PRACE Instytutu Maszyn Matematycznych PAN

Praca A 2 (19)

ANALYSIS OF NONLINEAR CIRCUITS ON DIGITAL COMPUTER

by Ryszard ŁUKASZEWICZ

P R A C E

Instytutu Maszyn Matematycznych

P o l s k i e j   A k a d e m i i   N a u k

T. II   Praca A 2/19/

ANALYSIS OF NONLINEAR CIRCUITS ON DIGITAL COMPUTER

by Ryszard ŁUKASZEWICZ

Warszawa 1963

Adres redakcji: Warszawa, ul.Koszykowa 79, tel. 8-37-29

Instytut Maszyn Matematycznych PAN
Praca A 2/19/

621.14-523.8:518.6:621-526

ANALYSIS OF NONLINEAR CIRCUITS ON DIGITAL COMPUTER

by Ryszard ŁUKASZEWICZ

Received November 24th, 1962

With reference to the direct method of circuit analysis on digital computer [1] the paper considers the way of using subroutines written in the SAKO or ALGOL autocodes. The subroutines describe the relations appearing in nonlinear elements of circuits. Nonlinear subroutines for typical nonlinearities occurring in servomechanism sets are discussed.

1. Introduction

In the paper 'Direct Method of Circuit Analysis on Digital Computers' [1] the method was presented of formulating first-order differential equations describing the operation of the analyzed circuit. The form of these equations is directly adapted to solve them on a digital computer. An example of the program realizing servomechanism computations is also presented.

As emphasized in the above-mentioned paper, one of the basic advantages of the described method is the convenience it gives to consider nonlinearities occurring in the circuit.

The purpose of the present paper is to make constructors acquainted with the method of preparing and using subroutines for computing nonlinear circuits. The method will be illustrated by examples of servomechanism analyses. The given procedure, and the descriptions of subroutine for the basic types of nonlinearities hold true for other circuits as well.

2. Analysis of a nonlinear servomechanism

Using the direct method of circuit analysis the consideration of nonlinearities of servomechanism elements is, in principle, reduced to the preparation of subroutines /further called nonlinear subroutines/ that describe relations determined by a given nonlinear element. The names of these subroutines are used when formulating differential equations describing the servomechanism operation.

For example, let us consider the system, the diagram of which [1], [3] is shown in Fig. 1. /The relay system of a similar type was computed by R. K. Adams [4] on a digital computer by means of a different method/.

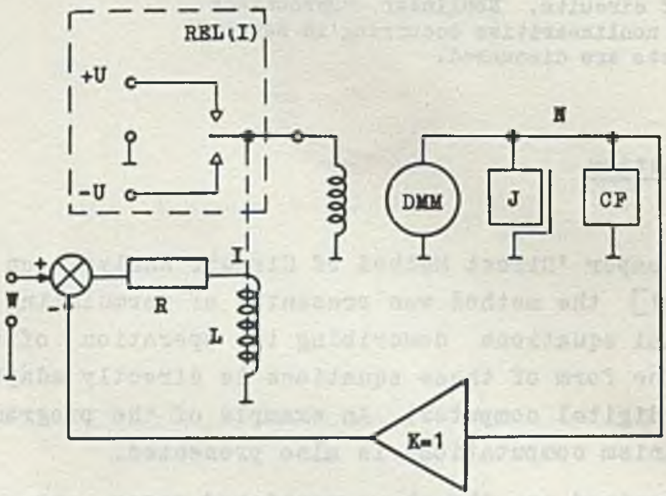


Fig. 1. Nonlinear Servomechanism
U = REL(I) - relay output voltage, N - angular speed, DMM(U, N) - driving moment of the motor, CF(DMM, N) - Coulomb friction.

According to the direct method of circuit analysis differential equations are the following:

$$\mathring{T} = 1$$
$$\mathring{I} = (1/L)\cdot(W - N - R\cdot T) \qquad /1/$$
$$\mathring{N} = (1/J)\cdot(DMM(U, N) - CF(DMM(U, N), N))$$

Three types of nonlinearities occurring in a servomechanism will be computed by means of the following subroutines:

1. Relay - REL(I, YR, YRO, HI), the output voltage of which depends upon current I.*)
2. Driving moment of the motor - DMM(U, N, *A) dependent on the controlling voltage U and on the angular speed of the motor N.**)
3. Coulomb friction CF(DMM(U, N), N, B) depending upon the driving moment of the motor DMM(U, N), the rotation speed N, and the parameter B determining the magnitude of the Coulomb friction.

The characteristics of these nonlinearities and the way of writing appropriate subroutines will be discussed later. Let us now write a program for computing the operation of the servomechanism considered.

