соответственно 3 и -3. Каждая единица стоящая на определенном месте в десятичной записи числа a, имеет свое определенное значение; единица, стоящая на первом месте, равна 10^p, на втором 10^{p-1}, ... на n-ном 10^{p-n+1}. При округлении числа заменяют в нем все цифры, начиная с некоторой, нулями. Ошибка, которая при этом допускается, не превосходит единицы, стоящей на месте последнего из нетронутых нами знаков.

Число a является приближением величины A с n верными знаками, если абсолютная погрешность числа не превосходит единицы, стоящей на месте последнего n-ного знака, т. е.

$$|A-a|<10^{p-n+1}.$$

Например, число 2,718 выражает число $e=2{,}718281\ldots$ с четырьмя верными знаками, а число 2,7183 выражает e с пятью верными знаками. Как видно на этом примере, число верных знаков можно понимать почти буквально: только последняя из „верных" цифр может отличаться от истинного значения, да и то не более чем на единицу. Лишь в исключительных числах это оказывается не так; например, число 9,999 выражает число 10,00 с четырьмя верными цифрами.

Из определения числа верных знаков следует правило: если приближенное число a выражает число A с n верными знаками, то относительная погрешность a не превосходит

$$\frac{1}{a_p\,10^{p-1}}.$$

Действительно, по определению относительная погрешность равна

$$s=\frac{\Delta}{a}\leq\frac{10^{p-n+1}}{a}<\frac{10^{p-n+1}}{a_p\cdot10^p}=$$
$$=\frac{1}{a_p\cdot10^{n-1}}.$$

Отсюда следует, что, например, при любом значении a_p три верных знака обеспечивают относительную погрешность не больше одного процента. Число, имеющее n верных знаков, пишут обычно с n значащими цифрами, даже если последние из них равны нулю. Например, если число 3,28 имеет пять верных знаков, то его надо записать в виде 3,2800.

Погрешности арифметических действий. Пусть положительные числа $1_1, A_2, \ldots, A_m$ выражаются приближенными числами $a_1, a_2, \ldots, a_m$ с абсолютными погрешностями $\Delta_1, \Delta_2, \ldots, \Delta_m$ и относительными погрешностями $s_1, s_2, \ldots, s_m$. Нужно найти погрешность суммы

$$a=a_1+a_2+\ldots+a_m,$$

являющейся приближенным значением числа

$$A=A_1+A_2+\ldots+A_m.$$

Так как

$$|A-a|=|(A_1+A_2+\ldots+A_m)-$$
$$-(a_1+a_2+\ldots+a_m)|\leq|A_1-a_1|+$$
$$+|A_2-a_2|+\ldots+|A_m-a_m|,$$

то абсолютная погрешность Δ суммы не превосходит суммы абсолютных погрешностей. Далее из соотношения

$$s=\frac{\Delta}{a}\leq\frac{\Delta_1+\Delta_2+\ldots+\Delta_m}{a}=$$
$$=\frac{a_1s_1+a_2s_2+\ldots+a_ms_m}{a}=$$
$$=s_1\frac{a_1}{a_1+a_2+\ldots+a_m}+$$
$$+s_2\frac{a_2}{a_1+a_2+\ldots+a_m}+\ldots+$$
$$+s_m\frac{a_m}{a_1+a_2+\ldots+a_m}$$

следует, что относительная погрешность суммы не превосходит наибольшей относительной погрешности слагаемых. Действительно, если s^* — наибольшее из чисел $s_1, s_2, \ldots, s_m$, то из предыдущего неравенства следует:

$$s\leq s^*\left(\frac{a_1}{a_1+\ldots+a_m}+\ldots+\right.$$
$$\left.+\frac{a_m}{a_1+\ldots+a_m}\right)=s^*.$$

Так как множителем при s_1 стоит величина $\frac{a_1}{a_1+a_2+\ldots+a_m}$, то основное влияние на величину относительной погрешности суммы оказывает относительная погрешность наибольшего слагаемого. Поэтому в сумме будет столько же верных цифр, сколько их в наибольшем слагаемом; складывая несколько чисел, имеющих одинаковое число верных цифр, надо сначала выписать наибольшее из слагаемых, а в остальных слагаемых отбросить все разряды, стоящие справа от последнего разряда наибольшего слагаемого. Например, в сумме

$$7728{,}75+370{,}846+0{,}712813=$$
$$=8\,100{,}308\,813$$

правильными могут быть лишь старшие шесть цифр, и поэтому суммирование нужно было провести следующим образом:

$$\begin{array}{r}7\,728{,}75\\370{,}85\\0{,}71\\\hline8\,100{,}31\end{array}$$

Оценивая погрешность разности двух чисел, нужно различать два случая. Если эти числа сильно разнятся по своей величине, то можно повторить все рассуждения, приведенные выше, и убедиться, что основное влияние на относительную погрешность разности окажет погрешность большего числа. Следовательно, и здесь лишние разряды должны быть отброшены.

Пусть, наоборот, уменьшаемое и вычитаемое, имея равное число верных знаков, близки между собой. Тогда та же абсолютная погрешность придаст им малую разность, отчего ее относительная погрешность резко возрастет.

Например, абсолютная погрешность чисел 150,46 и 150,35 не больше 10^{-4}, а относительная погрешность их разности, равной 0,11, может равняться 10^{-1}, т. е. возрасти в 1 000 раз. Следовательно, при вычислениях надо так преобразовать формулы, чтобы находить разность близких чисел непосредственно, не находя самих этих чисел.

Относительно умножения и деления чисел можно доказать утверждение: относительная погрешность результата ряда последовательных действий умножения и деления не превосходит суммы относительных погрешностей каждого из чисел.

Мы не будем останавливаться на доказательстве этого правила, но заметим, что если число действий невелико (порядка 10), то из сформулированного правила следует, что результат имеет на одну две верные цифры меньше, чем наименьшее число верных цифр в числах участвующих в действии. Однако, если число действий велико, результат может иметь значительно меньшую точность.

9-2. Решение алгебраических и трансцендентных уравнений

Постановка задачи. Пусть требуется решить уравнение

$$f(x) = 0, \qquad (9\cdot1)$$

где $f(x)$ — некоторая трансцендентная или алгебраическая функция. Иными словами, требуется найти точки $x_1, x_2, \ldots x_n, \ldots$, в которых функция $f(x)$ обращается в нуль (так называемые нули функции f). Числа x_i могут быть как действительными, так и комплексными, их может быть бесконечное число, как, например, корней уравнения $\sin x = 0$; они могут лежать близко друг от друга и даже иметь точки сгущения. Иногда ставится задача об отыскании наименьшего положительного корня уравнения (9-1) или об отыскании корней уравнения, лежащих на интервале (a, b) и т. д.

Мы ставим задачу об отыскании приближенных значений корней уравнения (9-1). Если $f(x)$ непрерывная функция, то точку x^* мы будем называть приближенным значением корня уравнения (9-1) с абсолютной погрешностью ε, если $f(x)$ принимает разные знаки в точках $x^* - \frac{\varepsilon}{2}$ и $x^* + \frac{\varepsilon}{2}$. Не существует универсального численного метода, позволяющего решить эту задачу для произвольной функции $f(x)$ и поэтому мы вынуждены рассматривать различные частные случаи.

Первым и наиболее трудным шагом для отыскания вещественных корней уравнения (9-1) является задача об отделении корней. Под этим мы понимаем отыскание двух таких точек a и b между которыми заключен один и только один корень уравнения. Если все корни отделены, то дальнейшие трудности связаны только с объемом вычислительной работы.

Некоторые сведения о расположении действительных нулей функции $f(x)$ может дать исследование ее производной, ибо в силу теоремы Ролля между двумя нулями $f(x)$ находится нечетное число нулей производной $f'(x)$. Иногда удается достигнуть отделения корней, исходя из физических соображений. Например, в задаче об определении собственных частот колебания роторов турбогенераторов, рассмотренной в гл. 5, заранее было известно, что все корни уравнения $D(\lambda) = 0$ положительны и отстоят друг от друга на расстоянии не меньшим, чем фиксированное число h. В этих условиях задача может быть решена так, как это описано в § 5-1 отысканием того интервала, на концах которого функция имеет разные знаки, и последующим делением этого интервала пополам. Этот метод деления интервала пополам является довольно трудоемким даже для быстродействующих машин.

Считая, что нам известны два числа a и b, между которыми заключен только один корень уравнения (9-1), мы дадим более быстрые способы уточнения его значений.

Итерационный метод. Если функция $\varphi(x)$ не обращается в нуль в некоторой окрестности корня α уравнения (9-1) и если

$$\varphi(x) = x - \psi(x) f(x),$$

то уравнение

$$x = \varphi(x) \qquad (9\cdot2)$$

имеет в этой окрестности единственное решение $x = \alpha$.

Предположим, что нам известно грубое значение x_1 корня α и мы хотим его уточнить. Назовем x_1 первым приближением, в качестве второго приближения возьмем

$$x_2 = \varphi(x_1),$$

в качестве третьего

$$x_3 = \varphi(x_2)$$

и т. д., в качестве n-ного

$$x_n = \varphi(x_{n-1})$$

и т. д.

Если последовательность x_n сходится, то ее предел есть корень уравнения (9-2). Действительно, обозначим через α^* предел этой последовательности. Тогда

$$\alpha^* = \lim_{n\to\infty} x_n = \lim_{n\to\infty} \varphi(x_{n-1}) = \varphi(\lim_{n\to\infty} x_{n-1}) = \varphi(\alpha^*).$$

Для сходимости последовательности x_n достаточно, чтобы в окрестности корня α было выполнено условие

$$|\varphi'(x)| < 1,$$

причем сходимость тем быстрее, чем меньше $|\varphi'(x)|$. Вычисления прекращают тогда, когда разность

$$|x_n - x_{n-1}|$$

становится меньше допустимой абсолютной погрешности, иными словами, полагают, что из условия

$$|x_n - x_{n-1}| < \varepsilon$$

следует неравенство

$$|x_n - \alpha| < \varepsilon.$$

В простейшем случае полагают $\psi(x) = 1$ и, следовательно, решают уравнение

$$x - f(x) = x.$$

Итерационные методы особенно удобны для программирования ввиду однообразия выполняемых в них действий.

