

BACS-A Block Diagram Oriented Digital Simulation Program for the IBM S/360

TSUNETOMO MATSUURA* AND MORIKI TOYAMA**

1. Introduction

It is particularly desirable to perform efficient digital dynamic simulation of control system with a problem oriented language not to require difficult digital programming techniques.

This paper describes the features of a block diagram oriented digital simulation language called BACS (Block diagram Analysis Compiler System) and the program structure of BACS.

BACS was developed in 1964 for IBM 7090 [1] and applied in the solutions of many types of control systems and improved in 1968 for IBM S/360.

Use of the program has successfully been made within the Mitsubishi Electric Corp. in the study of control systems by engineers who are unfamiliar with, or learned a few days course of FORTRAN.

BACS is designed for not only a block diagram system simulator, but also a general purpose program of the numerical solutions of differential equations.

2. Features of BACS

BACS provides the following desirable features of digital simulation language.

(1) BACS is a block diagram oriented style. It results from the fact that modern automatic control theory is much connected with a block diagram.

(2) BACS language has the simplicity of usage. BACS input program coded by users indicates clearer correspondence with a block diagram or differential equations.

(3) BACS has the flexibility in a user program which is compatible with the simplicity of usage. In performing the simulation of complex systems, a control engineer can use FORTRAN in his simulation program.

(4) Computer time processed by BACS compiler before the execution of a user simulation program is negligibly short in comparison with the execution time. It is more economical to use BACS than other similar languages.

(5) On-line simulation could be expected by BACS.

This paper first appeared in Japanese in *Joho Shori* (the Journal of the Information Processing Society of Japan), Vol. 10, No. 4 (1969), pp. 216-226.

* Information Systems Department, Mitsubishi Electric Corp.

** Process Computer System Department, Mitsubishi Electric Corp.

3. Structure of BACS

BACS program consists of BACS compiler and many routines such as integration, transfer functions, diagnostics, resetting parameters and run control. The structure program to be generated by BACS compiler for users is the subprogram which represents the modeled structure of a block diagram, and can be connected with any other programs. BACS compiler is a straight forward one-pass compiler which directly and only once translates each of all user-coded structure statements into several machine language instructions, and therefore provides the most economical performance.

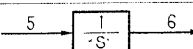
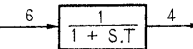
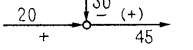
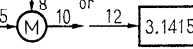
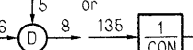
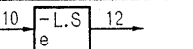
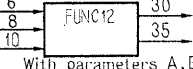
Function	Example structure	BACS structure statement
Integral		5 I = 6 0.0
1st order lag		6 F T = 4 1.53
Summer Subtractor		20 - 30 = 45 (+)
Multiplier		5 * 8 = 10 or 12 * 3.14159 = 14
Division		6 / 8 = 135 or 135 / CON = 20
Dead time		10 D = 12 L
User-coded function	 With parameters A,B	(12) = A B
Besides, limiter, dead zone, hysteresis, logic functions, sine, exponential square root, absolute value, quantizer, ramp function, uniform random number, etc. are contained in library. Algebraic functions are also utilized as signal functions.		

Table 1. BACS functions in library.

4. Input language

The BACS input program consists of three parts: (1) the structure program which represents the modeled structure with functions in library illustrated in table 1 and their connections of a block diagram, (2) numerical values assigned to the parameters in a block diagram as the input data for each run, and, if necessary, (3) user-coded functions freely defined in the form of FORTRAN subprogram. A symbolic coding form with free format was chosen in BACS in order to reduce the volume of a user program and to make its programming easy and to compile it fast. In using BACS, control engineers transfer their block diagrams into the representations with BACS functions by minor changes and call each function-output variable by a number randomly named at their option, that is, a line number in a block diagram, and write in free format BACS structure statements describing the functional relationships among variables of a block diagram. For

example, an integral function

$$\int_0^t X_{15}(u) du + K = X_{20}(t)$$

is written in BACS input form as follows :

15 I =20 K

5. *Integration method*

In BACS Runge-Kutta-Gill's, modified predictor-corrector Hamming's and many other integration methods are available at user's option.