Introducing uniform denotations of variables, and taking into account the relation U = REL(I) we obtain

new denotations of variables	differential equations
T = Y(0)	$\dot{Y}(0) = 1$
I = Y(1)	$\dot{Y}(1) = (1/L)\cdot(W - Y(2) - R\cdot Y(1))$
N = Y(2)	$\dot{Y}(2) = (1/J)\cdot(DMM(REL(Y(1)), Y(2)) + - CF(DMM(REL(Y(1)), Y(2)), Y(2))$ /2/

If the integration step value for solving the above equations with the required accuracy by the Runge Kutta method is assumed to be H, the greatest number value appearing in the course of computation is less than 10^3 and the speed N in the interval $0 \leq T \leq 100H$ is to be traced, then, the SAKO and ALGOL***) program realizing the required computations will be the following:

---

* The meaning of arguments YR, YRO, HI is explained on page 20 together with the subroutine description.

** The meaning of arguments *A is explained on page 18 in section containing the subroutine description.

*** The ALGOL programs are given in addition for readers using ALGOL and not being familiar with SAKO.

<u>In SAKO</u>

```
CHAPTER: 0

SET DECIMAL SCALE: 3
ARRAY(0):W,H,R,L,J,B,U,UO,HI
ARRAY(2):A,Y
READ IN DECIMAL:W,H,R,L,J,B,U,UO,HI,*A,*Y
JUMP TO CHAPTER: 1

CHAPTER: 1
INTEGERS:I,N,C,NK,Z
ARRAY(0):W,H,R,L,J,B,U,UO,HI
ARRAY(2):A,Y,X,M,S
SUBSTITUTE:F(.,W,R,L,J)
SUBSTITUTE:REL(.,U,UO,HI)
SUBSTITUTE:DMM(.,.,*A)
SUBSTITUTE:CF(.,.,B)
*1)PRINT(5.3):Y(0),Y(1),Y(2)
NEW LINE
(*Y)=RK(*X,*S,*M,F(),2,H)
REPEAT FROM 1:I=0(1)100
STOP 2
2)JUMP TO CHAPTER: 0

SUBROUTINE:(*M)=F(*Y,W,R,L,J)
M(0)=1
M(1)=(1/L)×(W-Y(2)-R×Y(1))
M(2)=(1/J)×(DMM(REL(Y(1)),Y(2))-CF(DMM(REL(Y(1)),Y(2)),Y(2)))
RETURN

SUBROUTINE:REL(I,YR,YRO,HI)
        ..........
        ..........
        ..........

SUBROUTINE:DMM(U,N,*A)
        ..........
        ..........
        ..........

SUBROUTINE:CF(M,N,B)
        ..........
        ..........
        ..........

SUBROUTINE:(*Y)=RK(*X,*S,*M,F(),N,H)
        ..........
        ..........
        ..........
```

in ALGOL:

```
begin
procedure  REL(I, YR, YRO, HI);
     ..........
     ..........
     ..........
procedure  DMM(U, N, A);
     ..........
     ..........
     ..........
procedure  CF(M, N, B);
     ..........
     ..........
     ..........
procedure  RK(Y,X,S,F,N,H);
     ..........
     ..........
     ..........
procedure  F(Y, W, R, L, J)  results: (M);  array  Y, M;
begin      M[0]:= 1 ;
           M[1]:= (1/L)×(W-Y[2]-R×Y[1]);
           M[2]:= (1/J)×DMM(REL(Y[1], U, UO, HI), Y[2], A)-
                 CF(DMM(REL(Y[1], U, UO, HI), Y[2], A)
                                                     end F
procedure  read (a) ;
comment:  a - list of read variables;
     ..........
     ..........
     ..........
procedure  print (a, b, c);
comment:  a  and  b - number of places before and after decimal
                      point
                  c - list of printed variables;
     ..........
     ..........
     ..........
```

```
begin
integer i;  array  Y, X, M, S, A[0:2];
real  W, H, R, L, J, B, U, UO, HI;
read  (W, H, R, L, J, B, U, UO, HI, A, Y);
for  i:= 0  step 1  until 100  do
begin  RK(Y, X, M, S, F(Y, W, R, L, J), 2, H);
       print (5, 3, Y)
                                            end end end of program
```

The program written in SAKO consists of the Main Program /chapter 0 and chapter 1 for the first declaration SUBROUTINE/ and of five subroutines.

The Main Program in the given example fulfills the role of a routine program only, i.e. it reads the numerical data into the machine, substitutes them to the corresponding subroutines as arguments; it calls in the subroutine Y = RK() 101 times, and prints the results.*) The computations themselves are realized by means of subroutines. The subroutine (*Y) = RK(*X, *S, *M,F(),N,H)**) realizing one integration step by the Runge Kutta method [7] then uses the subroutine (*M) = F(*Y, W, R, L, I)***) which computes the right sides of differential equations.**) In turn, this subroutine uses subroutines REL(I,YR,YRO,HI), DMM(U, N, *A), CF(M, N, B)****) computing the behaviour of nonlinear elements of the circuit.

---

* More detailed description of the Main Program is given in the Appendix.