Метод ложного положения или метод пропорциональных частей. Пусть на интервале (a, b) расположен только один корень α уравнения (9-1). Примем b за первое приближение x_1 и определим итеративный процесс формулами

$$x_2 = \frac{a f(x_1) - x_1 f(a)}{f(x_1) - f(a)};$$

$$x_3 = \frac{a f(x_2) - x_2 f(a)}{f(x_2) - f(a)};$$

. . . . . .

Геометрически x_{i+1} есть точка пересечения оси абсцисс с хордой, соединяющей точки $[a, f(a)]$ и $[x_i, f(x_i)]$ кривой $y = f(x)$. Легко проверить, что в нашем случае функция ψ имеет вид:

$$\psi(x) = \frac{x - a}{f(x) - f(a)}.$$

Метод Ньютона. Если в качестве ψ взять функцию

$$\psi(x) = \frac{1}{f'(x)},$$

то функция φ примет вид:

$$\varphi(x) = x - \frac{f(x)}{f'(x)}$$

и итерационный процесс запишется формулой

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}.$$

Геометрически x_{i+1} есть точка пересечения оси абсцисс с касательной к кривой $y = f(x)$, проведенной в точке $[x_i, f(x_i)]$. Заметим, что метод Ньютона сходится быстрее предыдущего. В § 3-7 по этому методу вычисляется значение квадратного корня из x.

Алгебраические уравнения. Пусть в уравнении (9-1) функция $f(x)$ есть полином степени n с вещественными коэффициентами, т. е. (9-1) имеет вид:

$$P_n(x) \equiv a_n x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n = 0. \qquad (9\cdot3)$$

В силу основной теоремы алгебры это уравнение имеет ровно n корней с учетом их кратности. При этом корень α называется k-кратным корнем уравнения (9-3), если выполнены условия

$$P(\alpha) = P'(\alpha) = \ldots = P^{(k-1)}(\alpha) = 0;$$

$$P^{(k)}(\alpha) \neq 0.$$

Среди этих корней могут быть и комплексные, причем они всегда бывают по-

парно комплексно-сопряженными, иными словами, числа $\alpha+i\beta$ и $\alpha-i\beta$ одновременно являются корнями уравнения (9-3).

Для приближенного отыскания корней уравнения (9-3) применимы описанные выше методы, но так как все они предполагают известным приблизительное расположение корней, то следует и начать с этого вопроса. Наиболее простым было бы изучение графика полинома, но этот способ доступен далеко не всегда и приходится пользоваться другими методами.

Приведем прежде всего следующую теорему. Если обозначить через A максимум модулей всех коэффициентов полинома $P(x)$, то число

$$1+\frac{A}{|a_0|}$$

служит верхней границей для модулей всех его корней как действительных, так и комплексных.

Прежде чем рассматривать вопрос о расположении действительных нулей отметим следующее. Обозначим

$$P_1(x)=x^nP\left(\frac{1}{x}\right),$$

$$P_2(x)=P(-x),$$

$$P_3(x)=x^nP\left(-\frac{1}{x}\right)$$

и пусть N_1, N_2 и N_3 — соответственно верхние границы их положительных корней. Тогда $1/N_1$ есть нижняя граница положительных корней полинома $P(x)$, а числа $-N_2$ и $-1/N_3$ служат соответственно нижней и верхней границами его отрицательных корней. Таким образом, можно впредь говорить лишь о верхней границе положительных корней полинома. Можно сформулировать две теоремы, касающиеся этого вопроса.

Теорема Лагранжа: если $a_0>0$, $a_k(k\geqslant 1)$ — первый из отрицательных коэффициентов и B — наибольшая из абсолютных величин отрицательных коэффициентов, то число

$$1+\sqrt[k]{B/a_0}$$

служит верхней границей положительных корней многочлена.

Теорема Ньютона: если при $x=c$ полином $P(x)$ и все его последовательные производные $P'(x)$, $P''(x),\ldots,P^{(n)}(x)$ положительны, то число c служит верхней границей положительных корней.

Полезно поступать следующим образом. Так как $P^{(n)}(x)=n!a_0>0$, то $P^{(n-1)}(x)$ — возрастающая функция, а следовательно, существует такое c_1, что $P^{(n-1)}(x)>0$ при $x\geqslant c_1$. Следовательно, при $x\geqslant c_1$ производная $P^{(n-2)}(x)$ возрастающая функция, а потому существует такое c_2, что $P^{(n-2)}(x)\geqslant 0$ при $x\geqslant c_2$. Продолжая так дальше, можно найти искомое c.

Для определения числа действительных корней существует метод Штурма, дающий полный ответ на этот вопрос. Но ввиду его большой громоздкости он практически мало применим и не будет здесь изложен. Сформулируем только теорему Декарта, согласно которой число положительных корней многочлена $P(x)$ (с учетом их кратности) равно числу перемен знаков в системе коэффициентов этого многочлена или меньше этого числа на четное число. Например полином

$$x^5+5x^3+3x^2-2x^4-7x^3+9x-10=0$$

имеет либо три, либо только один положительный корень.

Метод Лобачевского. Для решения алгебраических уравнений

$$x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n=0 \tag{9-4}$$

существует метод Лобачевского, который не требует предварительного отделения корней.

Предположим сначала, что все корни уравнения (9-4) вещественны и различны. Расположим их в порядке убывания модулей

$$|z_1|>|z_2|>\ldots>|z_n|.$$

Исходя из уравнения (9-4), составляется уравнение

$$z^n+a_1'z^{n-1}+\ldots+a_{n-1}'z+a_n'=0 \tag{9-4'}$$

по следующему правилу:

коэффициент a_{n-k}' при z^k равен квадрату коэффициента при x^k минус удвоенное произведение коэффициентов при x^{k+1} и x^{k-1} плюс удвоенное произведение коэффициентов при x^{k+2} и x^{k-2} и т. д. вплоть до первого или последнего члена исходного уравнения.

Иными словами

$$a_1'=a_1^2-2a_2,$$

$$a_2'=a_2^2-2a_1a_3+2a_4,$$

$$\ldots\ldots\ldots\ldots\ldots$$

$$a_k'=a_k^2-2a_{k+1}a_{k-1}+2a_{k+2}a_{k-2}.$$

$$\ldots\ldots\ldots\ldots\ldots$$

Исходя из уравнения (9-4'), можно построить новое уравнение так же, как (9-4') получено из (9-4). Продолжая таким образом, на $(k-1)$-том и k-том шагах получаем уравнения

$$x^n+a_1^{(k-1)}x^{n-1}+$$

$$+a_2^{(k-1)}x^{n-2}+\ldots+a_n^{(k-1)}=0, \tag{9-5}$$

$$x^n+a_1^{(k)}x^{n-1}+a_2^{(k)}x^{n-2}+\ldots+a_n^{(k)}=0,$$

для которых будут выполнены приближенные равенства [в уравнениях (9-5) новые переменные обозначены той же буквой x]:

$$a_1^{(k)}\approx[a_1^{(k-1)}]^2,$$

$$a_2^{(k)}\approx[a_2^{(k-1)}]^2,$$

$$\ldots\ldots\ldots$$

$$a_n^{(k)}\approx[a_n^{(k-1)}]^2. \tag{9-6}$$

т. е. в k-том преобразованном уравнении коэффициенты будут равны квадратам соответствующих коэффициентов предыдущего уравнения (с принятой в вычислениях точностью). В этих условиях можно утверждать, что

$$a_1^h=a_1^{(k)};\quad a_2^h=\frac{a_2^{(k)}}{a_1^{(k)}};$$

$$a_3^h=\frac{a_3^{(k)}}{a_2^{(k)}};\quad\ldots\ldots$$

$$a_n^h=\frac{a_n^{(k)}}{a_{n-1}^{(k)}}. \tag{9-7}$$

где $h=2^k$ (если, например, $k=5$, то $h=2^5=32$). Из этих равенств можно определить модули корней $z_1, z_2,\ldots,z_n$; знак корней определяется непосредственной подстановкой в уравнение (9-4).

