Simulation of an Information Channel on the IBM 704 Computer*

E. G. NEWMAN† AND L. O. NIPPE†

INTRODUCTION

EXPRESSIONS for the probabilities of multiple error patterns in symbols consisting of N bits are needed to evaluate the effectiveness of applying error correcting codes to these information channels. For example, if a Hamming[1] single error correcting code is applied to the information, the probabilities of all errors greater than one should be established. This will indicate the number of times the single error correcting code has failed.

In addition, if the effects of channel asymmetry and regional error dependence on the probabilities of multiple errors are known, this information can be used to develop channel characteristics leading to the minimum number of errors.

Expressions for error probabilities have been obtained for a binary symmetric channel[2] and for an asymmetric channel [see (2) or (6) in the Appendix] when errors are assumed to be independent. Attempts to introduce error dependence were made by using relationships between stochastic processes and familiar network concepts.[3,4] All results, however, for even relatively simple cases became too unwieldy to be useful. Therefore, a simulation program which gives approximate values for the desired error probabilities was developed.

Many practical information channels exist which exhibit both error dependence and asymmetry. An example of such a channel is a magnetic tape channel. A symbol of N bits can be recorded on N parallel tracks (Fig. 1). The error dependence is primarily the result of defects extending over certain regions of the tape which influence the correct transmission of successive bits in a particular track or in a number of adjacent tracks.

A single defect may produce errors on a single track only (Defect I, Fig. 1), or larger defects may cause errors in a number of adjacent tracks (Defect II, Fig. 1). These large defects produce multiple errors for all consecutive N-bit symbols which fall into this defective region. Multiple errors can also be produced by the simultaneous occurrence of different defects, each affecting the information in a particular bit of an N-bit symbol (Defects I and III, Fig. 1).

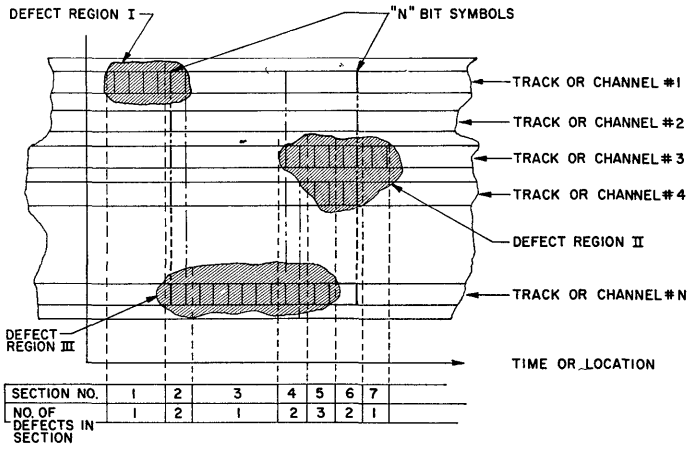


Fig. 1—Effects of regional defects on the generation of multiple errors in an N-bit symbol.

Depending on the type of recording used, certain types of defects may cause errors in the transmission of ones without affecting the transmission of zeros. Other types of defects or noise bursts affect the transmission of zeros only. Some defects can produce errors in both ones and zeros. These phenomena lead to channel asymmetry. The degree of this asymmetry is governed by the distribution of the various defect types.

For example, for a modified nonreturn to zero type of recording (Fig. 2), a one is recorded by changing the magnetization of the tape from one saturation level to the other. A zero leaves the magnetic tape at a previously established saturation level. As the read head senses only a change in the flux linking it, only ones produce signals.

Tape defects in the form of high spots in the surface of the magnetic coating or loose particles of oxide material produce a head-to-tape separation, which results in a loss of signal. This affects the transmission of ones only. The characteristic loss in signal amplitude is shown in Fig. 2. This phenomenon is not unlike the amplitude "fading" of radio signals and may, at high recording densities, produce a number of consecutive errors.