** This is a standard subroutine from the subroutine library. Its arguments are: *Y - initial values of differential equation variables, *X, *M, *S - reservation of places for operational variables, F() - name of the subroutine computing the right sides, N - number of differential equations, H - magnitude of the integration step.

*** Its arguments are: *Y - certain indirect values of circuit variables computed and substituted by the subroutine RK(), W - input signal value, R, L, I - values of the corresponding circuit parameters.

**** The content of these subroutines and the meaning of their arguments are given further.

In a well equipped computation centre dealing with construction, there usually are subroutines ready to describe a number of typical circuit nonlinearities, as well as subroutines solving differential equations. From the above example it is evident that, while programming circuit computations, the constructor's work is reduced to write the main program, the right side subroutine and to add appropriate standard subroutines.

3. Typical Nonlinear Subroutines

Most common nonlinearities in servomechanisms are: bend of the outline of the characteristic, saturation, dead zone, motor and relay characteristics*. Examples of appropriate subroutines are given below:

3.1. Bend of the outline of the characteristic /Fig. 2/

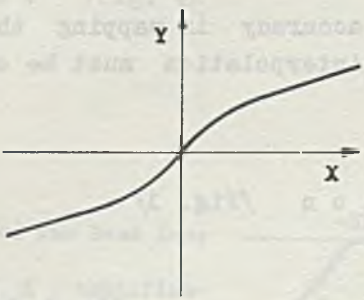


Fig. 2. Bend of the outline of the characteristic

The characteristic shown in Fig. 2 is symmetric to the centre of the coordinate set. Using parabolic interpolation for describing its outline above the axis X we obtain the expression

---

* The consideration of the backlash is a more complicated problem and it will be the subject of another paper.

$$Y = A \cdot X^2 + B \cdot X + C \qquad /3/$$

where A, B and C are interpolation coefficients.

The subroutine describing the bend of the outline of the characteristic will be the following:

In SAKO*)

```
SUBROUTINE : Y(X, A, B, C)
Y = SGN(A × X *2 + B × ABS(X) + C, X)
RETURN
```

Note: The values of arguments A, B and C in the described subroutine are substituted either directly by the operation formula calling the subroutine in or by means of the statement SUBSTITUTE: Y(·, A, B, C) /an example of using the statement SUBSTITUTE is to be found in the main program on page 6/.

In ALGOL*)

```
procedure Y(X, A, B, C); value X; real X;
begin Y := sgn(A × X ↑ 2 + B × abs(X) + C, X)
                                        end Y
```

In case greater accuracy in mapping the characteristic is required the applied interpolation must be of a correspondingly higher order.

3.2. S a t u r a t i o n /Fig. 3/

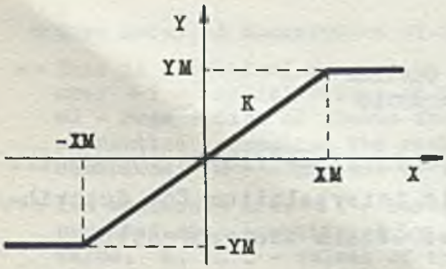


Fig. 3. Saturation Characteristic

|YM| - absolutely maximal value attainable by the output signal Y,
K - amplification of the input signal on the segment $-XM < X < XM$.

* In this subroutine, as well as in further nonlinear subroutines described in this paper, one of the SAKO functions is used, i.e. SGN(X, Y) and the function sgn(X, Y) is defined in ALGOL. In classical mathematics both of them denote the value $|X| \cdot \text{sign}\, Y$. As an example, the value $|5| \cdot \text{sign}\, -2 = -5$ corresponds to SGN(5, -2).

The appropriate subroutine following directly from Fig. 3 will be:

In SAKO

```
SUBROUTINE : SATURATION (X, XM, K)
IF ABS(X) > XM : NEXT, ELSE 1
X = SGN(XM, X)
1) SATURATION() = K × X
RETURN
```

In ALGOL

```
procedure SATURATION (X, XM, K); value X; real X;
begin
if abs(X) > XM then X := sgn(XM, X); SATURATION := K × X
                                          end SATURATION
```

Numerical values of arguments XM, K are substituted as indicated in the Note to the subroutine describing the bend of the outline of the characteristic.

