A MICROCOMPUTER–BASED SIMULATOR FOR DIGITAL CONTROL SYSTEMS

B. Farhang-Boroujeny

Department of Electrical Engineering
Isfahan University of Technology
Isfahan, Iran

Received June 1988

Abstract A microcomputer-based simulator for digital control systems is proposed. The proposed simulator is a hybrid system in which the plant is simulated by conventional method of analog computers and other parts of the system including generation of input signal is performed digitally, using a Z-80 based microcomputer. To overcome the difficulty of programming in assembly language, and at the same time to have a fast system, a special purpose simulating language is suggested. Examples of programming in the new language are also presented.

چکیده A Microcomputer-Based Simulator For Digital Control Systems یک دستگاه شبیه ساز سیستمهای کنترل دیجیتال مبتنی بر ریز کامپیوترها عرضه میگردد. شبیه‌ساز عرضه‌شده یک سیستم هایبریداست که در آن برای شبیه‌سازی سیستم تحت کنترل از روشهای معمول کامپیوترهای آنالوگ استفاده شده است ، و بقیه بخشهای آن به کمک یک ریز کامپیوتر مبتنی بر Z-80 شبیه‌سازی می‌شود . برای برخورد با مشکل برنامه‌ریزی کنترل کننده به زبان ماشین ، و در عین حال برای داشتن یک سیستم سریع ، یک زبان شبیه‌سازی خاص طراحی گردیده و مثال‌هایی از برنامه‌ریزی به این زبان نیز عرضه می‌شوند .

INTRODUCTION

The Existence of low price microprocessors in the market has evolutionized the control systems during the last decade. Conventional analog controllers (compensators) are replaced by the microcomputer based controllers. Some of the most important reasons for this replacement are the followings [1]-[5]:

i) Microcomputer based controllers are usually less expensive than their analog counterparts.

ii) They are also, smaller and lighter than their analog counterparts.

iii) Due to their very nature, the microcomputer-based controllers offer the highest degree of flexibility and they can be easily employed for the self training systems, while their analog counterparts are, generally, fixed and they can not offer such facilities. The microcomputer-based controllers also may be designed to have the self test features.

iv) Higher accuracy and higher reliability of microcomputer-based controllers should also, be considered.

With the above trends and privilages in mind, the new control engineers should have experience with digital controllers. Figure 1 shows the basic structure of a closed loop

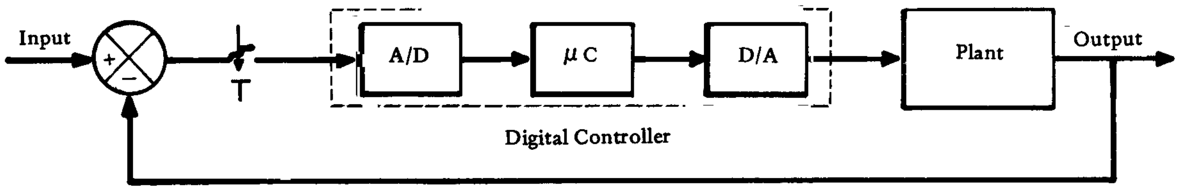


Figure 1. A digital control system

digital control system. In order to guarantee system stability and to have desired performance, a proper digital controller should be designed. The procedure for designing the digital controllers is well documented in the literature [1]-[3]. In general, a digital controller is expressed by the following input-output relation:

$$u_n = \sum_{i=0}^{M} a_i e_{n-i} - \sum_{i=1}^{N} b_i u_{n-i} \qquad (1)$$

where e_n and u_n are input and output samples of the controller respectively, and a_i and b_i are the controller coefficients, which are chosen for proper performance of the feedback system. In most of the practical systems, the values of M and N are either 1 or 2.

When the controller (compensator) is designed, i.e. a relation similar to (1), with proper coefficients chosen, it should be programmed on a microcomputer, or it should be implemented by a special purpose hardware. If a microcomputer is employed in the process of the programming, the programmer will encounter the following problems:

a) It takes time for the microcomputer to calculate the sample u_n. It means that there is time lag between sampling e_n and supplying u_n to the plant. This lag may degrade the system performance, unless the controller designer has already considered it.

b) To minimize the time lag between e_n and u_n, the fixed point numbers are used to present the controller coefficients, and input and output variables e_n and u_n The wordlength of these numbers should also be chosen to be minimal, as computational time increases linearly (or more) with wordlength.

c) Numerical, or microcomputer-based implementation of the controllers will result in the So-called roundoff noise,which adversely affects the system performance. The roundoff noise is inversely proportional to the chosen wordlength and it is more severe when fixed point numbers are used, as compared with floating point numbers. Although, the roundoff noise and its effect in digital control systems are studied theoretically to some extent [6-8], when real systems are implemented some odd effects will result which are difficult to be predicted theoretically. In these situations the best way to make sure of the system performance, under practical conditions, is to simulate it.

d) When fixed point numbers are employed, during the course of the controller calculations overflow may occur, which will result in a wrong u_n and it may cause undesired response at the system output. This effect can easily make the system unstable. In order to prevent such catastrophic effects, scaling techniques should be used [6], [9].