Other tape defects, such as pin holes in the magnetic coating, can produce errors in the transmission of zeros (Fig. 3). At the boundaries of the hole in the magnetic coating, a flux change will be sensed by the read head as it moves from a region free of magnetic particles to one where magnetized magnetic particles exist. This change of flux could be sensed as a one in place of a zero. In addition, amplifier noise can occasionally exceed a certain clipping level. If this noise peak occurs at the

* This work has been supported by the U. S. Dept. of Defense.
† IBM Product Dev. Lab., Poughkeepsie, N. Y.
[1] R. W. Hamming, "Error detecting and error correcting codes," *Bell Sys. Tech. J.*, vol. 29, pp. 147–160; April, 1950.
[2] L. Brillouin, "Science and Information Theory," Academic Press, Inc., New York, N. Y., ch. 6, pp. 62–70; 1956.
[3] W. H. Huggins, "Signal flow graphs and random signals," PROC. IRE, vol. 45, pp. 74–86; January, 1957.
[4] S. J. Mason, "Feedback theory—some properties of signal flow graphs," PROC. IRE, vol. 41, pp. 1144–1156; September, 1953.

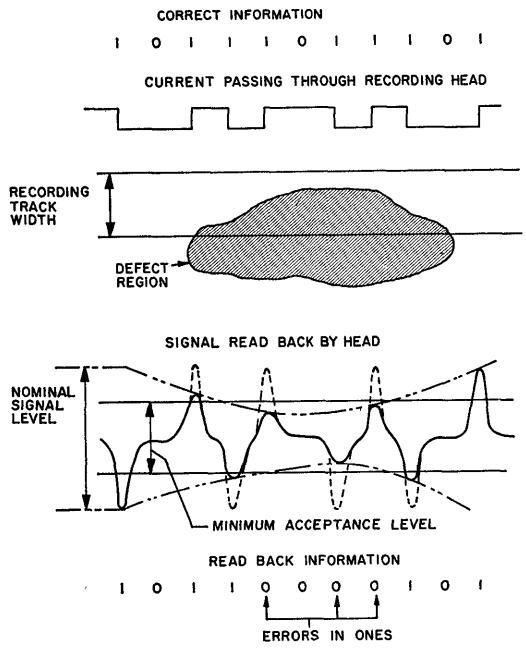


Fig. 2—Effect of tape defect causing head-to-tape separation on signal amplitude (modified nonreturn to zero recording).

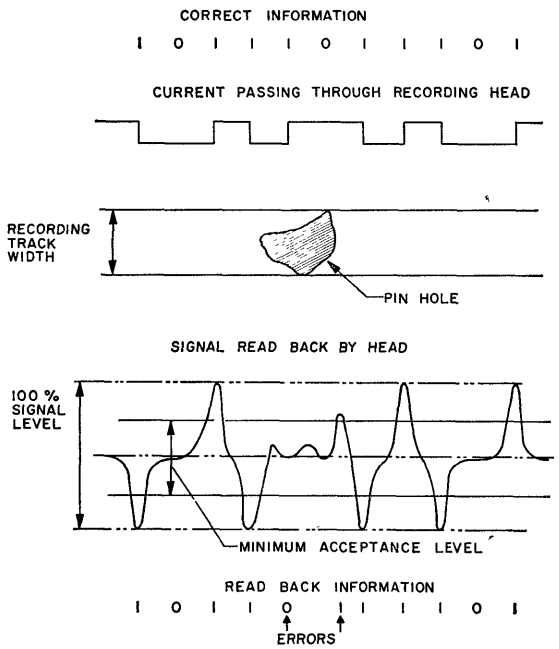


Fig. 3—Effect of defect in magnetic oxide (pin hole) on signal amplitude (modified nonreturn to zero recording).

time when a zero should be read, it may be sensed incorrectly as a one.

If the hole is large, ones recorded in this region will, of course, not be sensed. A hole in the magnetic oxide or a nonmagnetic particle in the magnetic coating can, therefore, produce errors in both ones and zeros.

These are some typical examples of the error-producing mechanisms which cause the complex characteristics of the magnetic tape channel.