3.3. Dead Zone /Fig. 4/

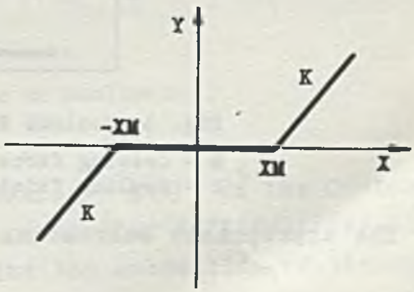


Fig. 4. Characteristic of the Dead Zone

-XM < X < XM - dead zone, K - amplification beyond the dead zone

The appropriate subroutine following directly from Fig. 4 will be:

<u>In SAKO</u>

```
SUBROUTINE: DEAD ZONE (X, XM, K)
Y = ABS(X) - XM
IF  Y > 0 : 1, ELSE NEXT
Y = 0
1) DEAD ZONE() = SGN (K × Y, X)
RETURN
```

<u>In ALGOL</u>

```
procedure  DEAD ZONE  (X, XM, K):  value X; real X:
begin real  Y;
Y  :=  abs (X) - XM;
DEAD ZONE := if  Y > 0 then sgn (K  Y, X)  else 0 .
                                end DEAD ZONE
```

3.4. Coulomb Friction /Fig. 5/

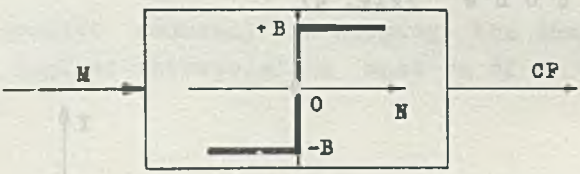


Fig. 5. Coulomb Friction Characteristic
M - driving force /or driving moment/,
B - Coulomb friction force, N - speed.

The appropriate subroutine following directly from Fig. 5 will be:

<u>In SAKO</u>

```
SUBROUTINE: CF(M, N, B)
()CF - COULOMB FRICTION
IF N = 0 : 1, ELSE NEXT
CF() = SGN(B, N)
RETURN
```

```
1) IF ABS(M) > B : NEXT, ELSE 2
CF() = SGN(B, M)
RETURN
2) CF() = M
RETURN
```

In ALGOL

```
procedure CF(M, N, B); value M, N; real M, N;
comment: CF - Coulomb friction;
CF := if N = 0 then (if abs(M) > B then sgn(B, M)
      else M) else sgn(B, N)
                          end CF
```

In general, the friction force B is not a constant value, it depends upon the speed N. For example, assume that this relation corresponds to the curve B(N) in Fig. 6.

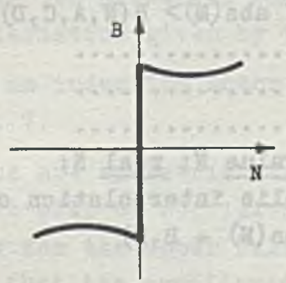


Fig. 6. Characteristic of Nonlinear Friction Force B

The consideration of the nonlinearity B(N) in the Coulomb friction subroutine does not provide particular difficulties. It is sufficient to add to that subroutine the subroutine of the second order B(N, A, C, D) and use its name to replace the symbol B in the Coulomb friction subroutine. As an example a few lines are given of the Coulomb friction subroutine modified in this way

In SAKO

```
SUBROUTINE:CF(M,N, B())
If N = 0 : 1, ELSE NEXT
CF = SGN(B(N), N)
    .........
    .........
    .........
RETURN

SUBROUTINE:B(N,A,C,D)
C)A,C,D - PARABOLIC INTERPOLATION COEFFICIENTS*)
B() = SGN(A × N*2 + C × ABS(N) + D, N)
RETURN
```

In ALGOL

```
procedure CF(M,N,B); value M,N; procedure B; real M,N;
CF := if N := 0 then(if abs(M)> B(N,A,C,D) then ...
                     ................
                     ................
                     ................
procedure B(N,A,C,D); value N; real N;
comment: A,C,D - parabolic interpolation coefficients;
B := sgn(A × N↑2 + C × abs(N) + B,N)
                                             end B
```

Using the Coulomb friction subroutine it should be remembered that the digital computer computes the behaviour of the analyzed circuit step by step.

Therefere, if the circuit is on, i.e. $N \neq 0$, and due to the Coulomb friction tends to be at rest, i.e. $N \to 0$, the value $N = 0$ is practically never reached, /the probability of the realization of computations during which the speed N reaches the zero value exactly on the boundary of the interval is negligible/, but the computations will show oscillations around the value $N = 0$ with

*) A,C,D are substituted as indicated in the note an page 10.

a period equal to a double magnitude of the integration step. This phenomenon can be rather simply avoided by means of appropriate changes in the Main Program, and a slight correction of the Coulomb friction subroutine. Analyzing the servomechanisms we are must interested in transient states of the circuit in which it is on; therefore, there is no need to consider the condition $N = 0$ except if it is initial. Besides, even in contrary cases, the above mentioned parasitic oscillations are small and easy to detect. Therefore, the above form of the Coulomb friction subroutine is satisfactory in the majority of cases .

3.5. Motor Characteristics

In view of a great variety of motor characteristics on account of types and differences dependent on particular copies of one type, a subroutine will be presented that permits to consider motor arbitrary characteristics given by the producer.

Assume that we have an induction motor the characteristics of which are shown in Fig. 7.

