Z. M. ZORZOR-WINTER

Integration of Common Differential Equation Systems by Numerical Methods of the Runge-Kutta Type on Digital Electronic Computers

The aims of this paper are to introduce those concerned to the solving of common differential equation systems by numerical procedures in one of the types of methods most used at present, methods of the Runge-Kutta type (methods with free steps).

1. DESCRIPTION OF METHOD

Given the differential equation:

$$y' = f(x, y) \tag{1}$$

with the initial conditions x_0 and $y_0 = y(x_0)$, where the function $f(x,y)$ satisfies the conditions of the theorem of existence and uniqueness of the solution for a given interval in R^2, our purpose will be, when unable to find an explicit form of the solution, to determine it point by point by numerical methods.

Let us assume that the curve arc $M_0\ M_n$ in Fig. 1 represents the solution of equation (1) in the interval x_0, x_n and the point M_0 has the co-ordinates (x_0, y_0). The simplest manner by which an approximate solution can be determined is Euler's method (tangent method). The method consists in identifying, for a small curve arc, the tangent with the curve itself. The tangent equation at point M_0 will be:

$$y = y_0 + (x - x_0)\, f(x_0, y_0).$$

Instead of point M_1 on the curve, corresponding to the abscissa $x_1 = x_0 + h$, we will take point N_1 on the tangent having the same abscissa, but having the ordinate $y_1 = y_0 + h\, f(x_0, y_0)$.

We can, by the same method, determine a sequence of points $N_2(x_2, y_2)$, $N_3(x_3, y_3)$ ..., for which the ordinates will be determined from the ordinate and abscissa of the preceding point by means of the formula:

$$y_{j+1} = y_j + h f(x_j, y_j) \tag{2}$$

$$h = x_{j+1} - x_j$$

The error resulting from the procedure given by formula (2) appears for two distinct reasons; in the first place a systematic error due to the distance between points M_j and N_j, and in the second place the error due to

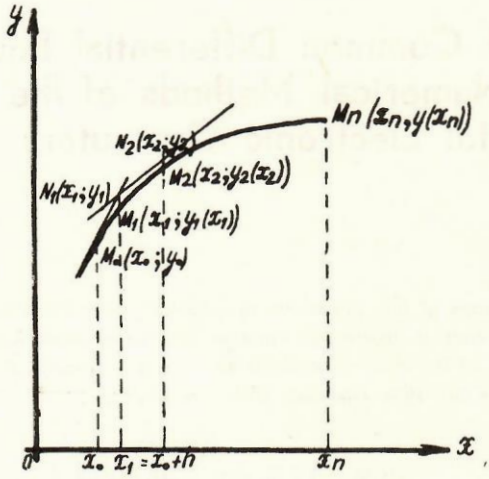


Fig. 1

rounding off when computing the function $f(x, y)$. By decreasing the step h, we obviously reduce the systematic error but in exchange we increase the rounding off error since the function $f(x, y)$ will then be computed more often.

We are thus led to make better approximations of the solution. We shall approximate the solution by parabola arcs which, with respect to the axes of co-ordinates considered, will be of the form $y = ax^2 + bx + c$. In this case two easily verified properties of the parabola are used (Fig. 2).

a) Given a parabola arc M_0M_1, the tangent at point N having as abscissa $\frac{x_0 + x_1}{2}$ is parallel to the chord M_0M_1.

b) The slope of the chord M_0M_1 is the arithmetical mean of the slopes of the tangents to the parabola at points M_0 and M_1. These properties lead to a new formula by which point M_1 on the solution curve of the differential equation (1) is better approximated.

Instead of point N having the abscissa $x_0 + h/2$, point N' of the same abscissa is determined by the method of the tangent. The slope of the tangent at N' is computed and the equation of the chord M_0M_1, parallel to this tangent, is written. We thus find formulae for the approximate determination of the ordinate of point M_1:

$$y_1 = y_0 + 1/2\,(K_1 + K_2)$$

where

$$K_1 = h\,f(x_0,\ y_0)$$

$$K_2 = h\,f(x_0 + h/2,\ y_0 + K_1)$$

This procedure can be repeated for the whole integration interval and the approximate value of the points on the solution curve can be computed by the formulae:

$$y_{j+1} = y_j + 1/2\ (K_1 + K_2) \qquad (3)$$

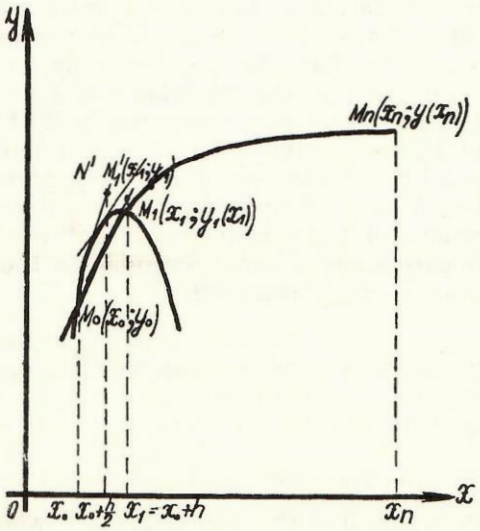


Fig. 2

where

$$K_1 = h\,f\,(x_j,\ y_j)$$

$$K_2 = h\,f\ (x_j + h/2,\ y_j + K_1)$$

Starting from the foregoing geometrical considerations there result, by generalization, the Runge-Kutta type formulae of numerical integration.

A Runge-Kutta type method consists in computing the coeficients $a_1, a_2, \ldots$; $b_1, b_2, \ldots$, as well as the numbers $c_1, c_2, \ldots$, so that the sequence of approximate values of the solution function $y(x)$ determined by the formulae given below be as close as possible to its exact values.