Channel Simulation

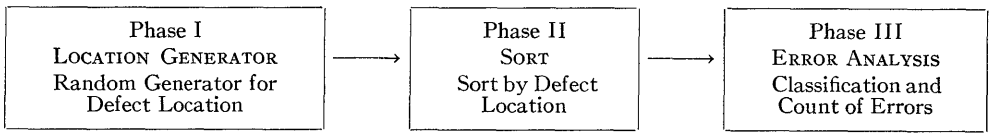
Because of the difficulty of obtaining analytic expressions for the probabilities of multiple error patterns in terms of the complex characteristics of a tape channel, simulation means had to be developed.

The simulation procedure can best be described by comparing it to the manufacturing of tape. The defects in the tape, produced during the manufacturing process, can be considered to be distributed in a random manner over the length of the tape. The program simulates this process by assigning random locations to each of the defect regions, as they are read from a list of inputs. This list gives the number and class of all the various defects which must be placed on a particular length of tape. This defect listing can usually be prepared on the basis of error statistics obtained from tests involving a single track or at the most a few adjacent tracks. A theoretical description of the channel characteristics can also be used. The program is completely flexible, that is, tape of any specifications can be "manufactured."

Random information could now be "recorded" over the whole length of the hypothetical tape and errors introduced in accordance with the defect type. The program, however, concerns itself only with errors. Thus, information is placed only into the defective regions and the resulting errors are classified and counted. This results in a considerable saving of computer time.

The 704 Program

The 704 Information Channel Simulator Program (Fig. 4) might best be considered by breaking the program into three major phases as indicated below:



Phase I might be called the "tape manufacturing" phase because it is responsible for the actual placement of defects on the hypothetical tape. All defects are assumed to be placed on the tape simultaneously and previous to the analysis phase.

Since the defect locations are generated randomly, it is next necessary to sort (Phase II) the defects by location so that they may later be processed in sequence.

Phase III performs the random generation of infor-

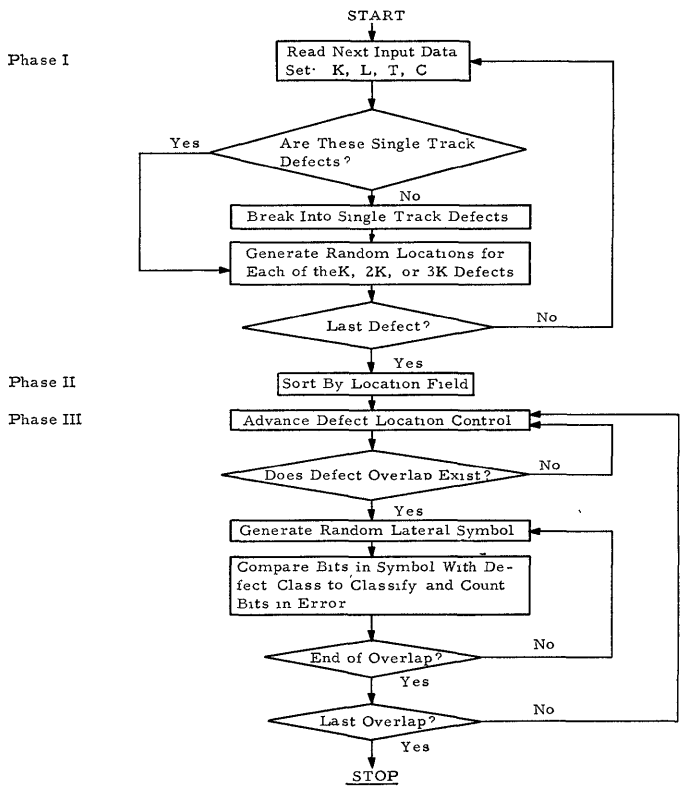


Fig. 4—General flow diagram.

mation to be written on the tape, the analysis of the influences of each defect on this information, and the classification and counting of resulting errors.