In this paper the design and implementation of a flexible digital control systems simulator is reported. This simulator was employed as an educational aid for a coruse in digital control systems at the Isfahan University of Technology, Isfahan, Iran, and it was found to be very instructive.

SYSTEM HARDWARE

Figure 2 shows a block diagram of the proposed simulator. The plant is simulated using the conventional method of analog computers, i.e. with the aid of a few operational amplifields, resistors, and capacitors. The microcomputer is used to perform the following actions:

a) To supply the proper input signal. Pulse, step, ramp and sine are possible options.

b) To get a sample from the output of the plant, subtract it from the input signal sample to construct the error signal sample e_n. It is also used to calculate u_n, according to Eq. 1, and to transmit the result to the plant input via D/A circuitry.

c) To take samples from different parts of the plant and to store them in proper tables. These tables will be displayed on the screen of an oscilloscope, after completion of simulation.

The microcomputer used is a Z-80 based system called MT-80Z [11]. This is an educational system with buffered buses to facilitate easy extention. To suit our purpose MT-80Z was augmented by adding A/D and D/A facilities to it. Figure 3 shows the hardware structure of the proposed microcomputer. It includes 2kbyte of ROM which contains the system monitor program, 4kbyte or RAM, 2kbyte of that is devoted to the simulator software, and the next 2kbyte are used for tables and variables, and some space is left for user programs. To aid sampling from different points of the plant an 8-channel A/D convertor is employed. The D/A convector is connected to both the plant and oscilloscope inputs. A Z-80-CTC has also been employed for sampling period adjustment.