Setting:

$$K_1 = h\,f(x_0,\ y_0)$$
$$K_2 = h\,f(x_0 + a_1h,\ y_0 + b_1K_1)$$
$$K_3 = h\,f(x_0 + a_2h,\ y_0 + b_2K_2)$$

we obtain the formula of numerical integration:

$$y_{j+1} = y_j + c_1K_2 + c_2K_2 + \ldots \qquad (4)$$

.....

Considering the Taylor series development of the solution of equation (*1*) (the coefficients of this development can be determined by the derivation of equation (*1*) and the series development of the approximate solution given by formula (*4*), we can obtain the Runge-Kutta type formula of numerical integration for various approximations by identifying the coefficients of the two developments.

To obtain an approximation of the second order a formula depending on one parameter will be found. By choosing a proper value for this parameter we shall again find the formulae (*3*). Generally, in view of obtaining an approximation of the order n, the formulae will depend on $n - 1$ parameters. For every choice of these parameters we shall obtain a formula for the approximation of the respective order. The determination of formulae from which there should result the error made by applying such a method, for the numerical integration of an equation, has only been achieved roughly. In this sense, expressions of the remainder of the approximate series (*4*), have been found. The extensions of these formulae to the solving of differential equation systems is easily achieved.

2. THE RUNGE-KUTTA METHOD WITH APPROXIMATION OF THE IVth ORDER

Given the system:

$$y_i' = f_i(x,\ y_1, \ldots y_n) \qquad (i = 1,\ 2, \ldots, n) \qquad (5)$$

with the initial values $y_i(x_0) = y_i^0$, $i = 1,\ 2, \ldots n$, where the functions in the right hand member satisfy the conditions of existence and uniqueness of the solution for a domain in R^{n+1}, to integrate it by a method giving an approximation of the IVth order. For a given choice of the parameters, we use the classical formulae of this approximation:

$$y_i^{j+1} = y_i^j + \frac{1}{6}(K_1^i + 2K_2^i + 2K_3^i + K_4^i) \qquad (6)$$

where

y_i^j, $i = 1,\ 2, \ldots\ n$, represents the values of the solution functions for the step j;

y_j^{i+1}, $i = 1,\ 2,\ 3, \ldots,\ n$ represents the values of the solution functions for the step $j + 1$,

and

$$K_1^i = h\ f_i(x^j,\ y_1^j,\ y_2^j,\ \dots\ y_n^j)$$

$$K_2^i = h\ f_i(x^j + h/2,\ y_1^j + K_1^1/2,\ y_2^j + K_1^2/2,\ \dots\ ,\ y_n^j + K_1^n/2)$$

$$K_3^i = h\ f_i(x^j + h/2,\ y_1^j + K_2^1/2,\ y_2^j + K_2^2/2,\ \dots\ ,\ y_n^j + K_2^n/2)$$

$$K_4^i = h\ f_i(x^j + h,\ y_1^j + K_3^1,\ y_2^j + K_3^2,\ \dots\ ,\ y_n^j + K_3^n)$$

h being the step of integration in the required interval.

3. LOGICAL DIAGRAM OF THE COMPUTING PROGRAMME

In Fig. 3 a logical diagram is represented, according to which we can write a programme meant to integrate a system of differential equations of the form (*5*) by the formulae (*6*).

For the understanding of this diagram we give hereunder the sense of the notations used:

J — represents the total number of step units which complete the integration interval;

P — is the numbering cell of the step units covered;

j — represents the index of the current step unit;

i — represents the current index of the unknown equation and function in the system $(1 \leqslant i \leqslant n)$;

l — represents the lower index of the expressions K_l^i;

n — is the dimension of the system;

N — is the numbering cell of the system functions;

X and x — represent the addresses in which the value of the variable is placed for the step unit j;

h — is the integration step unit;

k_i — represents n consecutive working addresses;

K_l^i — represents 4 n consecutive working addresses in which will be found the values of the expressions in formulae (*6*), K_1^i, K_2^i, K_3^i and K_4^i $(i = 1, 2, \dots\ n)$ for the step unit j;

y_i — represents n consecutive working addresses in which will be laid the various forms of the arguments of the expressions K;