6. *Output*

BACS standard output routine prints or plots the numerical values of the variables which are marked by the symbol "*" or any alphabetic character respectively at the head of output line numbers in structure statements. Furthermore, there is BACS diagnostic routine for a user program.

7. *User-coded algebraic function*

BACS user-coded function is a FORTRAN subprogram for special purposes in separate fields, in which all kinds of the FORTRAN statements can freely used. The examples in the application of user-coded functions are (1) to generate table functions, (2) to calculate initial values, (3) to analyze output data, (4) to perform run control, (5) to define complex non-linear functions including implicit functions, (6) to print specified solutions at any time, and (7) to call other programs. In the simulation run including user-coded functions, only the additional user-coded functions are processed by FORTRAN translator and BACS linkage-editor. Other parts of a user program are compiled by BACS system only.

8. *Successive runs*

Successive runs are performed for many parameters such as gains, time constants, initial conditions, etc in order to reduce turn-around time, or to find automatically the optimized values of parameters in control systems, in a model without recompilation by DATA or PROCON statements respectively.

9. *Example 1*

A simple example is provided by the study of a position control system made up of a Ward-Leonard drive. Figure 1 and 2 show the block diagram of the system and the BACS user program listing.

10. *Example 2*

The illustration of the pilot ejection system was used in the description of MIDAS, MIMIC and CSMP [3], but it is also illustrated in this paper by reason

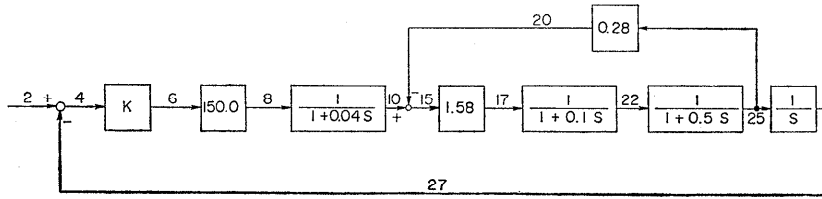


Fig. 1. Block diagram of Ward-Lonard drive system.

```

* EXAMPLE WARD-LEONARD SYSTEM STUDY
* SPEC. OF METHOD, STEP SIZE, PRINT
RUNGE 0.01 5.0 10.0 AUTOSCALE
* SPECIFICATION OF STRUCTURE PROGRAM
STP = 2 0.0 1.0
2 - 27 = 4
4 * K = E 6
6 * 150.0 = 8
8 F 0.04 = 10 0.0
25 * 0.28 = 20
10 - 20 = 15
15 * 1.58 = 17
17 F 0.1 = A 22 0.0
22 F 0.5 = V 25 0.0
25 I = T 27 0.0
END
DATA CASE 1
K 0.015
DATAEND

```

Fig. 2. BACS user program listing of Ward-Lonard drive system.

that it is a suitable study which compares with simulation languages. The purpose of this simulation study is to ascertain whether a pilot ejected from an aircraft will strike the vertical stabilizer of the aircraft or not. The pilot travels along rails at a specified exit velocity, V_E at an angle, θ_E , backward from vertical. The equations of motion are as follows:

$$\begin{aligned}
 \dot{X} &= V \cos(\theta) - V_A \\
 \dot{Y} &= Y \sin(\theta) \\
 \dot{V} &= 0 & (Y \leq Y_1) \\
 \dot{\theta} &= 0 & (Y \leq Y_1) \\
 \dot{V} &= -D/M - G \sin(\theta) & (Y > Y_1) \\
 \dot{\theta} &= -G \cos(\theta)/V & (Y > Y_1)
 \end{aligned}$$

$$D = 1/2 \rho(H) C_D S V^2$$

$$\text{where } V(0) = ((V_A - V_E \sin(\theta_E))^2 + (V_E \cos(\theta_E))^2)^{1/2}$$

$$\theta(0) = \tan^{-1}(V_E \cos(\theta_E)/(V_A - V_E \sin(\theta_E)))$$

Our main concern is not to determine the ballistic trajectory of the pilot itself for a large number of airplane velocity V_A and altitude H conditions, but to find the (V_A, H) domain or envelope of safe ejection with the form of $H \geq f(V_A)$. Figure 3 shows one of the block diagrams of the pilot ejection system equations. The structure program and one of the user-coded functions program listing is illustrated in figure 4. In FUNC5, the initial values V_0 and θ_0 for V and θ are calculated only once before each run. FUNC10 is the function which ascertains whether the ballistic trajectory of the pilot has been safe or not and which