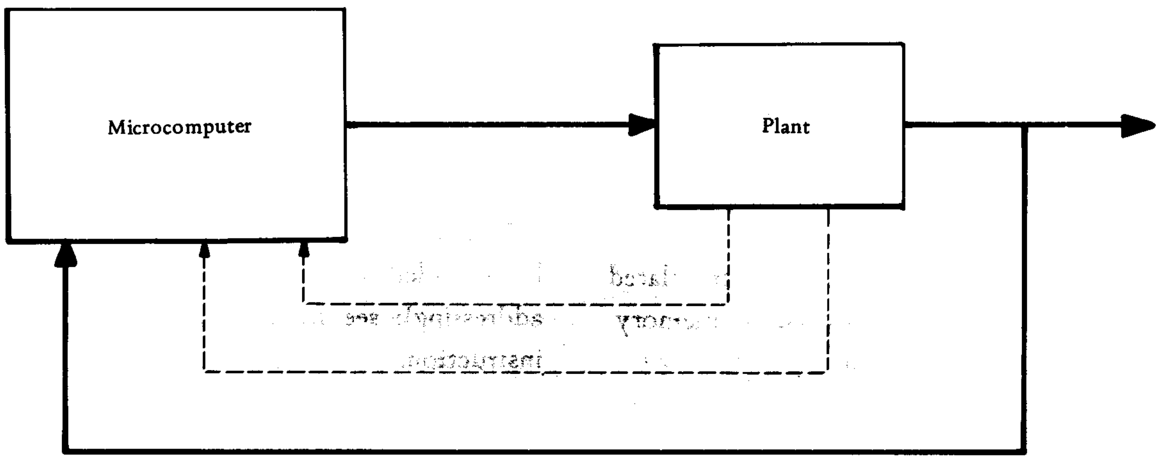


Figure 2. The simulator block diagram

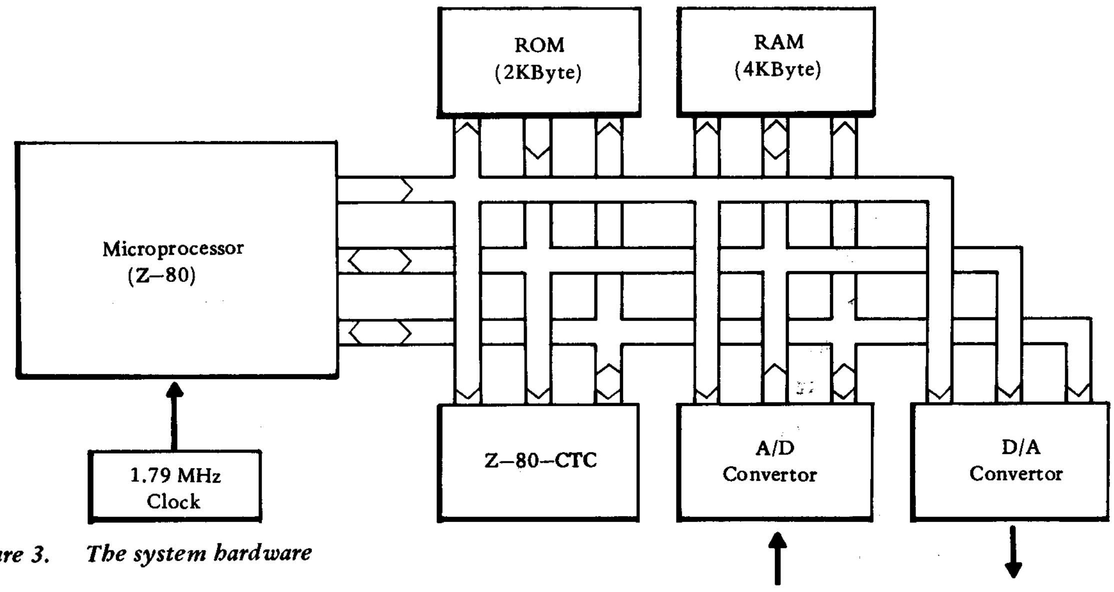


Figure 3. The system hardware

SYSTEM SOFTWARE

Application of assembly language for programming a microcomputer-based controller is rather cumbersome. On the other hand application of high level languages, like BASIC, will result in slow programs, which are not feasible in most situations.

To overcome the above problems, a special purpose language is suggested. It is similar to the block diagram language (BDL) which is suggested by Higuchi et al [10], for the implementation of digital filters. It is an assembly like language, and it is called in this paper digital control systems simulator language (DGCSSL). To consider its wider instruction set, as compared with the BDL, Table 1. shows the suggested instruction set of the DGCSSL. With the aid of this instruction set one can easily write a program to simulate a digital control system, and to present the results, as he likes, on a conventional oscilloscope. The programs are translated to codes, and they are stored in memory via a hexadecimal keyboard.

Also, shown in Table 1. are the instructions op-codes and their operand fields. Estimates of execution times of different instructions are also given. These times correspond to 1.79MHz clock.

The suggested instruction set is based on the assumption that we have a virtual machine with twenty accumulators (A0-A19) and eight counters (C0-C7). The wordlength of accumulators and counters are assumed to be 16 and 8 bits, respectively. Four tables, each of length 256 bytes, are also proposed (T0-T3). Examples of programming with the DGCSSL will be presented in the next section. But, before that the following comments should be given:

– The STRT instruction is employed to program the system timer (Z-80-CTC), to adjust the sampling period. During execution of this instruction, the microprocessor's interrupt line will be, also, enabled. The latter will be disabled by the STOP instruction.

– Moving the content of one accumulator into another is essential for simulation of delay units in a block diagram (MOVA). To be able to present the successive values of a variable on an oscilloscope, moving the content of an accumulator into a table is usefull (MOVT). A table can, also, be loaded from an A/D channel (ATDC Chi; Tj). The ITBP (increment Table Pointer) is used to access successive locations (bytes) of the tables.

– The counters in conjunction with the LOOP instruction can be used to control the length of a simulation run time. Other useful features can be gained by clever application of the LOOP instruction; an example will be given in the next section.

– The controller coefficients are suggested to be included in the instructions (immediate addressing); see MULA (Multiply and Add) instruction.

– The DELY instruction is useful when one is interested to see the effect of the time delay which results in the process of calculation of the controller output samples.

– In order to have a proper performance, the controller calculations should be over before the end of each sampling period. The WAIT instruction is essential to ask the microprocessor to wait for the start of the next sampling period.

– In many applications, special programs may be written to minimize the time delay in the calculation of the controller output samples. To give this opportunity and other options to the user, five user definable routines are proposed. The starting address of these routines are at some specified locations of the RAM memory.

Table 1 The instruction set of the DGCSSL.