These characteristics are symmetric to the centre of coordinate set, and therefore it will be sufficient to give their mathematical description only for the upper half of the diagram, i.e. $U \geqslant 0$. Assume furthermore that the considered range of the motor run is within the limits $-30 \leqslant N \leqslant +30$, as indicated in Fig. 7 by the broken line.

Reading from Fig. 7 the numerical values $M(U_i, N_j)$ for the value U_i, every fifth of which changes within the limits from 0 to 25, and for N_j, every tenth of which changes within the limits from -30 to $+30$, we obtain the array of number M_{ij}

$$\begin{matrix} M_{00} & M_{01} & M_{02} & M_{03} & M_{04} & M_{05} & M_{06} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ M_{50} & M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{56} \end{matrix} \qquad /4/$$

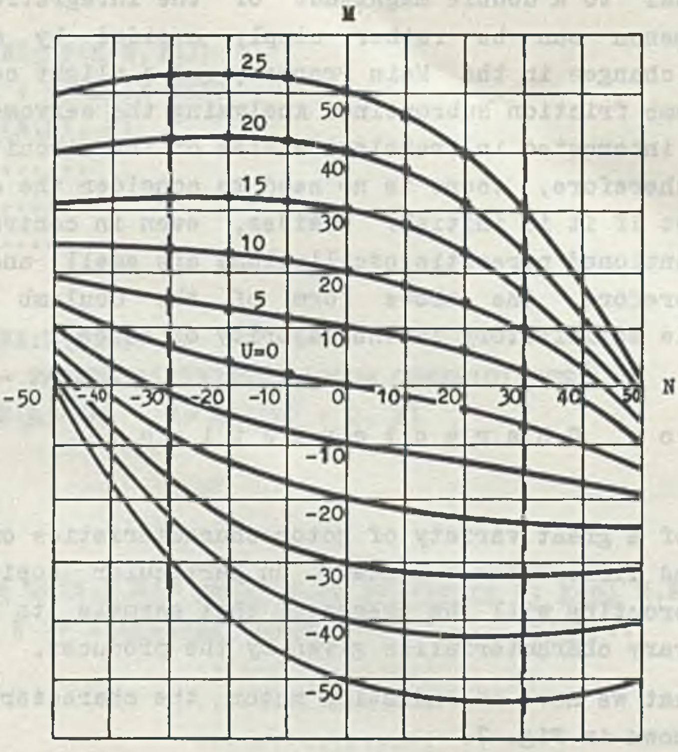


Fig. 7. Characteristics of Induction Motor
M - driving moment, N - rotation speed, U - control voltage.

This array determines the corresponding numerical values of the motor driving moment.

On the other hand, characteristics shown in Fig. 7 can be described by the equation of parabolic interpolation in the following way:

$$M'(u, n) = a_2(u)\, n^2 + a_1(u)\, n + a_0(u), \qquad /5/$$

where

$$\begin{aligned} a_2(u) &= a_{22}\cdot u^2 + a_{21}\cdot u + a_{20} \\ a_1(u) &= a_{12}\cdot u^2 + a_{11}\cdot u + a_{10} \\ a_0(u) &= a_{02}\cdot u^2 + a_{01}\cdot u + a_{00} \end{aligned} \qquad /6/$$

We have to choose values of coefficients a_{lk} appearing in the set of equations /6/ so as to obtain minimal mean square error φ between the values $M'(u_i, n_j)$, computed according to /5/ and /6/ and the real values $M(u_i, n_j)$,*) determined by the motor characteristics /Fig. 7/ recorded by means of number array M_{ij} /4/.

$$\varphi = \sum_{i=0}^{5} \sum_{j=0}^{6} \left[M'(u_i, n_j) - M(u_i, n_j) \right]^2 = \text{minimum} \qquad /7/$$

Differentiating this expression with respect to the successive values a_{lk} of the equation set /3/ we obtain 9 equations in which the index l takes the value 0, 1, 2 respectively for every value of k = 0, 1, 2.

$$\frac{d\varphi}{d\, a_{lk}} = \sum_{i=0}^{5} \sum_{j=0}^{6} 2\left[M'(u_i, n_j) - M(u_i, n_j)\right] \cdot \frac{\partial M'}{\partial a_{lk}} = 0 \qquad /8/$$

However, when writing equation /5/ taking /6/ into consideration, it is easy to notice that

$$\frac{\partial M'}{\partial a_{lk}} = n^l \cdot u^k, \qquad /9/$$

this, being substituted to the equation /7/, gives the set of nine equations with nine unknowns a_{lk} of the following form

$$\sum_{i=0}^{5} \sum_{j=0}^{6} \left[M'(u_i, n_j) - M(u_i, n_j)\right] n^l \cdot u^k = 0 \qquad /10/$$

The solution of this equation set gives the value of coefficients a_{lk} in /6/. Due to this, and using /5/, any value of the driving moment $M(u, n)$ can be computed in the considered area of motor characteristics /Fig. 7/.

---

* To make the record clear $u \equiv U$, $n \equiv N$ identities are taken.