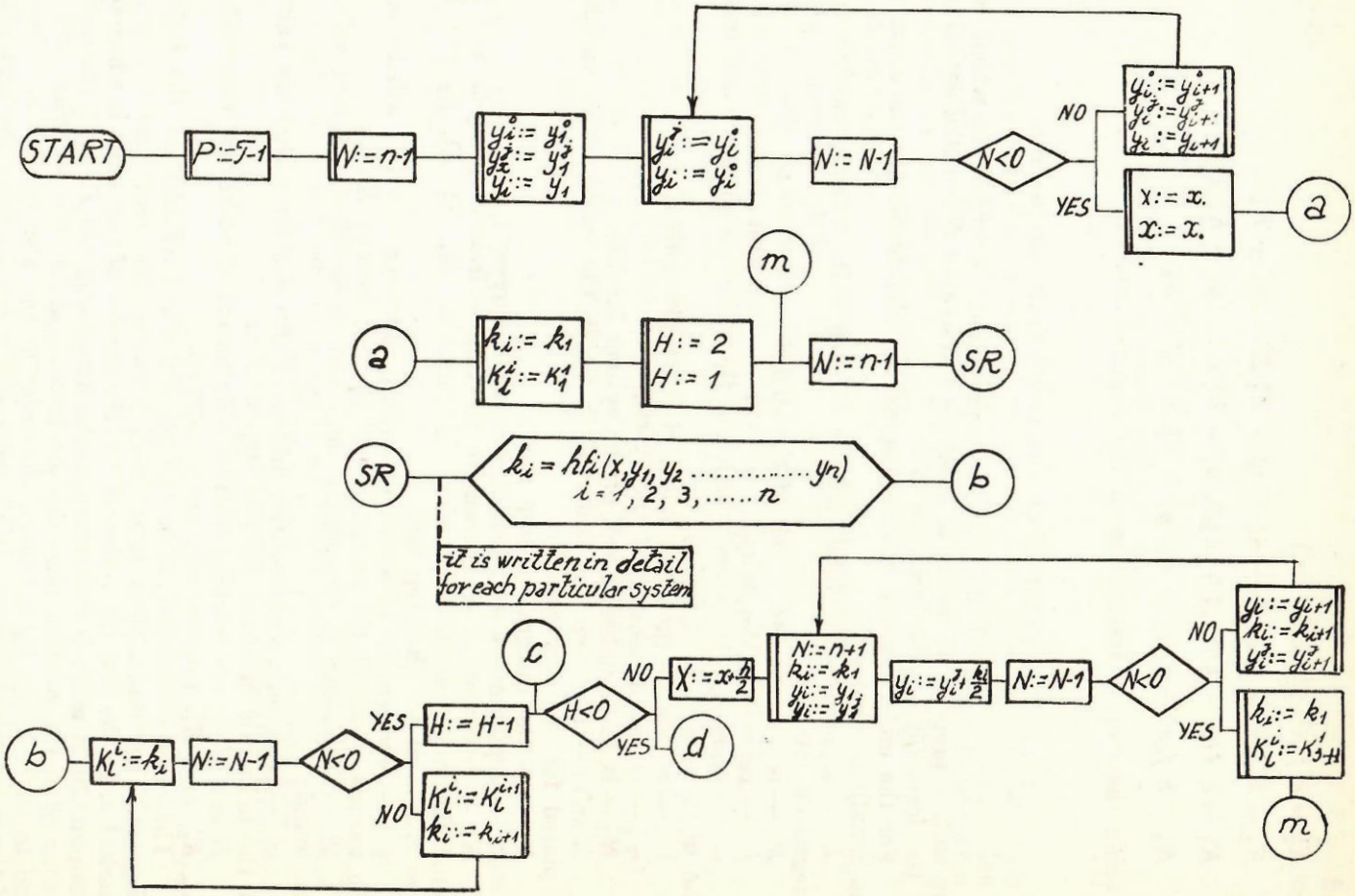
y_i^0 — represents n consecutive addresses in which the initial values are found;

y_i^j — represents n consecutive addresses in which are placed the values of the unknown functions for the step unit j;

H and H_1 — are working addresses by means of which the expressions K_2^i, K_3^i, and K_4^i, respectively, are computed.

The computation begins at the point marked *START* in the logical diagram. The computation is initiated between this point and the point marked *a*. In the first place the value of the number of step units in which the computation is made is introduced in the numbering cell P. Then the initial values y_i^0 of the solution functions are introduced in the working cells y_i^j and in the cells y_i, by a loop of the order n, and also the initial value x_0 of the independent variable, in the addresses x and X. Since the arguments