Instruction codes (Hexadecimal)	Mnemonics	Typical Execution Times (μScc.)	Comments
|00|	STRT	147	Start simulation
|01| i | j |	MOVA Ai, Aj	85	Ai ← Aj
|02| i | j |	MOVT Ti, Aj	106	Move Aj into the current location of Ti.
|03|	ITBP	44	Increment table pointer
|02| i | n |	MOVC Ci, n	62	Move counter immediate
|05| nL |nH|	JUMP nn	55	Jump instruction
|06| i |nL|nH|	LOOP Ci, nn	83	Decrement Ci and if the result is non-zero jump to address nn
|07| i | j |	ADDA Ai, Aj	93	Ai ← $A_i + A_j$
|08| : | j |	SUBA Ai, Aj	93	Ai ← $A_i - A_j$
|09| i | p|	SHFT Ai, p	150+16.8p	Shift arithmethic Ai p-bits. Positive p means shift right, & negative p means shift left
|0A| i |	DTAC Ai	103	Send Ai to the D/A convertor
|0B| i | j |	ATDC Chi, Tj or ATDC Chi, Aj	372 457	Take a sample from channel "i" and store it in Tj or Aj. If the second byte is 128+i the result will be stored in Tj, and in Aj if it is "i".
|0C| i | s | e |	DISP Ti, s, e	1936 + 11.7(e–5)	Display the specified locations of tab. Ti; s stands for starting address and e for ending address.
|0D| j | KL | KH | i |	MULA Ai, Aj, K	800	Ai ← Ai + K.Aj

|0E| i |	INPT Ai	105(pulse)
		92(step)
		145(ramp)
		1200(sine)
			Input a new sample into Ai

|0F|	STOP	49	STOP simulation
|10|	NOPR	31	No-operation; useful for program development
|11|m|	DELY m	41 + 100 m	m units delay; a unit is. lmsec.
|12|	WAIT	–	wait for end of a sampling period
|13|	USR0	–	User definable instruction
|14|	USR1	–	User definable instruction
|15|	USR2	–	User definable instruction
|16|	USR3	–	User definable instruction
|17|	USR4	–	User definable instruction

After writing the simulator routine, the programmer should translate it, and store it in system memory. The programmer should then run the simulator routine. The simulator routine is divided into two sections, namely the man-machine interactive routine and the fetch and execute routine. The actual simulation is performed by the latter. In the man-machine interactive routine, the following parameters are supplied to the microcomputer: (i) The simulator sampling period. This is the time interval during which samples are taken from different parts of the plant, and they are stored in the tables for later observation. This period may be different from the controller sampling period. For more detail and examples please refer to the next section. (ii) Fixed point position for number representation. (iii) Type of input signal, and the corresponding parameters. Possible options are pulse, step, ramp, and sine wave. (iv) The wordlengths of A/D convertor, virtual machine, and D/A convertor should be also chosen. Maximum wordlengths are 16 bits for accumulators, and 8 bits for A/D and D/A convertors. The user has options for either truncated or rounded variables.

EXAMPLES

In order to present the simulator features, a plant with the following system function is considered.

$$G(S) = \frac{30}{(1+S)^2(1+S/10)} \tag{2}$$

If this plant is used in a feedback loop, similar to Figure 1, but without any compensator, it will be an unstable system. A simple DGCSSL program for simulation of feedback systems, with no compensators is shown in Prog. 1. Figure 4 shows a result of this routine for step input, and for the proposed plant.

The presented plant may be compensated by the application of any standard method. If a sampling period of T=0.3sec. is chosen and w-plane technique is used to achieve a phase margin of 60Deg., we may find the following controllers:

i) Lag-controller

$$D(Z) = \frac{0.0365 - 0.02986Z^{-1}}{1 - 0.99335Z^{-1}} \tag{3}$$

ii) Lag-lead controller

$$D(Z) = \frac{0.2376 - 0.3168Z^{-1} + 0.1056Z^{-2}}{1 - 0.6469Z^{-1} - 0.3267Z^{-2}} \tag{4}$$

In general, before programming a controller, one should make sure that no overflow occurs during all stages of calculations. To achieve this, normalization should be made [6/9]. If direct form [2] is used to realize Eqs. 3 and 4, no normalization is required, because they are already normalized.