Предположим теперь, что среди корней уравнения (9-4) есть несколько пар комплексно сопряженных корней

$$z_i=r_i(\cos\varphi_i+\sqrt{-1}\sin\varphi_i);$$

$$z_{i+1}=r_i(\cos\varphi_i-\sqrt{-1}\sin\varphi_i);$$

$$z_j=r_j(\cos\varphi_j+\sqrt{-1}\sin\varphi_j);$$

$$z_{j+1}=r_j(\cos\varphi_j-\sqrt{-1}\sin\varphi_j)$$

$$\ldots\ldots\ldots\ldots$$

и пусть корни перенумерованы так, что

$$|z_1|>|z_2|>\ldots>|z_{i-1}|>r_i>$$

$$>|z_{i+2}|>\ldots>|z_{j-1}|>r_j>$$

$$>|z_{j+2}|>\ldots>|z_n|.$$

После выполнения тех же преобразований вновь получаются уравнения (9-5), для которых равенства (9-6) будут выполнены для коэффициентов при всех степенях x, кроме степеней i, j,... Напротив, коэффициенты $a_i^{(k)}$, $a_j^{(k)}$,... будут вести себя самым неопределенным образом и даже могут менять знак, что и будет служить явным признаком наличия комплексных корней. В этих условиях вместо равенств (9-7) получаются равенства

$$a_1^h=a_1^{(k)};\quad a_{i-1}^h=\frac{a_{i-1}^{(k)}}{a_{i-2}^{(k)}};$$

$$\ldots\ldots\qquad\ldots\ldots$$

$$a_{i-1}^h=\frac{a_{i-1}^{(k)}}{a_{i-2}^{(k)}};\quad r_i^h=\frac{a_{i+1}^{(k)}}{a_{i-1}^{(k)}};$$

$$r_i^h=\frac{a_{i+1}^{(k)}}{a_{i-1}^{(k)}};\quad a_{j+2}^h=\frac{a_{j+2}^{(k)}}{a_{j+1}^{(k)}};$$

$$a_{i+2}^h=\frac{a_{i+2}^{(k)}}{a_{i+1}^{(k)}};\quad\ldots\ldots$$

$$\ldots\ldots\qquad a_n^h=\frac{a_n^{(k)}}{a_{n-1}^{(k)}}.$$

$$\ldots\ldots\ldots$$

При помощи этих равенств определяются сначала вещественные корни $z_1, z_2,\ldots,z_{i-1}, z_{i+2},\ldots,z_j, z_{j+2},\ldots,z_n$ и модули $r_i, r_j,\ldots$ комплексных корней. Если комплексных корней одна

пара, то можно найти их аргумент из равенства

$$-a_1 = z_1 + z_2 + \ldots + z_{i-1} + 2r_i \cos\varphi_i + z_{i+2} + \ldots + z_n.$$

Если же комплексных корней две пары, то мы найдем их аргументы φ_i и φ_j из системы двух уравнений

$$-a_1 = z_1 + \ldots + z_{i-1} + 2r_i \cos\varphi_i + z_{i+2} + \ldots + z_{j-1} + 2r_j \cos\varphi_j + z_{j+2} + \ldots + z_n,$$

$$\frac{a_n}{a_0} = \frac{1}{z_1} \ldots \frac{1}{z_{i-1}} \frac{2}{r_i} \cos\varphi_i \ldots \frac{1}{z_{j-1}} \frac{2}{r_j} \cos\varphi_j \ldots \frac{1}{z_n}.$$

Определение аргументов комплексных корней в случае, если их больше двух пар, а также случаи кратных корней или корней, близких по модулю, здесь не рассматриваются. Все подробности можно найти в книге А. Н. Крылова [Л. 15].

Системы линейных уравнений. Формальное точное решение системы линейных уравнений можно получить по формулам Крамера. Однако, если число уравнений системы достаточно велико, то вычисление входящих в формулы Крамера определителей становится практически неосуществимо даже на современных быстродействующих машинах. Значительно более выгодным является метод Гаусса, по которому система

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

приводится к треугольному виду

$$\begin{aligned} a'_{11}x_1 + a'_{12}x_2 + \ldots + a'_{1n}x_n &= b'_1, \\ a'_{22}x_2 + \ldots + a'_{2n}x_n &= b'_2, \\ \ldots \\ a'_{nn}x_n &= b'_n, \end{aligned}$$

так как это описано в § 1-3 при вычислении определителя. Решение последней треугольной системы уже не представляет труда.

Существуют весьма экономные итеративные способы приближенного решения систем уравнений, однако мы на них не останавливаемся.

9-3. Интерполирование функций

Задача интерполирования. Предположим, что в $n+1$-й точке

$$x_0, x_1, \ldots, x_n$$

заданы значения

$$y_0 = f(x_0), y_1 = f(x_1), \ldots, y_n = f(x_n)$$

некоторой функции $y = f(x)$. Полином $P_n(x)$ степени n, принимающий в точках $x_0, x_1, \ldots, x_n$ значения $y_0, y_1, \ldots, y_n$

$$P_n(x_i) = f(x_i) = y_i, \quad i = 0, 1, \ldots, n,$$

называется интерполяционным полиномом, связанным с функцией $f(x)$ и совокупностью точек x_i (существование и единственность такого полинома доказываются в курсах высшей алгебры). Построение этого полинома и вычисление его значений в точках x, не совпадающих с точками x_i, $i = 0, 1, \ldots, n$, называется интерполированием, а точки x_i — узлами интерполяции. Иногда вычисление значения $P_n(x)$ в точках x, лежащих вне промежутка (x_0, x_n) (в данном случае предполагается, что $x_0 < x_1 < \ldots < x_n$), называют экстраполированием, сохраняя название интерполирование для значений x, лежащих внутри этого промежутка.

Необходимость в построении такого полинома возникает в различных случаях. Пусть, например, значения $y_0, y_1, \ldots, y_n$ получены нами экспериментальным путем и, следовательно, вид функции $f(x)$ нам не известен. Тогда за значения функции $f(x)$ в точках x, отличных от x_i ($i = 0, 1, \ldots, n$), мы принимаем значения интерполяционного полинома $P_n(x)$. Конечно, при этом на функцию $f(x)$ накладываются некоторые ограничения (ее считают достаточно гладкой). Иногда вид функции $f(x)$ известен, но настолько сложен, что вычисление ее значений является весьма трудоемким процессом. В этих случаях вычисляют значения функции $f(x)$ в ряде точек $x_0, x_1, \ldots, x_n$, строят по этим точкам интерполяционный полином и за значения функции $f(x)$ в точках x, отличных от x_i, принимают значения $P_n(x)$, вычислять которые значительно проще.

Случай равноотстоящих узлов. Рассмотрим наиболее часто встречающийся случай, при котором значения некоторой функции $y = f(x)$

$$y_1, y_2, y_3, \ldots, y_m \tag{9-8}$$

заданы в точках

$$x_1, x_2, x_3, \ldots, x_m, \tag{9-9}$$

отстоящих друг от друга на равных расстояниях h, т. е.

$$x_i = x_1 + h(i-1), \quad i = 1, 2, \ldots, m.$$

В этом случае для построения интерполяционных полиномов вводят так называемые разности, определяемые следующим образом.

Разностями первого порядка называют величины

$$\Delta y_1 = y_2 - y_1,$$
$$\Delta y_2 = y_3 - y_2, \quad \ldots \Delta y_{m-1} = y_m - y_{m-1}.$$

В свою очередь, разности первых разностей называются разностями второго порядка, которые, следовательно, равны

$$\Delta^2 y_1 = \Delta y_2 - \Delta y_1 = y_3 - 2y_2 + y_1;$$
$$\Delta^2 y_2 = \Delta y_3 - \Delta y_2 = y_4 - 2y_3 + y_2;$$
$$\ldots \ldots \ldots \ldots \ldots \ldots$$
$$\Delta^2 y_{m-2} = \Delta y_{m-1} - \Delta y_{m-2} = y_m - 2y_{m-1} + y_{m-2}.$$

Аналогично определяются разности третьего порядка и т. д. Вообще нетрудно показать, что разность k-того порядка имеет вид:

$$\Delta^k y_i = \sum_{m=0}^{k} (-1)^m C_k^m y_{i+k-m},$$

где через C_k^m обозначены биномиальные коэффициенты

$$C_k^m = \frac{k(k-1)\ldots(k-m+1)}{1 \cdot 2 \ldots m}.$$

Для удобства эти разности располагают в таблицу разностей, которая наглядно показывает, как ведут себя разности различных порядков. Например, для функции $f(x) = x^3$ таблица разностей дана в табл. 9-1.

Таблица 9-1

Таблица разностей функции $y = x^3$

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$	$\Delta^6 y$
0	0	1	30	150	240	120	0
1	1	31	180	390	360	120	0
2	32	211	570	750	480	120	0
3	243	781	1 320	1 230	600	120	—
4	1 024	2 101	2 550	1 830	720	—	—
5	3 125	4 651	4 380	2 550	—	—	—
6	7 776	9 031	6 930	—	—	—	—
7	16 807	15 961	—	—	—	—	—
8	32 768	—	—	—	—	—	—

Здесь разности пятого порядка постоянны, а шестого — равны нулю (легко показать, что разности n-ного порядка полинома n-ной степени постоянны).

Пусть теперь x — произвольная точка и

$$u = \frac{x - x_1}{h},$$

где x_1 — одна из точек (9-9). Полином степени n, принимающий в $n+1$-й точке $x_1, x_{1+1}, \ldots, x_{1+n}$ значения $y_1, y_{1+1}, \ldots, y_n$, дается интерполяционным формулой Ньютона

$$P_n(x) = y_1 + \frac{u}{1!}\Delta y_1 + \frac{u(u-1)}{2!}\Delta^2 y_1 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_1 + \ldots + \frac{u(u-1)(u-2)\ldots(u-n+1)}{n!}\Delta^n y_1.$$

Погрешность $R_n(x) = |f(x) - P_n(x)|$, допускаемая при замене функции $f(x)$ интерполяционным полиномом $P_n(x)$, выражается формулой

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{n!}(x - x_1) \times (x - x_{1+1}) \ldots (x - x_{1+n}), \tag{9-10}$$

где ξ — точка, лежащая между наибольшей и наименьшей из точек $x, x_1, x_{1+1}, \ldots, x_{1+n}$. Если вид функции $f(x)$ известен и $n+1$-ю производную $f^{(n+1)}(x)$ легко оценить, то из этой формулы можно найти величину погрешности. В противном случае необходимо пользоваться приближенным равенством

$$f^{(n+1)}(\xi)=\frac{\Delta^{n+1}f(x_i)}{h^{n+1}},$$

исходя из которого берут степень n интерполяционного полинома такой, чтобы n ные разности в пределах выбранной точности были постоянными (тогда $n+1$-е разности равны нулю, а вместе с тем и $R_n=0$).