THE RUNGE-KUTTA METHOD OF THE IVth ORDER—THE LOGICAL DIAGRAM



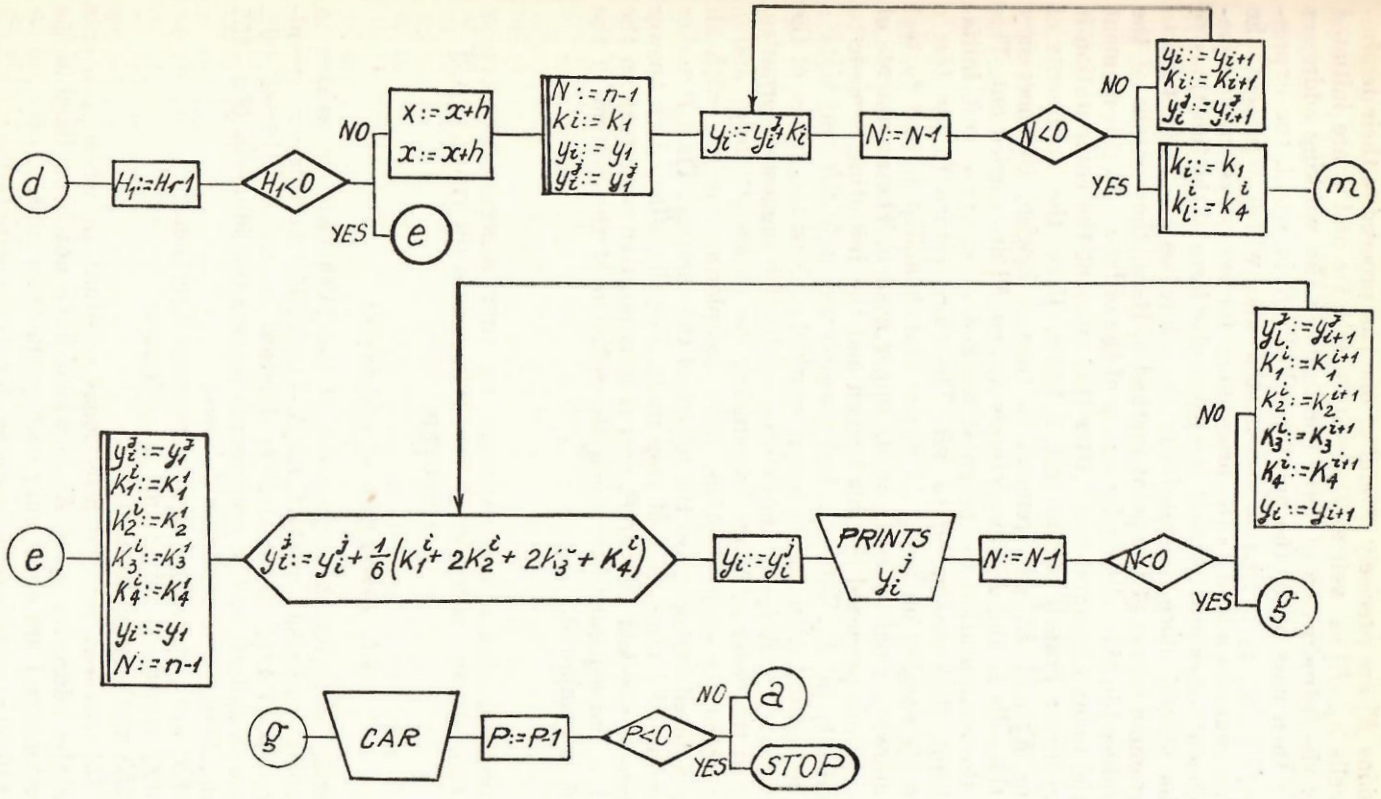

d
$H_1 := H_{r-1}$
$H_1 < 0$
NO
YES
$x := x+h$
$x := x+h$
e
$N := n-1$
$k_i := k_1$
$y_i := y_1$
$y_i^j := y_1^j$
$y_i := y_i^j + k_i$
$N := N-1$
$N < 0$
NO
YES
$y_i := y_{i+1}$
$k_i := k_{i+1}$
$y_i^j := y_{i+1}^j$
$k_i := k_1$
$k_i^i := k_4^i$
m
e
$y_i^j := y_i^j$
$K_1^i := K_1^1$
$K_2^i := K_2^1$
$K_3^i := K_3^1$
$K_4^i := K_4^1$
$y_i := y_1$
$N := n-1$
$y_i^j := y_i^j + \frac{1}{6}(K_1^i + 2K_2^i + 2K_3^i + K_4^i)$
$y_i := y_i^j$
PRINTS y_i^j
$N := N-1$
$N < 0$
NO
YES
$y_i^j := y_{i+1}^j$
$K_1^i := K_1^{i+1}$
$K_2^i := K_2^{i+1}$
$K_3^i := K_3^{i+1}$
$K_4^i := K_4^{i+1}$
$y_i := y_{i+1}$
g
g
CAR
$P := P-1$
$P < 0$
NO
YES
a
STOP

Fig. 3

of the expressions K_1^i are precisely x^j and y_i^j one can proceed to their determination. The cells k_1, K_1^1 as well as the tests marked c and d are initiated to this end by the introduction of some constants in the working addresses H and H_1. We then pass on to the point marked SR in which the expressions $k_i = h \times f_i(X, y_1, y_2, \ldots, y_n)$ are estimated and which represent the sub-routine that must be added to the programme for every system or equation in part, since it takes into account the particular form of the functions f_i. The sub-routine should place the results in the addresses k_i. The return to the main programme occurs at the point marked b. Here, the content of the cells k_i is introduced in the cells K_l^i by a loop of the order n. At this moment the value of the index l is equal to 1. After this moment the computation is sent along the upper branch of the test c twice. Here, the arguments of the expressions K_2^i and K_3^i are prepared in turn. Likewise, the necessary initiating for the cells k_1 and K_2^1, respectively k_1 and K_3^1 are carried out. This branch sends the computation to the point marked m where a test initiating is carried out, then passing on to SR. The third return to the test c does not allow its passing on along its upper branch and it is sent to test d. This test allows a single passing on its upper branch. The arguments of the expression K_4^i are prepared on this branch and the initiating is made to retain the results k_i in K_4^i. After a last passing through SR the tests c and d send the computation to the point marked e. Now the values of the functions for the step unit $j + 1$ are determined by the suggested formulae, through a loop of the order n. After introducing the values y^{j+1} in y_i^j and y_i the result is printed for each function. After completing one line with all the n values computed one operates the return of the carriage. Then P makes the test of running the number J of step units. For the affirmative answer the computation is checked at $STOP$, and if it is positive we return to the point marked a consequently to which the solution of the system in the next step unit is computed.

4. INTEGRATION PROGRAMME OF SYSTEMS OF DIFFERENTIAL EQUATIONS BY THE RUNGE-KUTTA METHOD OF THE IVth ORDER ON THE C.I.F.A.-4 COMPUTER

4.1. *Description of programme*

The general Runge-Kutta programme of the IVth order is written in Table 1 in absolute addresses, in the C.I.F.A.-4 code. The starting corresponding to the point $START$ in the logical diagram occurs at address 001. Stopping occurs at address 0. The programme covers the addresses 000—155 (110 addresses), being distributed as follows:

a) 000—120 are covered by the programme directions.

b) 121—123 constants for changing addresses.

c) 147—155 working addresses.

d) 125—141 covered by the directional constant in whose contents are comprised the addresses y_i, y_i^j, K_l^i, k_i which have not been fixed in the general programme and are chosen for each problem in part.

e) 142—146 constants are introduced for the computation.