Prog. 2 shows a proper DGCSSL routine

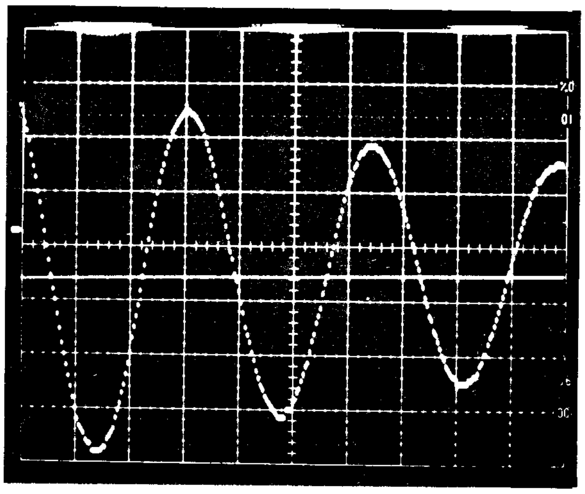


Figure 4. ***Simulation result for the case when no controller is included.***

DC gain of the plant=30

Resolutions: ***A/D 8 bits, rounded***

D/A 8 bits, rounded

Computer 16 bits, rounded

for simulation of the proposed digital control system, when controller (i) is used. The controller block diagram is, also, included to clarify the listing. By careful examination of this routine, one may find elegant application of counters C1 and C2, in conjunction with the LOOP instruction to take many samples from the input signal and the output of the plant, during each sampling period of the controller. In the above routine the controller calculations are performed once after every five passes of the main loop (MLOOP), but the desired samples are stored in the tables during each pass. So, in order to maintain the controller period of 0.3sec., the simulator sampling period should be chosen to be 0.06sec. (0.3/5).

In some situations, one may be interested to store the samples in the tables one out of N. This occurs when the rate of variation of the desired parameter is very low in comparison with the controller sampling period. This can be easily achieved if on line 15, 1 is moved into C1, and on line 24, N is moved into C2.

In Prog. 2 counter C3 in conjunction with the DELY instruction is used to facilitate simulation of a calculation delay. In the present routine a calculation delay of 0.13sec. is inserted. If lines 16, 17 and 18 are cancelled, the calculation delay will be equal to the execution time of instructions on lines 07 to 15 which for the present example it is very small in comparison with the controller sampling period.

The following amendments are needed to replace controller (i) by (ii), or to program the controller (i) using other structures: (a) Drawing a proper block diagram for the new controller. Normalization should be made if necessary. (b) Assignment of proper accumulators to different nodes of the proposed block diagram. (c) Replacement of the controller routine (lines 06 to 14 of Prog. 2) by the new controller routine. Thus, amendment of the Prog. 2 to suit different controllers is straight forward. Other interesting amendments can be made to get variety of simulation results, which may be instructive and interesting for design engineers. For example, due to the finite wordlength of calculations odd effect may occur in some situations. In these situations, observation of some signals from proper points of the plant may be a key to reveal the most important reasons for that (those) effect (s).

Figures 5-10 show different simulation results of the considered plant. The presented results and the following discussions are given to indicate the proposed simulator capabilities, and its efficiency when it is used as a design tool. Although, the discussions are resulted in a few interesting conclusions in relation to the implementation of digital control systems, our main aim has not been so. In fact, it is common for any design engineer to get some conclusions when he is implementing or simulating a real system. The rate of deduction of new results and conclusions is quite high when a control systems engineer is working with the proposed simulator. It is, both, because of the simplicity of the system, and existance of special problems (odd effects) in digital control systems.

Figure 5 shows the step response of the proposed plant when the designed lag controller is used to overcome the system instability. Although, application of controller (i) has resulted in a stable system, we find a sort of limit cycle in the system output. To explore the reason for this effect, the output of the controller is, also, shown in Figure 5. Careful examination of the presented results reveal that in the steady state, because of the high DC gain of the plant, the output of the controller should be kept at a low level; namely in the range of one or two quantization levels. One quantization level is too low, and two levels is high. To overcome this problem we should use a more accurate D/A convertor. However, in order to keep system price at a low level we may not be interested to replace the present 8-bits D/A by a more accurate one. A possible solution for this problem is to divide the plant and compensate that, by adding more gain in the controller part of the system. For example, we may have a plant with a DC gain of 30/8, and add a factor of 8 at the output of the digital controller. We should be careful that the latter gain, which is in digital form, must be accompanied by a hard limiter to prevent numerical overflow. The simulation result for this case is shown in Figure 6.