Из вида формулы (9-10) можно сделать заключение о наиболее целесообразном выборе точек x_i (входа в таблицу) при заданном значении x. Действительно, в формулу для R_n входит полином

$$(x-x_i)(x-x_{i+1})\ldots(x-x_{i+n}),$$

корнями которого являются точки x_i, x_{i+1}, x_{i+n} (при $x=x_i$ $f(x_i)=P(x_i)$ и $R(x_i)=0$). Если рассмотреть график этого полинома, то будет видно, что его максимумы, близкие к среднему корню $x_{i+\frac{n}{2}}$ (считая n четным числом), меньше, чем максимумы, близкие к крайним корням x_i и x_{i+n} (рис. 9-1). Поэтому x_i надо выбирать так, чтобы точка x оказалась возможно ближе к середине участка интерполяции x_i, x_{i+1}, x_{i+n}. При указанном выборе x_i и n можно считать, что интерполяционные значения имеют столько же верных знаков, сколько их в табличных значениях $y_1, y_2, \ldots, y_m$. Конечно, если x близко к краям таблицы (к x_1 или x_m), то указанный выбор x_i невозможен и мы должны брать наилучшую из возможных точек. В частности, если $x_1<x<x_2$, то входом в таблицу будет x_1, если же $x_{m-1}<x<x_m$, то входом в таблицу будет x_{m-1}. При фактическом интерполировании вход x_i в таблицу выбирают заново для каждого нового значения x, степень же n полинома обычно берут постоянной.

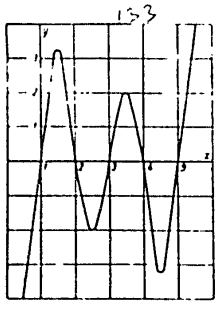


Рис 9-1

Неравноотстоящие узлы. В случае, когда точки (9-9) расположены произвольно, вводятся так называемые разделенные разности, которые определяются следующим образом.

разделенными разностями первого порядка называются величины

$$f(x_1, x_2)=\frac{f(x_1)-f(x_2)}{x_1-x_2},$$

$$f(x_2, x_3)=\frac{f(x_2)-f(x_3)}{x_2-x_3},$$

разделенными разностями второго порядка называются величины

$$f(x_1, x_2, x_3)=\frac{f(x_1, x_2)-f(x_2, x_3)}{x_1-x_3},$$

$$f(x_2, x_3, x_4)=\frac{f(x_2, x_3)-f(x_3, x_4)}{x_2-x_4}$$

и т. д.

При помощи разделенных разностей записывается интерполяционная формула Ньютона для неравных промежутков

$$P_n(x)=f(x_i)+(x-x_i)f(x_{i+1}, x_i)+$$
$$+(x-x_i)(x-x_{i+1})f(x_{i+2}, x_{i+1}, x_i)+\ldots+$$
$$+(x-x_i)(x-x_{i+1})\ldots$$
$$\ldots(x-x_{i+n-1})f(x_{i+n}, x_{i+n-1}, \ldots, x_i).$$

Интерполяционный полином, не требующий вычисления разделенных разностей, дается формулой Лагранжа

$$P_n(x)=f(x_i)\frac{(x-x_{i+1})(x-x_{i+2})\ldots(x-x_{i+n})}{(x_i-x_{i+1})(x_i-x_{i+2})\ldots(x_i-x_{i+n})}+$$
$$+f(x_{i+1})\frac{(x-x_i)(x-x_{i+2})\ldots}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})\ldots}\rightarrow$$
$$\rightarrow\frac{\ldots(x-x_{i+n})}{\ldots(x_{i+1}-x_{i+n})}+\ldots+f(x_{i+n})\times$$
$$\times\frac{(x-x_i)(x-x_{i+1})\ldots(x-x_{i+n-1})}{(x_{i+n}-x_i)(x_{i+n}-x_i)\ldots(x_{i+n}-x_{i+n-1})}.$$

Это есть полиномы степени n, которые в $n+1$ точках $x_i, x_{i+1}, \ldots, x_{i+n}$ принимают заданные значения $f(x_i)$, $f(x_{i+1}), \ldots, f(x_{i+n})$. Следовательно, оба полинома совпадают, но внешне они не схожи, так как члены в них сгруппированы различным образом. Заметим, что в данном случае нумерация точек совершенно произвольна, но обычно их нумеруют в порядке возрастания. Выбор степени n и входа x_i вновь определяется из формулы (9-10). Однако ввиду несимметричности точек $x_1, x_2, \ldots, x_m$ выбор входа x_i в этом случае значительно сложнее.

Обратное интерполирование. Пусть в точках (9-9) заданы значения (9-8) некоторой функции $y=f(x)$. Отыскание того аргумента x, при котором функция $f(x)$ принимает значение y^*, не совпадающее с табличными значениями (9-8), называется задачей обратного интерполирования.

Если точки (9-9) расположены на одинаковом расстоянии, то эта задача может быть решена при помощи формулы Ньютона. Положив в ней $P_n(x)=y^*$, ее можно записать в виде:

$$u=\frac{y^*-y_i}{\Delta y_i}-\frac{u(u-1)}{2!}\frac{\Delta^2 y_i}{\Delta y_i}-\ldots-$$
$$-\frac{u(u-1)\ldots(u-n+1)}{n!}\frac{\Delta^n y_i}{\Delta y_i}.$$

Иными словами, формуле Ньютона придан вид уравнения

$$u=F(u),$$

которое теперь можно решить итерационным способом, взяв в качестве первого приближения величину

$$u^{(1)}=\frac{y^*-y_i}{\Delta y_i}.$$

Определив u, можно найти искомое x на основании равенства

$$u=\frac{x-x_i}{h}.$$

В случае, когда точки (9-9) расположены произвольно, эту же задачу решают при помощи интерполяционной формулы Лагранжа, которую однако записывают в виде, разрешенном относительно x (иными словами, считая x зависимым, а y — независимым переменным):

$$x=x_i\frac{(y-y_{i+1})(y-y_{i+2})\ldots(y-y_{i+n})}{(y_i-y_{i+1})(y_i-y_{i+2})\ldots(y_i-y_{i+n})}+$$
$$+x_{i+1}\frac{(y-y_i)(y-y_{i+2})\ldots(y-y_{i+n})}{(y_{i+1}-y_i)(y_{i+1}-y_{i+2})\ldots(y_{i+1}-y_{i+n})}+$$
$$+\ldots+x_{i+n}\times$$
$$\times\frac{(y-y_i)(y-y_{i+1})\ldots(y-y_{i+n-1})}{(y_{i+n}-y_i)(y_{i+n}-y_{i+1})\ldots(y_{i+n}-y_{i+n-1})}.$$

Положив здесь $y=y^*$, мы найдем искомое значение x.

9-4. Численное дифференцирование и интегрирование функций

Численное дифференцирование. Необходимость в численном дифференцировании функций возникает либо тогда, когда значения функций заданы таблично, либо если аналитическая запись производных очень сложна. В обоих случаях для отыскания производной функции $f(x)$ в некоторой точке x составляется интерполяционный полином

$$P_n(x)=y_i+u\frac{\Delta y_i}{1!}+u(u-1)\frac{\Delta^2 y_i}{2!}+$$
$$+u(u-1)(u-2)\frac{\Delta^3 y_i}{3!}+\ldots$$
$$\ldots+u(u-1)(u-2)\ldots(u-n+1)\frac{\Delta^n y_i}{n!};$$

где

$$u=\frac{x-x_i}{h}$$

и за значения производных $f(x)$ принимаются значения производных полинома $P_n(x)$.

Введем обозначения

$$V_m=u(u-1)(u-2)\ldots(u-m+1),$$

откуда

$$\frac{dV_m}{du}=1!\left(\frac{1}{u}+\frac{1}{u-1}+\ldots+\frac{1}{u-m+1}\right)V_m;$$

$$\frac{d^2V_m}{du^2}=2!\left(\frac{1}{u(u-1)}+\frac{1}{u(u-2)}+\ldots+\frac{1}{(u-m+2)(u-m+1)}\right)V_m;$$

$$\frac{d^3V_m}{du^3}=3!\left(\frac{1}{u(u-1)(u-2)}+\frac{1}{u(u-2)(u-3)}+\ldots+\frac{1}{(u-m+3)(u-m+2)(u-m+1)}\right)V_m;$$

. . . . . .

$$\frac{d^m V_m}{du^m} = m!, \quad \frac{d^{m+1} V_m}{du^{m+1}} = 0.$$

При помощи этих формул, а также учитывая, что $\frac{d^k P_n}{dx^k} = \frac{1}{h^k}\frac{d^k P_n}{du^k}$, мы можем записать производные $P_n(x)$ в удобном для вычислений виде

$$\frac{hP_n'}{1!} = \Delta y_i + \left[\frac{1}{u} + \frac{1}{u-1}\right] u(u-1)\frac{\Delta^2 y_i}{2!} + \left[\frac{1}{u} + \frac{1}{u-1} + \frac{1}{u-2}\right] u(u-1)(u-2)\frac{\Delta^3 y_i}{3!} +$$

$$+ \ldots + \left[\frac{1}{u} + \frac{1}{u-1} + \ldots + \frac{1}{u-n+1}\right] u(u-1)(u-2)\ldots(u-n+1)\frac{\Delta^n y_i}{n!},$$

$$\frac{h^2 P_n''}{2!} = \Delta^2 y_i + \left[\frac{1}{u(u-1)} + \frac{1}{u(u-2)} + \frac{1}{(u-1)(u-2)}\right] u(u-1)(u-2)\frac{\Delta^3 y_i}{3!} + \ldots$$

$$\ldots + \left[\frac{1}{u(u-1)} + \frac{1}{u(u-2)} + \ldots + \frac{1}{(u-n+2)(u-n+1)}\right] u(u-1)\ldots(u-n+1)\frac{\Delta^n y_i}{n!},$$

$$\frac{h^3 P_n'''}{3!} = \Delta^3 y_i + \ldots + \left[\frac{1}{u(u-1)(u-2)} + \frac{1}{u(u-2)(u-3)} + \ldots\right.$$

$$\left. + \frac{1}{(u-n+3)(u-n+2)(u-n+1)}\right] u(u-1)\ldots(u-n+1)\frac{\Delta^n y_i}{n!}.$$

Если нужно вычислить производные $f(x)$ в одном из узлов интерполяции $x_{i+\rho}$ (т. е. при $u = \rho$), то надо сначала из каждой квадратной скобки исключить те слагаемые

$$\frac{1}{(u-\rho)(u-\rho-1)} + \frac{1}{(u-\rho)(u-\rho+1)},$$

которые не содержат множителя $u-\rho$, и в полученных таким образом формулах заменить $u-\rho$ единицей. В этом случае выгоднее заранее вычислить величины коэффициентов при $\Delta^k y_i$ (в рассматриваемом случае $u = \rho$).