Table 1

IVth ORDER RUNGE-KUTTA METHOD
PROGRAMME FOR ELECTRONIC COMPUTER C.I.F.A. — 4

Address	F_1	N_1	F_2	N_2	Remarks	Address	F_1	N_1	F_2	N_2	Remarks
000	00	000	00	000	STOP	67	01	023	03	125	
1	01	126	11	151	START	70	02	123	11	023	
2	01	125	11	150	Initiation	71	10	021	00	000	
3	01	127	11	005	j = 0	72	01	125	11	150	
4	01	130	11	006	i = 1,2,...,n	73	01	136	11	077	Computation
5	00	000	00	000		74	01	137	11	101	of values (e)
6	00	000	00	000		75	01	140	11	103	
7	03	122	11	150		76	01	141	11	104	
10	07	014	01	005		77	00	000	00	000	y_i^{i+1} $i = 1,2,...n$
11	02	121	11	005	Address	100	04	142	11	147	
12	01	006	02	122	change	101	00	000	00	000	
13	11	006	10	005		102	04	143	02	147	
14	01	146	11	153		103	00	000	00	000	
15	11	152	13	000		104	00	000	00	000	
16	01	131	11	023	*(a)*	105	01	150	03	122	
17	01	124	11	154		106	11	150	07	116	
20	01	123	11	155		107	01	077	02	121	
21	01	125	11	150		110	11	077	01	101	
22	10	SR	00	000	Sends to	111	02	121	11	101	
23	00	000	00	000	$\overline{SR}$ (b)	112	01	103	02	121	
24	01	150	03	122		113	11	103	01	104	
25	11	150	07	030		114	02	122	11	104	
26	01	023	02	121		115	10	077	00	000	
27	11	023	10	023		116	17	000	01	151	Test for
30	01	154	03	123	Test (c)	117	03	123	11	151	bottom
31	11	154	07	051		120	07	000	10	016	number (g)
32	01	152	02	145	$X := x^j + h/2$	121	00	001	00	001	Constants for
33	11	153	01	023		122	00	001	00	000	instruction
34	03	125	02	123		123	00	000	00	001	change
35	11	023	13	000		124	00	000	00	002	
36	01	123	11	150		125	00	00(n-1)	00	000	
37	01	132	11	041		126	00	000	00	00(j-1)	
40	01	133	11	042		127	01	y_1^0	11	y_i^j	
41	00	000	00	000		130	11	y_i	01	150	
42	00	000	00	000		131	01	k_i	11	k_1^1	
43	01	150	03	122		132	01	k_1	04	144	Instructing
44	11	150	07	021		133	02	y_i^j	11	y_1	constants
45	01	041	02	122		134	01	y_i^j	02	k_1	to be
46	11	041	01	042		135	11	y_1	01	150	completed
47	02	121	11	042		136	01	k_2^1	02	k_3^1	for each
50	10	041	00	000	Test (d)	137	01	k_1^1	02	k_4^1	problem
51	01	155	03	123		140	02	y_i^j	11	y_i^j	in part.
52	11	155	07	072	$X := x^j + h$	141	11	y_1	16	000	
53	01	153	02	145		142	+·333	333	333		
54	11	154	11	152		143	+·166	666	666		
55	01	125	11	150		144	+·5				
56	01	134	11	060		145	$h/2$				Working address
57	01	135	11	061		146	x_0				
60	00	000	00	000		147	S				
61	00	000	00	000		150	N				
62	03	122	11	150		151	P				
63	07	067	01	060		152	x				
64	02	121	11	060		153	X				
65	01	061	02	122		154	H				test (c)
66	11	061	10	060		155	H_1				test (d)

4.2. *Use of programme*

The memory is free between the addresses 155—777. According to the dimension n of the system the following working addresses will be chosen:

$y_1^0\ y_2^0 \ldots y_n^0$ consecutive addresses in which the initial values of the solution functions are introduced.

$y_1^j\ y_2^j \ldots y_n^j$ consecutive addresses in which the values of the functions at the step unit j will be found.

$y_1\ y_2 \ldots y_n$ consecutive addresses in which the arguments of the functions f_i are found.

$k_1\ k_2 \ldots k_n$ consecutive cells in which the results of the equal sub-routine for every i with $hf_i(X, y_1 \ldots y_n)$ will be found.

$K_1^1\ K_1^2 \ldots K_1^n$
$K_2^1\ K_2^2 \ldots K_2^n$ 4 n consecutive addresses with the specification that K_2^1
$K_3^1\ K_3^2 \ldots K_3^n$ follows K_1^n and K_3^1 follows K_2^n and K_4^1 follows K_3^n.
$K_4^1\ K_4^2 \ldots K_4^n$

After fixing these addresses the contents of the cells 125—142 is completed by writing instead of the above symbols the addresses chosen for the particular problem which we are solving. It should be noted that in address 125 the number $n - 1$ should be written in the middle position in octal, in cell 126 the number $J - 1$ on the last positions also in octal and in cell 145 the decimal value h/2. The writing of the sub-routine is made in the memory, which has remained available, of the computer, and it describes the computation of the expressions $k_1 = hf_i\ (X,\ y_1,\ y_2,\ \ldots,\ y_n)$, the arguments being placed in $y_1, y_2 \ldots y_n$, and X in 153. The sending to $\overline{SR}$ is made in the main programme at the address 022 by the direction 10 N 00 000, N representing the input address in $\overline{SR}$. The last direction of the sub-routine should achieve the return to the main programme and it will be 10 023.

4.3. *Numerical integration method of the Runge-Kutta type with a number of approximations lower than 4*

From the programme expounded above, by small changes of the directions, one can obtain programmes which follow Runge-Kutta formulae with a small number of approximations. In some cases it is more convenient to use simpler methods, the results remaining unaffected by the simplicity of the formulae. On the other hand, the solving process will be shorter.

4.3.1. *Tangent method (Euler).* The following formulae are used to integrate system (*1*):

$$y_i^{j+1} = y_i^j + h\, f_i(x^j,\ y_1^j, \ldots, y_n^j) \tag{7}$$

To integrate the system by means of the formulae (7), the changes indicated in Table 3 are made to Table 1. In the logical diagram this amounts to a single passing through the mark SR. The tests marked c and d will cause the direct passing on to mark e, where the expression in the block of procedure will change according to (7).