Figure 7 shows a simulation result of the

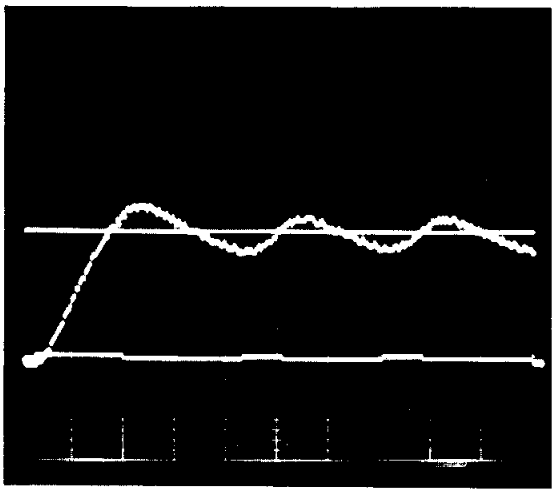


Figure 5. Simulation result for controller (i) and a step input. The controller output is also included (the lower signal).
DC gain of the plant = 30
DC gain of the controller = 1
Resolutions: A/D 8 bits, rounded
D/A 8 bits, rounded
Computer 16 bits, rounded

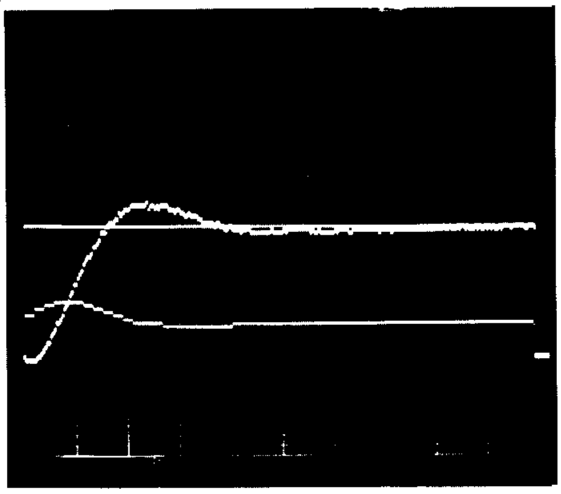


Figure 6. Simulation result for the controller (i) and a step input. The controller output is also included (the lower signal).
DC gain of the plant = 30/8
DC gain of the controller = 8
(To prevent numerical overflow a hard limiter is added at the output of the controller)
Resolutions: A/D 8 bits, rounded
D/A 8 bits, rounded
Computer 16 bits, rounded

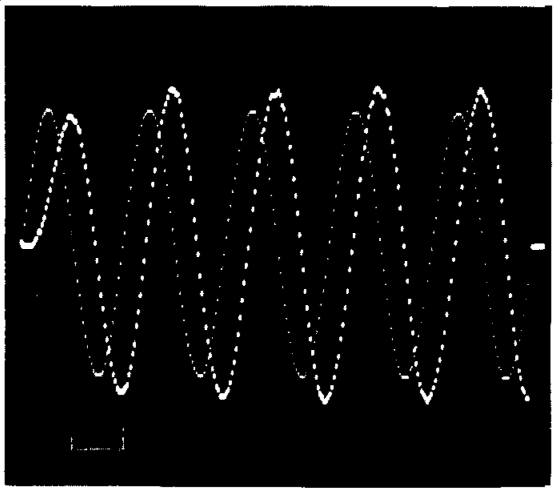


Figure 7. Simulation result for controllr (i) and a sine wave input.
DC gain of the plant = 30/8
DC gain of the controller = 8
Resolutions: A/D 8 bits, rounded
D/A 8 bits, rounded
Computer 16 bits, rounded

system for a sine wave input. From this result one can measure the amplitude and phase response of the feedback system. By changing the input frequency it is not difficult to get more results and to draw the complete frequency response of the system. It is, also, possible to use the user definable instructions to write a more complex program to get the system frequency response in one run.

Figure 8 shows the effect of finite word-length on the system step response. The existance of a deadband zone can be seen in this figure.

The results shown in Figures 9 and 10 correspond to the controller (ii), i.e. the lag-lead controller. In Figure 9 we see, as expected, the shorter rise time of the system step resonse, compared with the result of the lag controller.

Figure 10 shows the system ramp response when parallel form is used to realize the present controller. In this figure we can see the result of occurance of over-flow in the controller routine.

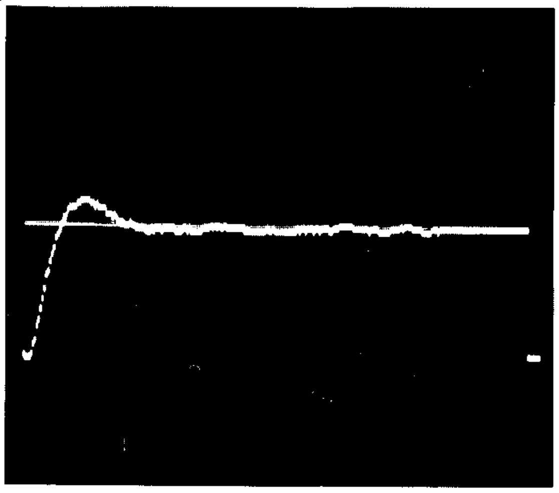


Figure 9. Simulation result for controller (ii) and a step input.