Численное интегрирование. Пусть на интервале (a, b) задана функция $f(x)$ и требуется вычислить интеграл

$$I = \int_a^b f(x)\,dx.$$

Процесс приближенного отыскания величины I называют обычно „механическими квадратурами". Они применяются каждый раз, когда интеграл не берется в конечном виде. Общая схема приближенного вычисления интеграла заключается в следующем: функцию $f(x)$ заменяют интерполяционным полиномом $P_n(x)$ и за величину интеграла принимают интеграл I_n

$$I_n = \int_a^b P_n(x)\,dx.$$

Рассмотрим случай, когда узлы интерполяции делят интервал (a, b) на n равных частей, длины $h = \frac{b-a}{n}$:

$$x_0 = a, \quad x_1 = a + h,$$

$$x_2 = x_0 + 2h, \ldots, \quad x_n = a + nh = b.$$

Обозначив

$$\Lambda_n(x) = (x-x_0)(x-x_1)(x-x_2)\ldots(x-x_n),$$

мы можем записать интерполяционную формулу Лагранжа в следующем компактном виде

$$P_n(x) = \sum_{i=0}^{n} f(x_i)\frac{\Lambda_n(x)}{(x-x_i)\Lambda'(x_i)}.$$

Интегрируя это выражение, получим

$$I_n = \sum_{i=0}^{n} f(x_i)\int_a^b \frac{\Lambda_n(x)}{(x-x_i)\Lambda'(x_i)}\,dx.$$

Подынтегральное выражение в этой формуле не зависит от функции $f(x)$, а зависит только от степени интерполяционного полинома и интервала (a, b). Следовательно, эти интегралы можно вычислить для $n = 1, 2, 3, \ldots$ и, например, интервала $(0, 1)$. Сделав это, мы запишем предыдущую формулу в виде

$$I_n = (b-a)\left(C_0^{(n)} f(a) + C_1^{(n)} f(a+h) + \right.$$

$$\left. + C_2^{(n)} f(a+2h) + \ldots + C_n^{(n)} f(b)\right), \quad (9\text{-}11)$$

которая верна для любого интервала (a, b) и где

$$C_i^{(n)} = \int_0^1 \frac{\Lambda_n(x)}{(x-x_i)\Lambda_n'(x_i)}\,dx$$

Величины $C_i^{(n)}$ называются коэффициентами Котеса. Их численные значения для $n = 1, 2, \ldots, 6$ приведены в табл. 9-2.

Если, например, интерполирование ведется по четырем точкам ($n = 3$), то

$$I_3 = \frac{(b-a)}{8}\{f(a) + 3f(a+h) + 3f(a+2h) + f(b)\}.$$

Чтобы оценить ошибку D_n, допускаемую при численном интегрировании, можно рассмотреть формулу (9-10), из которой следует, что

$$|D_n| < \frac{M_{n+1}}{(n+1)!}\int_a^b |\Lambda_n(x)|\,dx,$$

где M_{n+1} — максимум модуля $n+1$-й производной $f^{(n+1)}(x)$ на интервале (a, b). Так как каждый из множителей, входящих в $\Lambda_n(x)$, по модулю не больше длины интервала $L = b - a$, то

$$|\Lambda_n(x)| < L^{n+1},$$

откуда

$$|D_n| < \frac{M_{n+1}}{(n+1)!}L^{n+2}. \quad (9\text{-}12)$$

Так как $\frac{L^{n+2}}{(n+1)!} \to 0$ при $n \to \infty$, то, увеличивая степень n интерполяционного полинома, можно сделать ошибку как угодно малой (считая, конечно, что M_{n+1} растут не слишком быстро). Но такое увеличение n связано с очень громоздкими вычислениями, и поэтому на практике поступают иначе. Разделим интервал (a, b) на N одинаковых интервалов длины $l = \frac{L}{N}$. Пусть точки деления будут

$$a_0 = a, \ a_1, \ a_2, \ldots, \ a_N = b.$$

Формулу (9-11) можно применить ко всему интервалу (a, b), а к каждому из интервалов (a_k, a_{k+1}) в отдельности (на каждом интервале a_k, a_{k+1} строится свой интерполяционный полином). Тогда интеграл I_n станет равным сумме N частичных интегралов. Если применить формулу (9-12) к каждому из этих ин-

Таблица 9-2

Коэффициенты Котеса

n	$C_i^{(n)}$	Шаг интерполирования $h = \frac{b-a}{n}$	Интерполяционные узлы
1	$\frac{1}{2}, \frac{1}{2}$	$b-a$	a, b
2	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$	$\frac{b-a}{2}$	$a, a+h, b$
3	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	$\frac{b-a}{3}$	$a, a+h, a+2h, b$
4	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	$\frac{b-a}{4}$	$a, a+h, a+2h, a+3h, b$
5	$\frac{19}{288}, \frac{75}{288}, \frac{50}{288}, \frac{50}{288}, \frac{75}{288}, \frac{19}{288}$	$\frac{b-a}{5}$	$a, a+h, a+2h, a+3h, a+4h, b$
6	$\frac{41}{840}, \frac{216}{840}, \frac{27}{840}, \frac{272}{840}, \frac{27}{840}, \frac{216}{840}, \frac{41}{840}$	$\frac{b-a}{6}$	$a, a+h, a+2h, a+3h, a+4h, a+5h, b$

тегралов в отдельности, то получается, что допустимая ошибка будет меньше, чем

$$N\frac{M_{n+1}}{(n+1)!}l^{n+1}=l^{n+1}\frac{M_{n+1}}{(n+1)!}.$$

Уменьшая длину l частичного интервала, и применяя формулы интерполирования полиномами n, можно достигнуть любой требуемой точности.

Формулы механических квадратур для частичных интервалов можно просуммировать. В частности, при $n=1$ получается так называемая формула трапеций

$$I\approx\frac{b-a}{2N}\{f(a_0)+2f(a_1)+$$
$$+2f(a_2)+\ldots+2f(a_{N-1})+f(a_N)\},$$

ошибка которой меньше

$$l^2\frac{M_2(b-a)}{2}.$$

При $n=2$ получается формула Симпсона

$$I\approx\frac{b-a}{6N}\{f(a_0)+4f\left(\frac{a_0+a_1}{2}\right)+$$
$$+2f(a_1)+4f\left(\frac{a_1+a_2}{2}\right)+2f(a_2)+$$
$$+\ldots+2f(a_{N-1})+4f\left(\frac{a_{N-1}+a_N}{2}\right)+f(a_N)\},$$

ошибка которой меньше, чем

$$l^4\frac{M_4(b-a)}{6}.$$

Геометрический смысл этих формул чрезвычайно прост: в каждом частичном интервале кривая $y=f(x)$ заменяется в первом случае хордой, проходящей через крайние точки (т. е. интеграл равен площади, полученной таким образом трапеции), а во втором — параболой, проходящей через крайние точки и середину.

§ 3. Решение обыкновенных дифференциальных уравнений

Постановка задачи. Пусть ищется решение дифференциального уравнения

$$y'=f(x,\ y),\qquad (9\text{-}13)$$

удовлетворяющее условиям

$$x_0\leqslant x\leqslant X,\ y(x_0)=y_0.\qquad (9\text{-}14)$$

Предположим, что решение этой задачи существует и единственно. Если мы не можем выразить его в виде формул или если формулы неудобны для практических расчетов, то ставится вопрос о приближенном решении задачи (9.13), (9.14): найти в некоторых точках $x_0<x_1<x_2<\ldots<x_n=X$ значения $y_1, y_2, \ldots, y_n$, приближенно равные значениям $y(x_1), y(x_2), \ldots, y(x_n)$ точного решения $y(x)$ [через y_i обозначается приближенное значение функции $y(x)$ в точке x_i].