CHECKING EXAMPLE (SUB-ROUTINE) C.I.F.A — 4

Non-codified programme					Remarks
Address	F_1	N_1	F_2	N_2	
200	01	y_2	04	h	$\overline{SR}$
201	11	k_1	01	y_3	
202	04	h	11	k_2	
203	01	y_2	02	y_3	
204	04	(−h/A)	11	k_3	
205	01	y_1	04	(−γh/A)	
206	02	k_3	02	(0.1γh/A)	
207	11	$k3_3$	10	023	Return
210	h				to main
211	−h/A				programme
212	0.1γh/A				
213	−γh/A				
214	y_1^0				
215	y_2^0				
216	y_3^0				
252	y_1				
253	y_2				
254	y_3				
255	k_1				
256	k_2				
257	k_3				
260	y_1^j				
261	y_2^j				
262	y_3^j				
263	k_1^1				
264	k_1^2				
265	k_1^3				
266	k_2^1				
267	k_2^2				
270	k_2^3				
271	k_3^1				
272	k_3^2				
273	k_3^3				
274	k_4^1				
275	k_4^2				
276	k_4^3				

Codified programme				
Address	F_1	N_1	F_2	N_2
200	01	253	04	210
201	11	255	01	254
202	04	210	11	256
203	01	253	02	254
204	04	211	11	257
205	01	252	04	213
206	02	257	02	212
207	11	257	10	023
210	+.1			
211	−.72			
212	.056952			
213	−.056952			
214	0			
215	0			
216	0			

Completed instructing constants of main programme

Address	F_1	N_1	F_2	N_2
125	00	002	00	000
126	00	000	00	143
127	01	214	11	260
130	11	252	01	150
131	01	255	11	263
132	01	255	04	144
133	02	260	11	252
134	01	260	02	255
135	11	252	01	150
136	01	266	02	271
137	01	263	02	274
140	02	260	11	260
141	11	252	16	000
022	10	200	13	000
145	+ 0.05			
146	0			

Table 3

THE TANGENT METHOD

Changes in main programme				
Address	F_1	N_1	F_2	N_2
017	01	000	11	154
020	01	000	11	155
100	13	000	13	000
102	13	000	13	000
110	02	000	11	077
111	02	102	11	101
136	13	000	13	000
137	01	K_1^1	13	000

4.3.2. *The Runge-Kutta method with two approximations.* To integrate system (*1*) in this case, the following formulae are used:

$$y_i^{j+1} = y_i^j + \frac{1}{2}(K_1^i + K_2^i)$$

where

$$K_1^i = hf_i(x^j,\ y_1^j \ldots y_n^j) \tag{8}$$

$$K_2^i = hf_i(x^j + h,\ y_1^j + K_1^1,\ y_2^j + K_1^2, \ldots, y_n^j + K_1^n$$

The programme described in Table 1 will carry out the computation according to the formulae (*8*) provided the content of some addresses be changed as in Table 4. In the logical diagram in Fig. 3 this amounts to renounce to the test marked *c*. After passing twice through *SR* it reaches the point marked *e* where the formulae in the block of procedure will change according to (*8*).

Table 4

THE TWO-APPROXIMATIONS METHOD

Changes in main programme				
Address	F_1	N_1	F_2	N_2
017	01	000	11	150
100	13	000	13	000
102	04	144	13	000
107	001	077	02	000
136	13	000	13	000
137	01	K_1^1	02	K_2^1

Remark 1. — From the general programme one can also obtain, by several changes, a programme which applies a formula with three approximations.

Remark 2. — The control example in paragraph 5.1 was integrated both by the method with four approximations and that with two approximations. The results obtained by both methods were very close in value.

5. PROBLEMS SOLVED BY MEANS OF THE RUNGE-KUTTA PROGRAMME WITH FOUR APPROXIMATIONS

5.1. *Example 1 (Control example)*

The following equation is considered:

$$Az'''(t) + z''(t) + 2'(t) + \gamma z(t) = 0.1\gamma \qquad (9)$$

where

$$A = \frac{5}{36}, \quad \gamma = 0.791 \qquad (10)$$

with the initial conditions $z(0) = z'(0) = z''(0) = 0$, and the integration was made in the interval (0,10) with the integration step $h = 0.1$. The solution of equation (*10*) in the given conditions represents from the technical standpoint the answer to a follow-up system of the third order at a step input. To determine this solution we turn equation (*10*) into a system of three differential equations of the first order, making the following substitutions:

$$z = y_1$$

$$z' = y_2 = y_1'$$

$$z'' = y_3 = y_2'$$

$$z''' = y_3'$$

Hence, the following system is obtained:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ y_3' &= \frac{1}{A}\,(0.1\ \gamma - \gamma y_1 - y_2 - y_3) \end{aligned} \qquad (11)$$

the initial conditions being

$$y_1(0) = y_2(0) = y_3(0) = 0$$

The routine for computing the expressions $k_i = hf_i(X, y_1, y_2, y_3)$, $i = 1, 2, 3$ which are attached to the programme in Table 1, is written both coded and uncoded in Table 2. The uncoded version is given to make the manner in which the working cells were chosen stand out, as well as the way in which the directional constants of the main programme in addresses 127—141 were completed. The accuracy of the results was checked by comparing the results obtained on the digital computer with the values of the solution function, computed manually.