DC gain of the plant = 30/8

DC gain of the controller = 8

Resolutions: A/D 8 bits, rounded

D/A 8 bits, rounded

Computer 16 bits, rounded

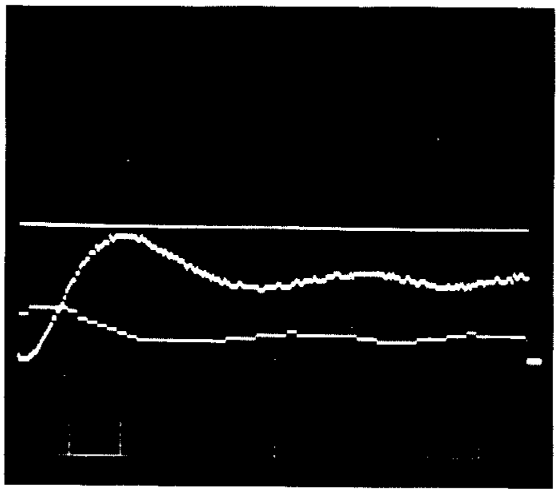


Figure 8. Simulation result for controller (i) and a step input. The controller output is also included (the lower signal).

DC gain of the plant = 30/8

DC gain of the controller = 8

Resolutions: A/D 8 bits, rounded

D/A 8 bits, rounded

Computer 12 bits, truncated

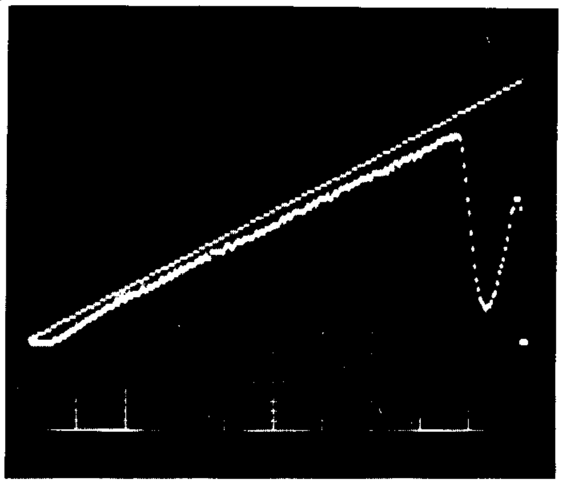


Figure 10. Simulation result for controller (i) and a ramp input. No normalization is used to indicate the effect of occurance of overflow.

DC gain of the plant = 30/8

DC gain of the controller = 8

Resolutions: A/D 8 bits, rounded

D/A 8 bits, rounded

Computer 16 bits, rounded

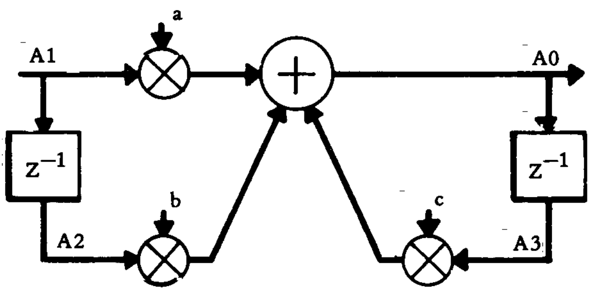
Program 1 **Simulation program for a system without compensator.**

```
00 PROG1:   MOVC C0, 256       ; No. of Samples to be taken
            ;
            ; The following instruction is used to program CTC.
            ; After execution of this instruction interrupts occur every
            ; T sec.
01          STRT
02 MLOOP:   INPT  A0           ; Take a new input sample
03          ATDC  Cho, A1      ; Take a sample from output of the plant
            ;
            ; The following three instructions are served to get error
            ; sample, and send it to the plant through the D/A convertor.
            ;
04          MOVA A2, A0
05          SUBA  A2, A1
06          DTAC  A2
            ;
07          MOVT T0, A0        ; Save the input sample in Table T0
08          MOVT T1, A1        ; Save the plant output sample in Table T1
09          ITBP               ; Increment the tables pointer
            ;
10          WAIT               ; Wait for the next interrupt
11          LOOP  C0, MLOOP    ; Decrement C0 and jump to the start of
            ;                    the main loop (MLOOP) if C0 = 0.
12          STOP               ; Stop interrupts
            ;
            ; The following instructions are served to display the
            ; simulation results on an oscilloscope CRT.
            ;
13 DLOOP:   DISP  T0, 0, 255
14          DISP  T1, 0, 255
15          JUMP DLOOP
```