Input Data

Before discussing any of the three phases of the program in detail, it is necessary to have an accurate picture of the input data, which are prepared on the basis of actual test data or from a theoretical description of the channel characteristics.

To formulate the input data for the program the following statistical information is needed:

1) The *a priori* probabilities that defects of a particular length will occur on a track. These defects can produce errors in ones or zeros or both on this track only.
2) The *a priori* probabilities that "centers" of large defects of a particular size will occur on a particular track. These defects can produce errors in ones or zeros or both in a region which encompasses a number of adjacent tracks.

For program use, the above information is converted into a set of values K, L, T, and C, where

K = the number of defects per track of a certain type to be placed on the hypothetical tape,
L = the length of the defect in bits,
T = the track number of the defect center, and
C = the defect class, and indicates whether the defect involves errors in ones, zeros, or both, and whether one, two, or three adjacent tracks are involved. For the defects which produce errors in more than one track, the same defect length per track was assumed.

The above set of values is defined as an input data set. A different set may be used for each track.

Phase I—Location Generator

Each input data set read per track represents K defect areas. In order to assign a random longitudinal location X to each, a random number modulo M is generated. The pseudo random number generator program used here is PE RAND. It produces a 35-bit random number by multiplying two odd, 35-bit numbers, selected from a group of ten such numbers, to produce a 70-bit product. The center 35 bits of this product are used as the random result and may be divided by a previously specified number, M, to make the result modulo M. This generator has been thoroughly tested. The probability that any bit of the resulting random number is a one lies between 0.45 and 0.55. This was considered entirely satisfactory for this application.

This defect location just generated is stored along with the appropriate length, track, and class. Any given defect may involve one or more adjacent tracks as specified by its class. Arbitrarily, the program was written to handle defects extending over no more than three adjacent tracks. The program converts these adjacent track defects to individual track defects of equal length and assigns the same location to each.

Phase II—Sort

After a random location has been generated for each defect, these defects are sorted by location fields. There are two general sort programs available for the 704 through the SHARE organization. Either may be used, or the sorting may be done on another machine.

Phase III—Error Analysis

It can be seen from the flow diagram of Fig. 4 that there are three basic operations other than decision making that must be performed by this phase. They are

1) Comparison of defect locations,
2) Random generation of information in the form of N-bit symbols, and
3) Classification and counting of errors within the affected N-bit symbols.

As each defect is read from the sorted list, its location must be compared with the location of other defects to see if any of the defects overlap. When defects on different tracks overlap, multiple errors may result. When defects on the same track overlap, they are joined into one defect.

When the number of overlapping defects changes, a new "section" is formed (Fig. 1). The program keeps track of the number of overlaps involved at any instant.

The number of bits involved in these overlapping defects represents the maximum number of errors which can occur in the symbols in the overlap. For example, consider Section 2 in Fig. 1. This particular section is two bits long and involves Defects I and II. Thus, no more than four bits may be in error in this particular overlap, two bits in each of two N-bit symbols. The program provides three counters for double, triple, and higher order overlaps.

Next, the program must generate random information for each N-bit symbol in the overlap. Considering each bit individually, it will be in error if any one of the following three conditions exists:

1) The bit is a one and the defect produces errors in ones,
2) The bit is a zero and the defect produces errors in zeros,
3) The defect produces errors in ones and zeros.

A test for errors is now made on the basis of these three conditions. The previously described PE RAND program is used to generate the random information.

Each bit in the affected N-bit symbol is tested, and, if an error has occurred, the appropriate error counter is updated. The program provides counters for double and triple errors

Machine Time on the IBM 704 Computer

Table I indicates approximate machine times per phase for typical runs involving 3500 and 10,000 defect regions, respectively. It shows that the time required to run Phases I and III is linearly related to the number of defect regions involved, while the time for Phase II (Sort) increases more rapidly with the number of defects. This indicates that a larger number of runs involving fewer defects will give shorter over-all machine times.