Разложение в ряд Тейлора. Предположим, что решение $y(x)$ уравнения (9.13) разлагается в ряд Тейлора в окрестности каждой точки x_i:

$$y(x)=y(x_i)+\frac{(x-x_i)}{1!}y'(x_i)+$$
$$+\frac{(x-x_i)^2}{2}y''(x_i)+\ldots\qquad (9\text{-}15)$$

Дифференцируя равенство (9.13) достаточное число раз (считая y функцией x) можно найти любую производную $y^{(k)}(x_i)$ и, следовательно, вычислить ряд (9.15) с любой степенью точности. Таким образом,

$$y'(x_i)=[f(x,\ y)]_{\substack{x=x_i\\ y=y(x_i)}},$$
$$y''(x_i)=[f_x(x,\ y)+yf'_y(x,\ y)]_{\substack{x=x_i\\ y=y(x_i)}}=$$
$$=[f_x+f_yf]_{\substack{x=x_i\\ y=y(x_i)}},$$
$$y'''(x_i)=[f_{xx}+yf'_{xy}+(f_x+yf'_y)f'_y+$$
$$+f(f'_x+f'_yy)]_{\substack{x=x_i\\ y=y(x_i)}}=$$
$$=[f''_{xx}+2ff_{xy}+f_xf_y+$$
$$+ff_y^2+f^2f_{yy}]_{\substack{x=x_i\\ y=y(x_i)}}\qquad (9\text{-}16)$$

Подставляя полученные значения $y^{(k)}(x_i)$ в формулу (9.15) и полагая $x=x_{i+1}$, $x_{i+1}-x_i=h$, получим окончательно

$$y(x_{i+1})=y(x_i)+\frac{h}{1!}f(x_i,\ y(x_i))+$$
$$+\frac{h^2}{2!}[f'_x(x_i,\ y(x_i))+$$
$$+f(x_i,y(x_i))f'_y(x_i,y(x_i))]+$$
$$+\frac{h^3}{3!}[f''_{xx}(x_i,y(x_i))+\ldots]+\ldots.$$

Удерживая в этой формуле достаточное число членов и полагая $i=0,1,\ldots,n$, можно последовательно вычислить приближенные значения $y_1, y_2, \ldots, y_n$ с любой степенью точности.

Пусть, например, задано уравнение

$$\frac{dy}{dx}=-y\operatorname{tg}x\equiv f(x,\ y),\ y(0)=1,$$

решением которого, как легко проверить, является функция $y=\cos x$. Дифференцируя это уравнение, найдем

$$f'_x=\frac{-y}{\cos^2x};\quad f'_y=-\operatorname{tg}x,$$
$$f''_{xx}=-\frac{2y\sin x}{\cos^3x},$$
$$f''_{xy}=-\frac{1}{\cos^2x};\quad f''_{yy}=0.\qquad (9\text{-}17)$$

Таким образом, $y'(0)=0$, $y''(0)=-1$; $y'''(0)=0$, и, следовательно, значение $y(x)$ в точке $x_1=h$ ($x_0=0$) равно

$$y(x_1)=1-\frac{h^2}{2!}+y^{(IV)}(x_1)\frac{h^4}{4!}+\ldots$$

При h достаточно малом остаток $y^{(IV)}(x_1)\frac{h^4}{4!}+\ldots$ мал и приближенное значение y_1 величины $y(x_1)$ равно:

$$y_1=1-\frac{h^2}{2!}\approx\cos h.$$

Подставляя x_1 и y_1 в равенства (9.17), можно найти приближенное значение y_2 в некоторой точке x_2 и т. д. Итак, указанный метод сводится к вычислению отрезка ряда Тейлора (9.15) при помощи равенств (9.16). Если функции $f(x,\ y)$ достаточно сложна, то такие вычисления становятся весьма трудоемкими. Для уменьшения вычислительной работы желательно научиться определять значение отрезка ряда Тейлора с достаточной точностью, не прибегая к формулам (9.16). Указанный прием лежит в основе трех нижеследующих методов.

Метод Эйлера. Если ограничиться двумя членами ряда Тейлора, то получим формулу Эйлера

$$y_{i+1}=y_i+hf(x_i,y_i);\qquad (9\text{-}18)$$
$$i=0,1,\ldots,n-1.$$

Положив в этой формуле $i=0$, получим значение y_1 (y_0 задано в условии задачи); положив затем $i=1$ и используя вычисленное ранее значение y_1, найдем y_2 и т. д.

Усовершенствованный метод Эйлера — Коши. Значение трех первых членов ряда (9.15) с точностью до малых более высокого порядка можно получить по формуле Коши

$$y_{i+1}=y_i+\frac{1}{2}(k_1+k_2),\qquad (9\text{-}19)$$

где

$$k_1=hf(x_i,y_i),\quad k_2=hf(x_{i+1},\ y_i+k_1);$$
$$i=0,1,\ldots,n-1.$$

Метод Рунге — Кутта. Наконец, значение первых пяти членов ряда (9.15) с точностью до малых более высокого порядка можно получить по формуле

$$y_{i+1}=y_i+\frac{1}{6}(k_1+2k_2+2k_3+k_4),\qquad (9\text{-}20)$$

где

$$k_1=hf(x_i,\ y_i),$$
$$k_2=hf\left(x_i+\frac{h}{2},\ y_i+\frac{k_1}{2}\right),$$
$$k_3=hf\left(x_i+\frac{h}{2},\ y_i+\frac{k_2}{2}\right),$$
$$k_4=hf(x_i+h,\ y_i+k_3);$$
$$i=0,1,\ldots,n-1.$$

Заметим, что для определения первых пяти членов ряда (9.15) по формулам (9.16) необходимо вычислить значение функции f и всех ее смешанных производных до третьего порядка включительно, общее число которых (вместе с функцией f) равно 10. Между тем по формулам Рунге — Кутта достаточно вычислить значение функции f только в четырех точках. Кроме того, формулы (9.16) требуют вычисления произведений этих производных, что также связано с большой затратой времени.

Порядок точности приближенных формул. Если при помощи некоторого приближенного метода вычисляется отрезок, содержащий с точностью до малых более высокого порядка $n+1$ член ряда (9.15), то говорят, что этот метод имеет точность n-ного порядка, так как при этом вычисляются члены, содержащие h в степени до n включительно. Например, метод Рунге — Кутта является методом четвертого порядка.

Можно показать, что разность $y(x_i) - y_i$ стремится к нулю как h, h^3 и h^4 соответственно для методов Эйлера, Эйлера—Коши и Рунге—Кутта. Это утверждение становится очевидным, если при вычислении y_i пользоваться не приближенным значением y_{i-1}, а точным $y(x_{i-1})$. Однако ошибка на i-ом шаге складывается не только из ошибки, возникающей от отбрасывания остатка ряда Тейлора, но и из накапливающейся от шага к шагу ошибки, связанной с заменой $y(x_k)$ на y_k, $k = 1, 2, \ldots, i-1$.

Из указанных выше трех методов метод Рунге — Кутта является наиболее точным. В свою очередь, метод Эйлера—Коши дает большую точность, чем метод Эйлера.

Метод Рунге – Кутта легко распространяется на системы дифференциальных уравнений.

Соответствующие формулы приведены в § 7.2.

Программа интегрирования дифференциальных уравнений методом Рунге – Кутта описана в § 1.2, 7.2, 7.3.

Определение точности приближенного решения. При выборе величины шага h с сравнительно простых соображений получается величина

$$S = \left|\frac{\partial f}{\partial y}\right| h,$$

называемой характеристикой шага. При изменении величины $\left|\frac{\partial f}{\partial y}\right|$ необходимо менять шаг h так, чтобы все время выполнялось равенство

$$S = \left|\frac{\partial f}{\partial y}\right| \cdot h = \text{const}. \qquad (9\text{-}21)$$

Заметим, что правые части формул (9-18), (9-20) содержат только значения x_i и y_i в одной предыдущей точке и поэтому на каждом этапе шаг можно произвольно изменять. Для получения умеренной точности шаг выбирают таким, чтобы константа в (9-21) равнялась 0,1—0,05.

В методе Рунге — Кутта в качестве оценки величины шага рассматривают также отношение

$$\frac{|k_3 - k_2|}{|k_2 - k_1|},$$

которое не должно превышать нескольких сотых.

При численном интегрировании дифференциальных уравнений точность решения можно оценивать сравнением на каждом шаге результатов, получаемых целым и половинным шагом. При этом величину шага изменяют таким образом, чтобы на каждом шаге разность решений, полученных целым и половинным шагом, находились в установленных пределах (см. § 7.3).

Экстраполяционный метод Адамса. Другой метод приближенного решения уравнения

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0, \quad x_0 \leqslant x \leqslant X \qquad (9\text{-}13)$$

основан на следующем. Как и выше, разобьем интервал (x_0, X) точками $x_0 < x_1 < x_2 < \ldots < x_n = X$ и проинтегрируем уравнение (9-13) в пределах от x_i до x_{i+1}. Получим

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y(x))\,dx. \qquad (9\text{-}22)$$

Предположим, что каким-нибудь способом удалось вычислить приближенное значение R_{i+1} интеграла, стоящего в правой части этого равенства. Подставив в выражение (9-22) величину R_{i+1}, получим формулу

$$y_{i+1} = y_i + R_{i+1}, \quad i = 0, 1, \ldots, n-1$$

приближенного интегрирования дифференциального уравнения (9-13). Величина R_{i+1} может быть определена различными способами, в зависимости от чего получаются различные приближенные формулы. Если, например, заменить значение подынтегральной функции ее значением в левой точке x_i, то мы вновь получим формулу Эйлера

$$y_{i+1} = y_i + (x_{i+1} - x_i) f(x_i, y_i).$$

Среди большого числа подобных формул рассмотрим лишь экстраполяционную формулу Адамса. Предположим, что приближенные значения $y_0, y_1, \ldots, y_i$ уже определены. Заменим подынтегральную функцию в выражении (9-22) интерполяционным полиномом Ньютона $P_{i-r,i}(x)$, который в точках $x_{i-r}, x_{i-r+1}, \ldots, x_{i-2}, x_{i-1}, x_i$ принимает значения $f(x_{i-r}, y_{i-r}), \ldots, f(x_{i-1}, y_{i-1}), f(x_i, y_i)$. Для простоты запишем этот полином в виде:

$$P_{i-r,i} = f_i + \frac{u}{1!}\nabla f_i + \frac{u(u+1)}{2!}\nabla^2 f_i + \frac{u(u+1)(u+2)}{3!}\nabla^3 f_i + \ldots + \frac{u(u+1)\ldots(u+r-1)}{r!}\nabla^r f_i,$$

$$u = \frac{x - x_i}{h},$$

$$h = x_{i+1} - x_i, \quad f_i = f(x_i, y_i).$$

Здесь, через ∇^k обозначены обратные разности, которые определяются аналогично обычным разностям (см. § 9.3), но только используют значения функции, идущие налево от входа x_i. Таким образом,

$$\nabla f_i = f_i - f_{i-1},$$

$$\nabla^2 f_i = \nabla(\nabla f_i) = f_i - 2f_{i-1} + f_{i-2}.$$

Проинтегрировав этот полином в пределах от x_i до x_{i+1}, мы получим формулу Адамса

$$y_{i+1} = y_i + h\left(f_i + \frac{1}{2}\nabla f_i + \frac{5}{12}\nabla^2 f_i + \frac{3}{8}\nabla^3 f_i + \frac{251}{720}\nabla^4 f_i + \frac{95}{288}\nabla^5 f_i + \frac{19\,087}{60\,480}\Delta^6 f_i + \ldots\right). \qquad (9\text{-}23)$$

Число удерживаемых в этой формуле членов зависит от требуемой точности.