5.2. *Example 2. Determining the distribution of pulse voltages along a high-voltage divider.*

The following system of equations was established in view of solving this problem:

$$u_1(t) + R_i(t) + 2\frac{di}{dt} = u_0$$

$$\frac{u_1(t)}{R_1} + c_1\frac{du_1}{dt} - \frac{u_2(t)}{R_2} - c_1\frac{du_2}{dt} = 0$$

$$\frac{u_1(t)}{R_1} + c_1\frac{du_1}{dt} - u_2\left(\frac{1}{R_1} + \frac{1}{R_2}\right) - (c_1 + c_2)\frac{du_2}{dt} + \frac{u_3}{R_3} + c_2\frac{du_3}{dt} = 0 \qquad (12)$$

$$\frac{u_2(t)}{R_2} + c_2\frac{du_2}{dt} - u_3(t)\left(\frac{1}{R_2} + \frac{1}{R_3}\right) - (c_2 + c_3)\frac{du_3}{dt} = 0$$

with the initial conditions $u_1(0) = u_2(0) = u_3(0) = i(0) = 0$ where the function u_0 represents the pulse voltage and was considered to be of the form

$$u_0(t) = \begin{cases} \dfrac{1}{t_f}\,t & t \leqslant t_f \\ 1 & t > t_f \end{cases} \qquad (13)$$

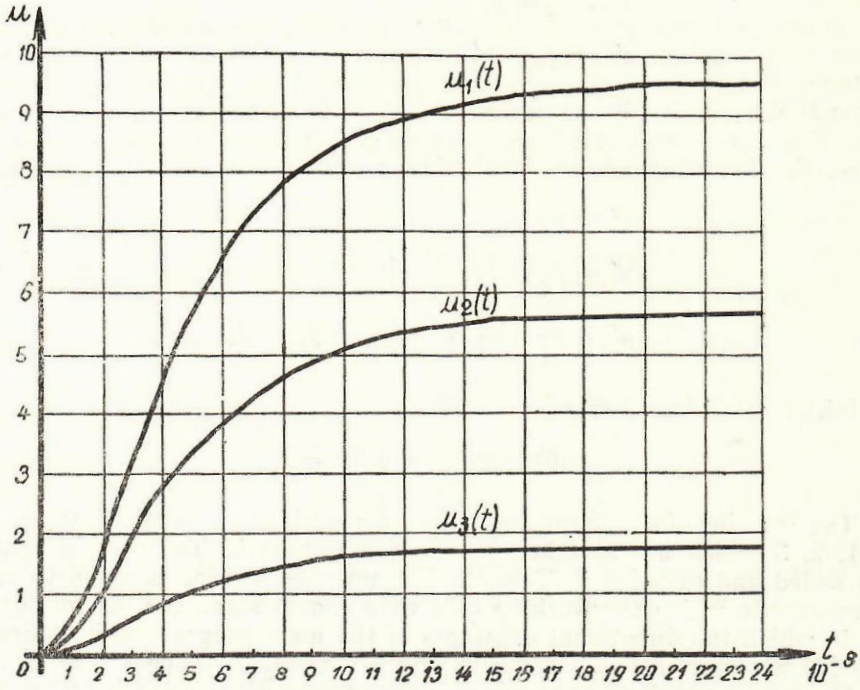
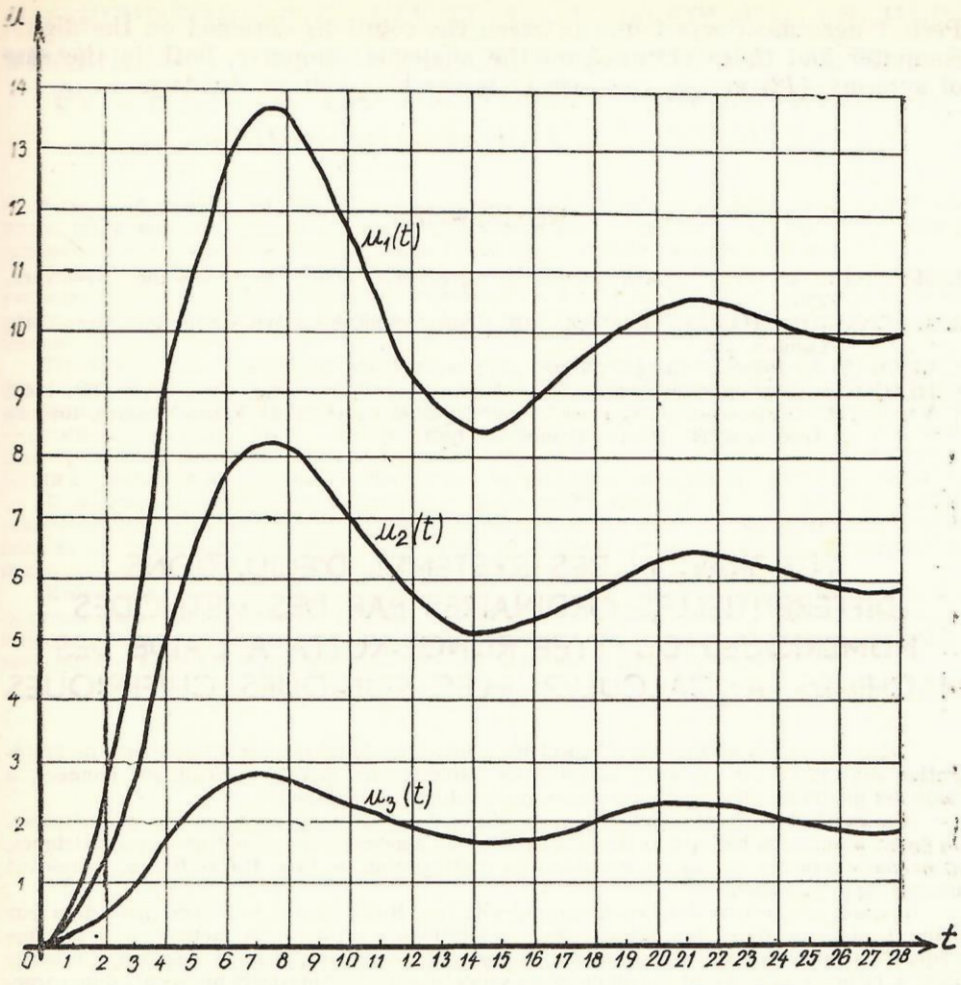