TABLE I
704 Machine Time in Minutes per Run

Number of Defect Regions	Phase I	Phase II	Phase III	Total
3500	3	4	4	11
10,000	9	22	12	43

Results of Simulation Program

The program was tested by using it to simulate a symmetric binary channel with no error dependence. For this channel, exact error probabilities can be computed [see (1) in the Appendix]. Results of these computations were compared with results of the simulation program. A few of these results are shown in Fig. 5 for N equal to 24, 12, and 8. This channel had an *a priori* error probability per track of $p=2\times10^{-3}$. The average error for the double error probability, as compared with the theoretical results, varied in these examples from 0.76 to 1.77 per cent.

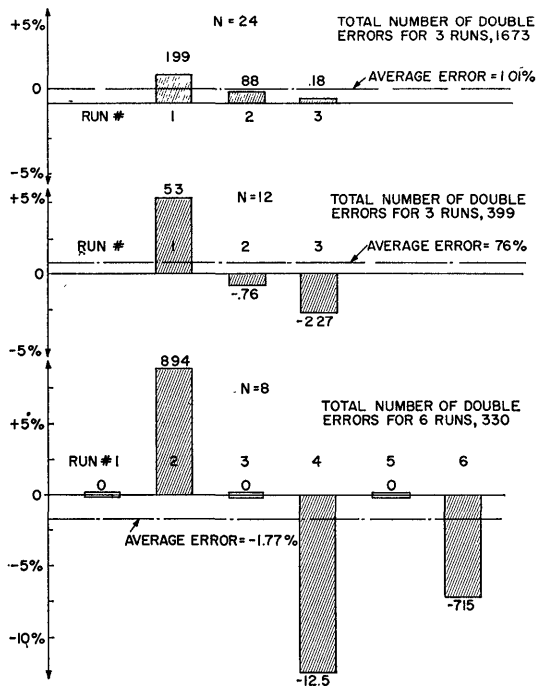


Fig. 5—Error in per cent for double error probability. Symmetric channel (defect length = 1); *a priori* error probability, $p=2\times10^{-3}$.

As may be expected, the accuracy of the results depends on how many runs have been made. The number of runs required will be different for each problem and will depend on how long it takes to accumulate a significant count for the various error patterns of interest. The theoretical limit to the final accuracy of the results of the simulation program is set by the inherent inaccuracy of the pseudo random number generator and on the accuracy with which the mathematical model describes the actual information channel.

The simulation program has been designed for information channels with rather complex characteristics. An example of such a channel is a Z channel. In a Z channel, one of the binary symbols is always transmitted correctly; thus, it is a channel with the greatest possible degree of asymmetry. For the example selected, the probabilities of occurrence of defects of various lengths are shown in Fig. 6. This hypothetical distribution was developed to study the effect of channel asymmetry and error dependence on error probabilities. In this distribution, the defects contain as many single bits per track as in the previous example for the symmetric channel.

For this Z channel, the approximate double error probabilities for $N=24$, 12, and 8 are shown in Fig. 7. The results for this example indicate that good approximations for multiple error probabilities can be obtained by assuming that each long defect region is split up into as many separate regions one bit long, as there are bits in the original defect region.

The multiple error probabilities for this Z channel ($p_1=2\times10^{-3}$) can be computed, using (3) or (8) in the Appendix.

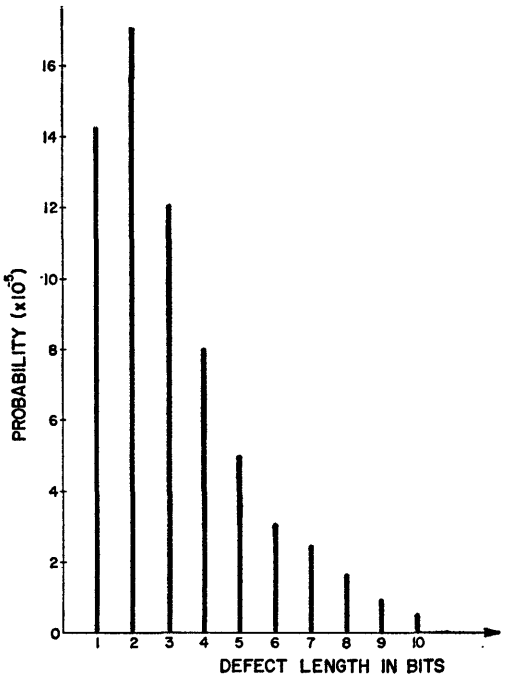


Fig. 6—Hypothetical probability of occurrence of defects of various lengths.