Чтобы применить формулу Адамса, необходимо определить первые r значений $y_1, y_2, \ldots, y_r$. Делается это обычно одним из рассмотренных выше методов. При этом эти значения должны быть определены особенно точно, чтобы уменьшить влияние погрешности первоначальных значений $y_1, \ldots, y_r$ на весь дальнейший ход решения задачи.

Величина шага h определяется на основании введенной выше характеристики шага. Заметим, что в этом случае изменение шага не может быть осуществлено так легко, как это имело место в предыдущих формулах. Действительно, формула (9-23) содержит значения функции в предшествующих равноотстоящих точках. Следовательно, уменьшение шага на некотором этапе счета потребует дополнительного определения значений переменных в промежуточных точках, что может быть произведено при помощи рассмотренных выше интерполяционных формул.

ПРИЛОЖЕНИЯ

SUPPLEMENTS

Declassified in Part - Sanitized Copy Approved for Release @ 50-Yr 2014/02/10 : CIA-RDP81-01043R003800160006-3

ПРИЛОЖЕНИЕ 1

ПЕРЕЧЕНЬ ОПЕРАЦИЙ МАШИНЫ М-3

Код операции	Условное обозначение операции	Наименование операции
00	+	К содержимому первого адреса прибавляется содержимое второго адреса, результат записывается в ЗУ по второму адресу
10	·	К содержимому первого адреса прибавляется содержимое второго адреса, но результат в ЗУ не записывается
20	↓+	К результату предыдущего действия прибавляется содержимое первого адреса, и результат записывается в ЗУ по второму адресу
30	·	К результату предыдущего действия прибавляется содержимое первого адреса, но в ЗУ запись не производится
40	+||	К содержимому первого адреса прибавляется содержимое второго адреса. Результат записывается в ЗУ по второму адресу и печатается
50	'+ .	К модулю содержимого второго адреса прибавляется модуль содержимого первого адреса, но результат в ЗУ не записывается
60	↓ . ||	К результату предыдущего действия прибавляется содержимое первого адреса Результат записывается в ЗУ по второму адресу и печатается
70	↓ +|.	К модулю результата предыдущего действия прибавляется модуль содержимого первого адреса, но в ЗУ запись не производится
01	—	Из содержимого второго адреса вычитается содержимое первого адреса, результат записывается в ЗУ по второму адресу
11	—	Из содержимого второго адреса вычитается содержимое первого адреса, но результат в ЗУ не записывается
21	↓	Из результата предыдущего действия вычитается содержимое первого адреса, а результат записывается в ЗУ по второму адресу
31	·	Из результата предыдущего действия вычитается содержимое первого адреса, результат в ЗУ не записывается
41	||	Из содержимого второго адреса вычитается содержимое первого адреса, результат записывается по второму адресу и печатается
51	-	Из модуля содержимого второго адреса вычитается модуль содержимого первого адреса, результат в ЗУ не записывается
61	↓ — ||	Из результата предыдущего действия вычитается содержимое первого адреса Результат записывается по второму адресу и печатается
71	↓|—	Из модуля результата предыдущего действия вычитается модуль содержимого первого адреса, но в ЗУ запись не производится
02		Содержимое второго адреса делится на содержимое первого адреса Результат записывается в ЗУ по второму адресу
12		Содержимое второго адреса делится на содержимое первого адреса Результат в ЗУ не записывается
22	÷	Результат предыдущего действия делится на содержимое первого адреса, и результат записывается в ЗУ по второму адресу
32	↓ :	Результат предыдущего действия делится на содержимое первого адреса Результат в ЗУ не записывается
42	||	Содержимое второго адреса делится на содержимое первого адреса. Результат записывается в ЗУ по второму адресу и печатается
52	| :.	Модуль содержимого второго адреса делится на модуль содержимого первого адреса но результат в ЗУ не записывается
62	↓ ||	Результат предыдущего действия делится на содержимое первого адреса. Результат записывается в ЗУ по второму адресу и печатается
72	↓| |.	Модуль результата предыдущего действия делится на модуль содержимого первого адреса, но в ЗУ запись не производится

Продолжение

Код операции	Условное обозначение операции	Наименование операции
03	×	Содержимое второго адреса умножается на содержимое первого адреса. Результат записывается в ЗУ по второму адресу
13	×.	Содержимое второго адреса умножается на содержимое первого адреса. Результат в ЗУ не записывается
23	↓×	Результат предыдущего действия умножается на содержимое первого адреса. Результат записывается в ЗУ по второму адресу
33	↓×.	Результат предыдущего действия умножается на содержимое первого адреса. Результат в ЗУ не записывается
43	×||	Содержимое второго адреса умножается на содержимое первого адреса. Результат записывается в ЗУ по второму адресу и печатается
53	|×|.	Модуль содержимого второго адреса умножается на модуль содержимого первого адреса, но результат в ЗУ не записывается
63	↓×||	Результат предыдущего действия умножается на содержимое первого адреса. Результат записывается в ЗУ по второму адресу и печатается
73	↓|×|.	Модуль результата предыдущего действия умножается на модуль содержимого первого адреса, но в ЗУ запись не производится
06	∧	Содержимое второго адреса логически умножается на содержимое первого адреса. Результат записывается в ЗУ по второму адресу
16	∧.	Содержимое второго адреса логически умножается на содержимое первого адреса Результат в ЗУ не записывается
26	↓∧	Результат предыдущего действия логически умножается на содержимое первого адреса. Результат записывается в ЗУ по второму адресу.
36	↓∧.	Результат предыдущего действия логически умножается на содержимое первого адреса. Результат в ЗУ не записывается
46	∧||	Содержимое второго адреса логически умножается на содержимое первого адреса. Результат записывается в ЗУ по второму адресу и печатается
56	|∧|.	Модуль содержимого второго адреса логически умножается на модуль содержимого первого адреса Результат в ЗУ не записывается
66	↓∧||	Результат предыдущего действия логически умножается на содержимое первого адреса. Результат записывается в ЗУ по второму адресу и печатается
76	↓|∧|.	Модуль результата предыдущего действия логически умножается на содержимое первого адреса, но в ЗУ не записывается
07; 27	Вв	Ввод чисел. Число с перфорированной ленты записывается по второму адресу На регистре не сохраняется
05; 15	ПЧ	Содержимое первого адреса передается по второму адресу и сохраняется в АУ на регистре второго числа
45, 55	ПЧП	Содержимое первого адреса передается по второму адресу и печатается, но в АУ не сохраняется
24	ПУ	Передача управления. Следующая команда берется по первому адресу. Результат предыдущего действия записывается по второму адресу и сохраняется в АУ
64	ПУП	То же, что операция 24, но число еще и печатается
74	ПУ2	Следующая команда берется по второму адресу. В АУ остается модуль результата предыдущего действия
34	УП	Условный переход. Следующая команда берется по второму адресу, если результат предыдущего действия имел знак «+», и по первому адресу, если результат предыдущего действия имел знак «—»
00, 14, 44, 54, 17, 37, 47, 77	Стоп	Операции Стоп отличаются друг от друга содержанием регистров арифметического узла, а также селективного и пускового регистров

ПРИЛОЖЕНИЕ 2

ПЕРЕЧЕНЬ ОПЕРАЦИЙ МАШИНЫ „УРАЛ"

Код операции	Условное обозначение операции	Условие выработки сигнала «—1»	Наименование операции	Примечание
01	Сл 1а	$z<0$	Алгебраическое сложение числа в сумматоре с числом в ячейке а	—
02	Сл 2а	$z<0$	Гашение сумматора и посылка в него содержимого ячейки а	—

Продолжение

Код операции	Условное обозначение операции	Условие выработки сигнала ω = 1	Наименование операции	Примечание
03	Вч1а	$z<0$	Алгебраическое вычитание числа, находящегося в ячейке a из числа в сумматоре	—
04	Вч2а	$z<0$	Алгебраическое вычитание модуля числа, находящегося в ячейке a из модуля числа, находящегося в регистре	—
05	Ум1а	$z<0$	Умножение числа в регистре на число в ячейке a и прибавление результата к числу в сумматоре	Операция Ум1a позволяет вычислять суммы парных произведений вида $a_1b_1+a_2b_2+$ без пересылки промежуточных сумм в ЗУ на магнитном барабане
06	Ум2а	$z<0$	Умножение числа в сумматоре на число в ячейке a	Операция Ум2a позволяет вычислять произведения вида $a_1a_2a_3$ без пересылки промежуточного результата в ЗУ на магнитном барабане
07	Дa	$z<0$	Деление числа в сумматоре на число в ячейке a	—
10	Фза	$z=0$	Прифомирование числу в сумматоре знака числа в ячейке a	—
11	Сдa	$z=0$	Сдвиг числа в регистре на количество разрядов, соответствующее числу в сумматоре, расположенному в разрядах с 19-го по 35-й	При выполнении операции Сд переполнение блокируется. Если число в сумматоре положительно, то сдвиг производится влево, если отрицательно, то — вправо
12	Ила	$z=0$	Выделение части числа в сумматоре посредством числа в ячейке a по правилу поразрядного логического умножения	—
13	Фра	$z=0$	Поразрядное логическое сложение числа в сумматоре с числом в ячейке a	В каждом разряде производится сложение по правилам $0+0=0$, $0+1=1+0=1$, $1+1=1$, без образования единиц переноса
14	Сра	$x \neq y$	Сравнение содержимого сумматора с числом в ячейке a	При несовпадении набора разрядов в сумматоре и в ячейке a в соответствующие разряды результата записываются единицы
15	Нра	$z=0$	Нормализация числа в сумматоре с записью полученного результата в ячейку a	В сумматоре фиксируется число, соответствующее количеству сдвигов
16	Пса	$z<0$	Пересылка числа из сумматора в ячейку a	Число в сумматоре сохраняется
17	Пра	Если посылаемое в регистр число равно нулю	Очищение регистра и пересылка в него числа из a	
20	Пск	—	Очищение сумматора и запись в него числа k, стоящего в адресной части команды	—
26	Сиа	$z<0$	Алгебраическое сложение числа в сумматоре с числом в ячейке a	То же, что Са1а, но блокирует останов при переполнении