The problem of writing a subroutine computing numerical values of coefficients a_{lk} on the basis of the numerical array /4/, using algorithm /10/, is omitted. This is a problem of a formal nature, to be handled by the programmer if a standard subroutine of this kind is not to be found in the subroutine library.

A general subroutine*) is given below using /5/ and /6/, computing the driving moment of motor M in the functions of its rotation speed N and control voltage U. The values of interpolation coefficients a_{lk} of equation /6/ are assumed to be known; they are computed on the basic of known values $M(U_i, N_j)$ determined by the motor characteristic /Fig. 7/.

In SAKO

```
SUBROUTINE:DMM(U,N,*A)
C)DMM-DRIVING MOMENT OF MOTOR
C)U-CONTROL VOLTAGE
C)N-ROTATION SPEED
C)A-INTERPOLATION COEFFICIENTS
STRUCTURE (2,2): A
Z=1
IF 0>U:NEXT, ELSE 1
U=-U
N=-N
Z=-1
1) A2=A(2,2)×U*2+A(2,1)×U+A(2,0)
A1=A(1,2)×U*2+A(1,1)×U+A(1,0)
A0=A(0,2)×U*2+A(0,1)×U+A(0,0)
MNS()=Z×(A2×N*2+A1×N+A0)
RETURN
```

---

* Due to the fact that the control motor voltage takes only two values +U and -U a simpler subroutine can be used in the program /page 4 and 5/ to compute the nonlinear servomechanism /Fig. 1/. Therefore, analogically to formula /5/, the corresponding characteristic can be defined as follows:

$$M(n) = a_2 n + a_1 n + a_0$$

where a_2, a_1, a_0 are the coefficients of parabola passing through the chosen points of motor characteristic for a given value U.

In ALGOL

```
procedure DMM(U, N, A)  value U, N;  array A; real U, N;
comment: DMM - driving moment of motor
U - control voltage
N - rotation speed
A - interpolation coefficients;
begin real Z,A2,A1,A0
if  0>U  then begin  U := -U;  N := -N;  Z := -1  end
                     A2  := A[2,2]×U↑2 + A[2,1]×U + A[2,0];
                     A1  := A[1,2]×U↑2 + A[1,1]×U + A[1,0];
                     A0  := A[0,1]×U↑2 + A[0,1]×U + A[0,0];
                     DMM := Z×(A2 × N↑2 + A1 × N + A0)
                                                       end DMM
```

In SAKO the numerical values of the array A are substituted either directly by the name of the function calling in the subroutine or by means of the SUBSTITUTE : DMM(·, ·, *A).

It should be emphasized that in the case of parabolic interpolation /2/ and /3/ when the obtained mean-square error φ is too great, the interpolation used must be of correspondingly higher order. However, the procedure itself for computing the interpolation coefficients a_{lk} as well as the content of subroutine DMM remain the same. Moreover it is to be pointed out that the described subroutine, based on the method of the smallest mean-square error, can be used for computing the relations occurring in any element of the circuit described by the characteristics of the type $Z = F(X, Y)$.

3.6. Relay Element with Hysteresis

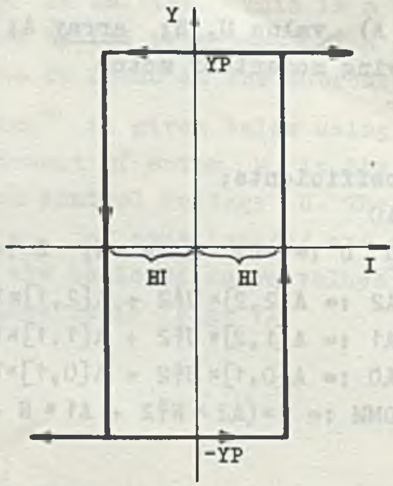


Fig. 8. Relay Characteristic

YR - amplitude of the relay output signal, 2HI - width of the hysteresis loop.

The appropriate subroutine following from Fig. 8 will be:

In SAKO

```
SUBROUTINE: RELAY (I, YR, YRO, HI)
C)YRO - INITIAL OUTPUT SIGNAL OF RELAY
IF ABS(I) > HI: NEXT, ELSE 1
YRO = SGN(YR, I)
1)RELAY() = YRO
RETURN
```

In ALGOL

```
procedure RELAY (I, YR, HI); value I; real I;
comment YRO enters into RELAY as a nonlocal entity;
begin
if abs(I) > HI then YRO := sgn(YR, I); RELAY := YRO
                                        end RELAY
```

Using a SAKO subroutine it should be remembered that the number value of the argument YRO can be substituted only once by means of the statement SUSTITUTE : RELAY(·, ·, YRO), before further computation steps of the main program are realized /for an example refer to the main program on page 4/. Hence, it appears that when realizing the subroutine the variable YRO is given as the resulting value YR or -YR stored by the subroutine as the initial value for the next step. Moreover, if the system has several relays, the state of each of them must be computed by means of a separate subroutine with the appropriate name.