Program 2 **A program for simulation of digital control systems with a first order controller. The controller is implemented in direct form.**

```
00  PROG2:   MOVC    C0, 256        ;
01           MOVC    C1, 1          ; This counter controls the controller sampling
                                    ; priod.
02           MOVC    C2, 1          ; This counter controls the simulator sampling
                                    ; period.
03           STRT
04  MLOOP:   INPT    A4
```

```
                LOOP    C1, MLOOP1    ; A jump will occur over the following
                                      ; instructions if this pass does not
                                      ; correspond to a controller sampling.
                ATDC    Ch0, A0       ; Taking a sample from the plant output
                MOVA    A1, A4        ; and calculation of the error
                SUBA    A1, A0        ; sample e(n).
                ;
                ; **** Calculation of the controller output ****
                ;
                SUBA    A0, A0
                MULA    A0, A1, a
                MULA    A0, A2, b
                MULA    A0, A3, c
                MOVA    A2, A1
                MOVA    A3, A0
                ;
                MOVC    C1, 5
                ;
                ; The following three instructions are served to insert
                ; a calculation delay of 0.13 sec.
                ;
                MOVC    C3, 3
      MLOOP1:   LOOP    C3, MLOOP2
                DELY    100
                ;
                DTAC    A0            ; Transmitting the controller output
      MLOOP2:   LOOP    C2, MLOOP3
                ATDC    Ch0, T0       ; Saving the plant output sample in table T0
                MOVT    T1, A4        ; Saving the input signal sample in table T1
                ITBP
                MOVC    C2, 1
      MLOOP3:   WAIT
                LOOP    C0, MLOOP
                STOP
                ;
                ;**** Display loop ****
                ;
      DLOOP:    DISP    T0, 0, 255
                DISP    T1, 0, 255
                JUMP    DLOOP
```



CONCLUSION

A hybrid simulator based on a Z-80 microcomputer, for simulation of digital control systems, was proposed. To overcome the difficulty of assembly language programming, and to have a fast system, a special purpose language was proposed. The latter which was called digital control systems simulator language (DGCSSL), was designed so that any controller, once drawn in block diagram form, could be readily programmed. In the DGCSSL the wordlengths of D/A, A/D and computer could be programmed. The designed simulator could be used either by engineers as a design tool, or by students and those who are learning digital control systems. as an instructive aid.

ACKNOWLEDGMENT

The author wishes to thank Dr. V. Tahani for many helpful discussions and comments upon the drafts of this paper.

REFERENCES

1. B. C. Kuo.: *"Digital control systems"*, Holt-Saunders, 1980.
2. P. KATZ.: *"Digital control using microprocessors"*, Prentice-Hall, 1981.
3. C. L. PHILLIPS, H. T. NAGLE, JR.: *"Digital control system analysis and design"*, Prentice-Hall Inc., 1981.
4. N. J. KRIKELIS, and S. D. FASSOIS.: *"Microprocessor Imple mentation of PID Controllers and Lead-Lag compensators"*, IEEE Trans. Ind. Electron., Vol. IE-31, pp. 79–85, Feb. 1984.
5. A. K. LIN, and W. W. KOEPSEL.: *"A Microprocessor Speed Control System"*, IEEE Trans. Ind. Electron. Cont. Inst., Vol. IECI–24, pp. 241–247, Aug. 1977.
6. M. E. AHMED, and P. R. BELANGER.: *"Scaling and Rounoff Noise in Fixed-Point Implementation of Control Algorithms"*, Trans. Ind. Electron., Vol. IE–31, pp. 228–234, Aug. 1984.
7. ———————.: *"Limit Cycles in Fixed-Point Implementation of Control Algorithms"*, IEEE Trans. Ind. Electron., Vol. IE–31, pp. 235–242, Aug. 1984.
8. V. C. JASWA.: *"Quantization Noise and Limit Cycling in Digital Closed-Loop Systems"*, IEEE Trans. Ind. Electron., Vol. IE–31, pp. 149–151, May 1984.
9. L. R. RABINER, and B. GOLD.: *"Theory and Application of Digital Signal Processing"*, Prentince-Hall, 1975.
10. T. HIGUCHI, T. SAITO, and A. KANOMATA.: *"A Microprocessor Based Digital Filter Progreammed in a Block Diagram Language"*, IEEE Trans. Ind. Electron. Cont. Inst., Vol. IECI–24, pp. 231–234, Aug. 1977.
11. *"MT-80Z user manual"*, Multi-Corporation International Limited, London, England.