Продолжение

Код операции	Условное обозначение операции	Условие выработки сигнала ω = 1	Наименование операции	Примечание
21	Е1a	—	Условная передача управления (первая операция передачи управления)	В зависимости от сигнала, выработанного в предыдущем такте, управление передается при ω=0 следующей команде, при ω=1 команде, находящейся в ячейке a
22	Е2a	—	Безусловная передача управления (вторая операция передачи управления)	Управление передается команде, находящейся в ячейке a, независимо от значения ω, выработанного в предыдущем такте
23	Е3k	—	Третья операция передачи управления	По команде Е3k в зависимости от положения ключа k на пульте управления исполняется либо пропускается одна команда, следующая непосредственно за командой Е3k. Номер ключа k меняется от 1 до 7
24	Е4a	—	Четвертая операция передачи управления, обеспечивающая повторение цикла нужное число раз	Ставится в конце цикла. Управление передается ячейке a, если цикл еще не повторился нужное число раз, и следующей команде — в противном случае
25	Нцn	—	Подготовительная операция для повторения цикла нужное число раз	Ставится перед циклом. При каждом повторении цикла в отрицательных командах увеличиваются на единицу адреса исполних ячеек (12-й разряд команды=0) или на две единицы (12-й разряд команды=1). В первом случае цикл повторяется $n+1$ раз, во втором $1+n/2$ раз
30	Изa	—	Операция изменения команд	При выполнении команды содержимое ячейки a передается на регистр команд и суммируется со следующей командой
37	Осa	—	Остановка машины	Машина останавливается и число из ячейки a переписывается в сумматор в прямом коде
31	Лп$a_н$ О1С ОО$a_к$	—	Переписка информации с перфоленты в ЗУ на магнитном барабане	Содержимое зоны C перфоленты переписывается в ЗУ на магнитном барабане в группу ячеек $a_н - a_к$
31	Лм$a_н$ О3С О2$a_к$	—	Переписка информации с магнитной ленты в ЗУ на магнитном барабане	Содержимое зоны C магнитной ленты переписывается в группу ячеек $a_н - a_к$ ЗУ на магнитном барабане
31	Лм$a_н$ О3С ОО$a_к$	—	Переписка информации из ЗУ на магнитном барабане на магнитную ленту	Содержимое группы ячеек $a_н - a_к$ ЗУ на магнитном барабане переписывается в зону C магнитной ленты
32	Пч	—	Выдача на печать содержимого сумматора	При включенном ключе перфорирования содержимое сумматора перфорируется
34	Ии	—	Образование интервала в печати на бумаге	

Примечания: 1. z — содержимое сумматора после выполнения операции.
2. При выполнении [illegible] вырабатываются специальные управляющие сигналы: [illegible] сигнал условной передачи управления («сигнал ω») и сигнал переполнения («сигнал φ»). [illegible]
3. Сигнал φ = 1 вырабатывается при операциях [illegible] [illegible]

Supplement III

ЛИТЕРАТУРА

1 Базилевский Ю. Я., Универсальная вычислительная машина для инженерных исследований, «Приборостроение», 1956, № 4

2 Базилевский Ю. Я., Универсальная электронная вычислительная машина «Стрела», «Приборостроение», 1957, № 3

3 Белынский В. В., Долкарт В. М., Каган Б. М., Лопато Г. П., Матюхин Н. Я., Малогабаритная электронная вычислительная машина М-3, изд. ФВИНТИ 1957

4 Брук И. С., Быстродействующая электронная вычислительная машина М-2, «Электричество», 1956, № 9

5 Брук И. С., Матюхин Н. Я., Белынский В. В., Иосифьян А. Г., Каган Б. М., Долкарт В. М., Лопато Г. П., Малогабаритная универсальная электронная вычислительная машина М-3, «Электричество», 1958, № 1

6 Быстродействующие вычислительные машины, перевод с английского, под ред. Панова Д. Ю., Изд. иностр. лит., 1952

7 Веников В. А., Иванов-Смоленский А. В., Горушкин В. Н., К вопросу об эффективности форсирования возбуждения генераторов, «Электричество», 1955, № 1

8 Веников В. А., Литкенс И. В., О влиянии регулирования возбуждения на пропускную способность дальних электропередач, «Электричество», 1955, № 11

9 Жданов П. С., Устойчивость электрических систем, Госэнергоиздат, 1948

10 Каган Б. М., Расчет устойчивости электрических систем автоматического регулирования на цифровых вычислительных машинах, «Вестник электропромышленности», 1957, № 10.

11 Каган Б. М., Гурин Я. С., Долкарт В. М., Применение автоматических электронных счетных машин для расчета серии электрических машин, сб. Работы Министерства электротехнической промышленности СССР по автоматизации и механизации народного хозяйства, т. 2, 1956.

12 Келдыш М. В., Лаврентьев М. А., Шура-Бура М. Р., Математические вопросы теории счетных машин, «Вестник Академии наук СССР», 1956, № 11

13 Китов А. И., Электронные цифровые машины, Изд. «Советское радио», 1956.

14 Коллатц Л., Численные методы решения дифференциальных уравнений, Изд. Иностр. лит., 1953

15 Крылов А. Н., Лекции о приближенных вычислениях, Гостехиздат, 1954.

16 Лебедев С. А., Электронные вычислительные машины, Изд. АН СССР, 1956

17 Лебедев С. А., Электронные вычислительные машины, Доклад на сессии АН СССР 1956.

18 Лебедев С. А., Исследование искусственной устойчивости, Сборник «Устойчивость электрических систем», Труды ВЭН, вып. 40, Госэнергоиздат, 1940.

19 Ляпунов А. М., Общая задача устойчивости движения, Гостехиздат, 1950.

20 Мидл В. Э., Численное решение дифференциальных уравнений, Изд. Иностр. лит., 1955.

21 Сахаров И. Е., Тер-Микаэлян Т. М., Расчет критических скоростей роторов турбогенераторов на электронной вычислительной машине, «Вестник электропромышленности», 1957, № 10.

22 Фадеева В. Н., Вычислительные методы линейной алгебры, Гостехиздат, 1950.

23. Хаусхолдер А. С., Основы численного анализа, Изд. ИЛ, 1956.

24. Цукерник Л. В., Кочанова Н. А., Анализ статической устойчивости сложных энергосистем при помощи электронных счетных машин, «Электричество», 1957, № 7.

25. Цымкин Я. З., Бромберг П. В., О степени устойчивости, «Изв. АН СССР ОТН», 1945, № 12.

26. Этерман И. И., Гончинская Г. Д., Караваишкина Г. И., Решение математических задач на универсальной цифровой машине «Урал», «Приборостроение», 1956, № 5.

27. Johnson D. L., Ward J. B., The Solution of Power System Stability Problems by Means of Digital Computers, Power Apparatus and Systems, 1957, № 28.

28. Cypser R. J., Computer Search for Economical Operation of a Hydrothermal Electric System, Power Apparatus and Systems, 1954, № 14.

29. Gilman A. F., Kirchmayer L. K., Haberman R. J., Thomas R. W., Automatic Digital Computer Applied to Generation Scheduling, Power Apparatus and Systems, 1954, № 14.

30. Gilman A. F., Habermann R. J. Henderson J. M., Kirchmayer L. K., Digital Calculation of Network Impedances, Power Apparatus and Systems, 1955, № 21.

31. Ward J. B., Hale A. W., Digital Computer Solution of Power-Flow Problems, Power Apparatus and Systems, 1956, № 24.

32. Williams S. B., Abetti P. A., Magnuson E. F., Application of Digital Computers to Transformer Design, Power Apparatus and Systems, 1956, № 35.

33. Velnott C. O., Induction Machinery Design Being Revolutionized by the Digital Computer, Power Apparatus and Systems, 1957, № 28.

34. Hunt P. M., The Electronic Digital Computer in Aircraft Structural Analysis, Aircraft Engineering, 1956, 28, ч. I, № 328, ч. II, № 326.

35. Mazelesky B., O'Connel R. F., The Intergrated Use of Analog and Digital Computing Machines for Aircraft Dynamic Load Problems, J. Aeronaut. Sci., 1956, 23, № 8.

36. Black O., Electronic Computers in Optical Design, Research, 1956, 9, № 8.

37. Lavesley R. K., The Application of Electronic Digital Computer to Some Problems of Structural Analysis Struct. Engineer, 1956, 34, № 1.

38. Coob J. R., McIntire R. L., Natural Gasolines Like Robot Calculators, Oil Gas Journal, 1956, 54, № 50.