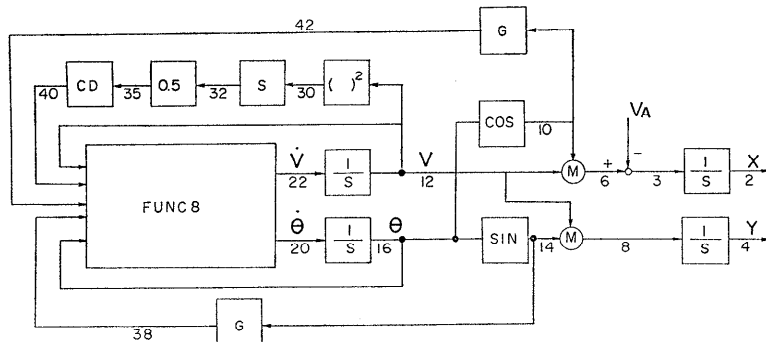


Fig. 3. Block diagram of pilot ejection system.

```

* EXAMPLE PILOT EJECTION SYSTEM STUDY
* SPECIFICATION OF METHOD, STEP SIZE, FINAL TIME ETC.
  RUNGE   DT   FMT
* SPECIFICATION OF STRUCTURE PROGRAM VA VE THE
  (5)    =    10   VO   THO   VA   VE   THE
  16 COS   =    6
  10 *    12 =    3
  6 -     VA =    2   0.0
  3 I     =    14
  16 SIN   =    8
  14 *    12 =    4   0.0
  8 I     =    30
  12 *    12 =    32
  30 *    S   =    35
  32 *    0.5 =    40
  35 *    CD  =    38
  14 *    G   =
  (8)    =    12   Y1   M   H   RI   HTR
  22 I   =    16   VO   THO   VA   FTM   HTR
  20 I   =    H
  (10)   =
END
DATA
DT 0.2 FTM 3.0 M 7.0 G 32.0 Y1 4.0
VE 40.0 THE 15.0 S 10.0 CD 1.0 H 0.0
VA 500.0 RI 1.0 HTR 1.0
PROCON

* DEFINITION OF FUNC8 FOR VDOT, THEDOT
SUBROUTINE FUNC8(TIME,X,Y1,A,H,RI,HISTRY)
DIMENSION X(800),DATA(2,100)
IF(RI.GT.1.0) GO TO 30
READ(5,10)(DATA(1,I),DATA(2,I),I=1,15)
10 FORMAT(6E12.4)
RI=2.0
30 IF(HISTRY.GT.1.0) GO TO 50
IF(X(4).GT.Y1) GO TO 50
X(22)=0.0
X(20)=0.0
RETURN
50 X(22)=-X(40)*CURVE(H,DATA)/A-X(38)
X(20)=-X(42)/X(12)
HISTRY=2.0
RETURN
END

```

Fig. 4. BACS program listing of pilot ejection system.

resets new values to the parameters in order to find the safe conditions at only one input trial and which terminates the runs.

11. Conclusion

The features of a simulation language BACS described in this paper are the simplicity of usage, the flexibility in programming and the efficiency of the operating system including the compiler. Therefore, BACS appears to be suited and practical for the simulation of control systems.

References

- [1] Matsuura, T. and M. Toyama, Block Diagram Simulator with a Digital Computer, *Journal of Society of Analog Technique of Japan*, 4, 6 (Aug. 1964).
- [2] Takahashi, M., T. Matsuura, M. Toyama, and S. Hayashi, Block Diagram Simulator with a Digital Computer, *Fifth International Congress of AICA* (Aug. 1967).
- [3] Brennan, R. D. and M. Y. Silberberg, The System/360 Continuous Modeling Program, *Simulation*, 11, 6 (Dec. 1968).