4. Example of Computation

For the described nonlinear servomechanism /Fig. 1/ the SAKO program is given on page 4. The latter renders it possible to compute the servomechanism operation on a digital computer.

As an example, let us assume the numerical data written below in the order as they are read into the machine, and determined by the instruction READ IN DECIMAL: their form is like the one obtained on the teleprinter sheet:

```
NUMBER DATA:

W = 20.
H = 0.1
R = 1.
L = 1.
I = 5.
B = 10.
U = 25.
UO = 25.
HI = 0.25
A = 50. -0.385 -0.0095
          *
Y = 0.  0.  0.
        *
```

The example discussed was computed on the ZAM-2 digital computer. The time of computation was circa 6 min. The results are presented in the form of a diagram in Fig. 9.

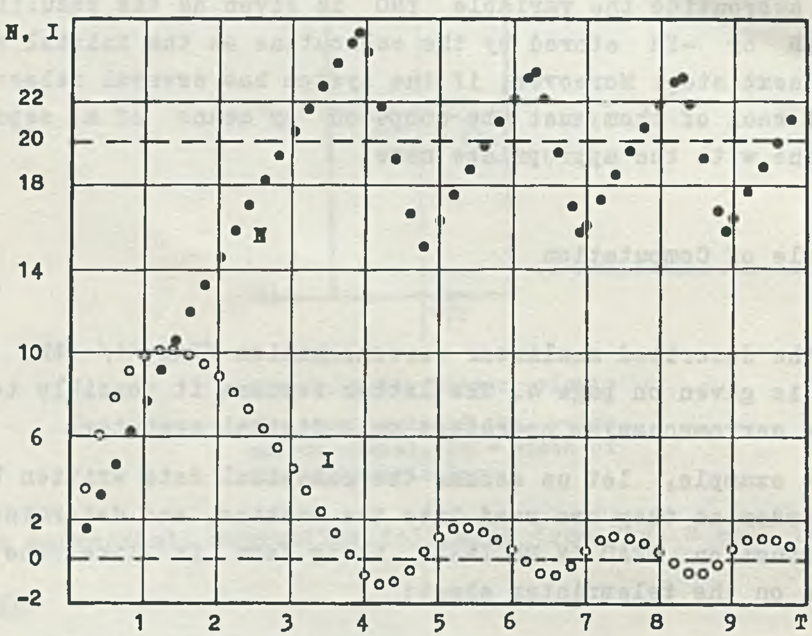


Fig. 9. Nonlinear Servomechanism Computation Results
N - output speed of the servomechanism, I - relay control current.

5. Conclusion

Nonlinear subroutines described in the paper and the way of their use indicate how simply circuits with any type of non-linearity can be computed directly on digital computers with the use of SAKO or ALGOL. The further step making the computer use much easier, is to provide the subroutine library with a large assortment of subroutines describing operations of different types of nonlinearity. Then, for computing the examined circuit, the constructor must only describe its operation by means of differential equations of the 1st order by the direct method presented in [1]. He will write the main program /which is rather simple, as shown in the given example/, and add to it the Runge Kutta subroutine, the right side subroutine and appropriate nonlinear subroutines from the subroutine library.

APPENDIX

The SAKO program in an example of a nonlinear servomechanism /Fig.1/.

The SAKO program presented in this paper serves to compute the response of a nonlinear servomechanism /Fig. 1/ to the stepping input signal on a digital computer. Symbols used in the program are similar to those used in servomechanism diagram denotations as well as to those introduced to the description of nonlinear subroutines.

The discussed program ought to be divided into two chapters, as otherwise, if translated by the translator into the machine language, it could not be comprised as a whole in the operational storage*) of the ZAM-2 digital computer performing the computations.

CHAPTER: '0' sets the computation scale and reads numerical values of servomechanism parameters into the digital computer operational storage. This is done by means of the statement READ IN DECIMAL : W, H, R, ... which, when realized, reads in the values recorded on a punched tape, successively into operational storage places correspondingly called W, H, R, ...

It should be emphasized that the places in the operational storage for these parameters have been previously declared as arrays.

---

* A digital computer has two storages, the operational storage, and the external one. The machine performs all operations by means of the operational storage. Separate chapters, translated into machine language are recorded in its external storage. On transition to the next chapter, this very chapter is rewritten from the external storage to the operational one, in place of the chapter performed previously.

The declaration ARRAY (dimensions of the array):(list of names of the arrays) reserves certain definite places in the operational storage for names indicated on the list according to their order and according to the order in which the ARRAY declarations appear successively in the given chapter. Keeping the same order in both chaters, when statements of chapter 1 are entered into the operational storage as a result of the statement JUMP TO CHAPTER 1, numerical values read into the chapter 0 are transmitted to chapter 1 without any change.