Fig. 4



Fg. 5 $R = 120$

and the functions $u_1(t)$, $u_2(t)$ and $u_3(t)$ represent the pulse voltage in the divider considered. R_j, c_j and R are constants in the divider diagram. In view of integrating system (*12*) the derivatives of the unknown functions are expressed, the system being written in the canonical form, so as to allow the Runge-Kutta type integration formulae to be applied.

System (*12*) was integrated for two values of the resistence R. The accuracy of the results was checked by comparing the solution curves obtained on the digital computer to those obtained by solving the problem on a *MEDA* analogue computer.*

The solutions obtained in these two cases are shown in Fig. 4 and 5.

* The solution of system (*12*) on an analogue computer was carried out by electronical engineer Sanda Lehaci from the Computation Laboratory of the I.C.P.E.

Perfect agreement was found between the solutions obtained on the digital computer and those obtained on the analogue computer, both in the case of systems (*12*) and in the case of larger high-voltage dividers.

REFERENCES

1. Mc. CRACKEN, D.D., *Digital Computer Programming*, John Wiley Sons Inc., New-York, 1960.
2. Mc. CRACKEN, D.D., *A Guide to ALGOL Programming*, John Wiley Sons Inc., New-York, London, 1963.
3. LEGRAS, J., *Précis d'Analyse Numérique*, Dunod, Paris, 1964.
4. HERRIOT, J., *Methods of Mathematical Analysis and Computation*, Mc. Grow Hill, 1963.
5. VAIDA, D., *Utilizări ale calculatoarelor electrice de la Institutul de Fizică Atomică*, Editura Academiei R. P. R., Bucureşti, 1961.

INTEGRATION DES SYSTEMES D'EQUATIONS DIFFERENTIELLES ORDINAIRES PAR DES METHODES NUMERIQUES DU TYPE RUNGE-KUTTA A L'AIDE DES MACHINES A CALCULER ELECTRONIQUES CHIFFRIQUES

Dans cet article on présente l'aspect mathématique de la solution du problème de l'intégration numérique des systèmes d'équations différentielles dans des conditions données, à l'aide des méthodes du type Runge-Kutta (méthodes des pas libres).

Les méthodes de solutionnement sont d'abord exposées au point de vue géométrique. On donne ensuite des indications sur la généralisation algébrique de ces méthodes géométriques, au moyen desquelles on obtient des procédés d'intégration du type Runge-Kutta, entraînant diverses approximations.

Prenant comme base de calcul les formules du type Runge-Kutta du IV-ème ordre dans leur forme classique, on décrit le schéma logique de solutionnement par cette méthode d'un système d'équations différentielles. On donne un exemple de programme pour la machine à calculer CIFA-4, ce programme ayant été appliqué au solutionnement de plusieurs problèmes d'un caractère technique. On montre que ce programme peut être transformé en programme de la méthode Runge-Kutta à une ou deux approximations par la modification de certaines instructions.

Dans la dernière partie de l'article on décrit deux problèmes présentant un intérêt technique, qui ont été résolus à l'aide de ce programme. Le premier exemple représente l'équation de la réponse d'un système de poursuite du III-ème ordre pour une entrée droite et constitue même l'exemple de contrôle du programme, étant donné que les vérifications ont été exécutées par calcul manuel.

Le second exemple représente la détermination de la répartition des tensions de pulsation dans un diviseur de haute tension. Dans ce second exemple les courbes-solution sont représentées sous forme de diagrammes étant donné que les systèmes ont également été intégrés sur un calculateur analogue. Les résultats obtenus par les deux méthodes de solution ont coïncidé.

ИНТЕГРИРОВАНИЕ СИСТЕМ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПРИ ПОСРЕДСТВЕ ЦИФРОВЫХ МЕТОДОВ РУНГЕ-КУТТА НА ЦИФРОВЫХ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

В труде, в первую очередь рассматривается математическая сторона разрешения вопроса цифровой интеграции систем дифференциальных уравнений при заранее заданных условиях и при посредстве методов типа Рунге-Кутта (методы свободного шага).

Методы решения излагаются сначала с геометрической точки зрения. Затем даются указания в связи с алгебраическим обобщением этих геометрических методов, при помощи которых получаются способы интегрирования типа Рунге-Кутта с различными приближениями.

Приняв за основу исчисления формулы типа Рунге-Кутта IV разряда в их классической форме, описывается логическая схема решения системы дифференциальных уравнений при помощи этого метода. Приводится пример программы для вычислительной машины CIFA-4, — программы, которая была применена для решения ряда задач технического характера. Далее указывается, что эта программа может быть превращена в программы метода Рунге-Кутта с одним или двумя приближениями при посредстве изменения некоторых инструкций.

В последнем разделе приводятся две, представляющие технический интерес задачи, решенные при помощи этой программы. Первый пример представляет уравнение ответа следящей системы III разряда на вход-ступень и даже является примером контроля программы, поскольку проверка была произведена при помощи вычисления вручную.

Второй пример представляет определение распределения импульсного напряжения в делителе высокого напряжения. В этом — втором — примере кривые решения представлены в форме графиков, так как системы были интегрированны и на аналогичной вычислительной машине. Результаты, полученные с помощью обоих методов, совпали.