Three successive statements are in principle the main content of CHAPTER : 1. They begin with the statement marked by number 1). The statement PRINT (5.2) : Y(0), Y(1), Y(2) causes the printing of numerical values /which 5 significant digits before the point and two of them after the point/ which are present in the storage places reserved by the declaration ARRAY(2) : Y. The next statement (*Y) = RK (*X, *S, *M, F(), 2, H)*) is the formula calling in the SUBROUTINE Y = RK(), that realizes one integration step by the Runge-Kutta method. After the performance of the computation achieved by the above subroutine /using also the remaining subroutines/, new numerical values are to be found in the array Y that correspond to the variables characterizing the response of the servomechanism at the moment T = T + H.

The statement REPEAT FROM 1:I = 0(1)100 causes 101 time repetitions enclosed between the statement marked by number *1) and the statement REPEAT ...**)

This results in 100 lines, each containing 3 numbers printed on a teleprinter. These number values correspond to variables T,I,N

---

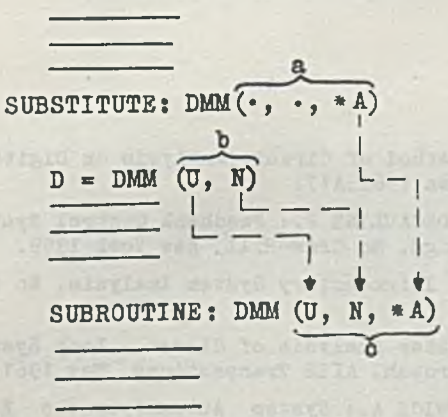
* The asterisk before a literal symbol, e.g. *Y means that this symbol is the name of a numerical array which we understand to be a finite and ordered group of numbers

** In realizing this statement, the variable I takes successively, starting from 0, the values I = I + 1 till it reaches the value 100 /in the given example no use is made of this property/. The digital computer then performs the next statement which is in our program the statement STOP 2 which makes the machine discontinue its work.

of the set of equations (1) that describes the servomechanism operation. Each line gives the results of the successive integration step performed by means of the Runge Kutta method.

We still have to describe the meaning of the statement SUBSTITUTE: (name of the subroutine) (list of arguments substituted). To get a good comprehension of the above, the best way would be to investigate the procedure of calling in, e.g. the subroutine DMM(U, N, * A), where DMM is its name, and U, N, * A are so-called arguments. The subroutine operates with numerical values, determined by its arguments, according to its statement list.

Symbols used in the given subroutine /in the discussed case, symbols U, N, * A/ are independent from those of the main program and from those of other subroutines /i.e. the mentioned symbols can be replaced for instance by those A, B, *C/. It means that when a subroutine is called in the main program must give numerical values to the above variables. This is accomplished either by means of the formulae calling in the subroutine or by the statement SUBSTITUTE, as shown in the following example:



where:

a - list of arguments substituted by the statement SUBSTITUTE,
b - list of arguments substituted by the calling in formula,
c - list of the subroutine arguments.

The order and character of symbols placed on the list of the formula calling in the subroutine, or those of the statement SUBROUTINE, must correspond to the order and character of symbols in the declaration SUBROUTINE, as indicated by the broken line. Points on the list of substituted arguments of the statement SUBSTITUTE serve only to determine the proper succession of the substituted arguments, i.e. the number of points on the list preceding the given argument determine its succession among other arguments.

Considering the above explanation when analyzing the first chapter the reader should have no difficulties to understand the meaning of the statements SUBSTITUTE occurring in this program. Statement STOP 2, according to their content, stop the machine on the statement denoted by 2. Then the button START is pressed on the machine desk, the machine starts again to realize the program beginning with chapter 0.

The content of the subroutines used in the presented program was discussed in the main content of this paper.

References

1. ŁUKASZEWICZ R.: Direct Method of Circuit Analysis on Digital Computers, Prace ZAM, Warszawa 1962:A17.

2. GILL J.C., PELEGRIN M., DECAULINE P.: Feedback Control Systems, Analysis, Synthesis and Design, Mc Graw-Hill, New York 1959.

3. LYNCH W.A., TRUXAL J.G.: Introductory System Analysis, Mc Graw Hill, London 1961.

4. ADAMS R.K.: Digital Computer Analysis of Closed - Loop Systems Using the Number Series Approach, AIEE Transactions, May 1961.

5. ŁUKASZEWICZ L., MAZURKIEWICZ A.: System Automatycznego Kodowania SAKO, Ossolineum, Warszawa 1963.

6. SZMELTER J., DELOFF K.: Podręcznik SAKO dla początkujących, Politechnika Łódzka, Łódź 1962.

7. NAUR P.: Report on the Algorithmic Language ALGOL 60, Communications of the ACM, May 1960:3,5.

8. RALSTON A.: Mathematical Method for Digital Computers, John Willey, London 1960.

W.D.N. Zam. 211/0 63 Nakł. 500 egz.

A 2(19)