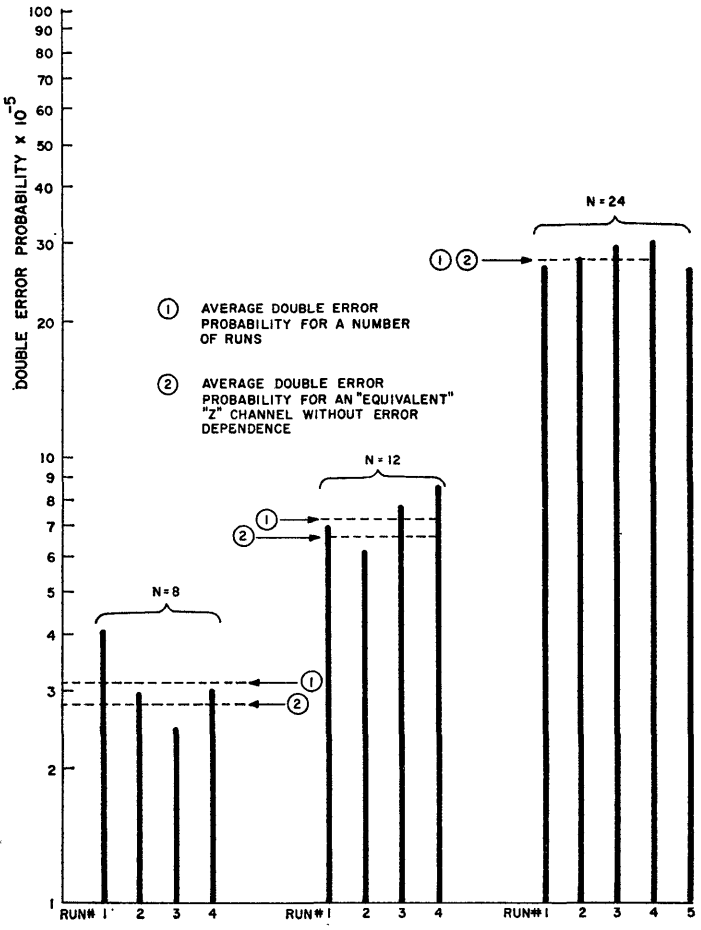


Fig. 7—Double error probability for a Z channel for error dependence shown in Fig. 5.

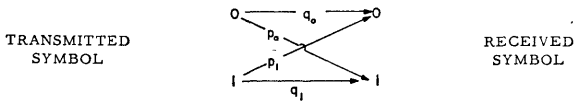


Fig. 8.

Summary

The results of the simulation program indicate that good approximations to the probabilities of multiple-error patterns in symbols consisting of N bits can be obtained.

The computer time required for the simulation is quite reasonable and compares favorably with the costs for the testing of numerous complete versions of a system. The simulation procedure is, therefore, particularly useful during the early stages of development.

The program, because of its flexibility, also lends itself to purely theoretical investigations of error statistics.

Appendix

Expressions for Error Probabilities for an Asymmetric Binary Channel with No Error Dependence

The binary asymmetric channel can be represented by Fig. 8 where

q_0 = the probability of a transmitted zero being received as a zero, $q_0 = 1 - p_0$,
q_1 = the probability of a transmitted one being received as a one, $q_1 = 1 - p_1$,
p_0 = the probability of a transmitted zero being received as a one,
p_1 = the probability of a transmitted one being received as a zero.

Let there be N bits in a character, of which r are ones and $(N-r)$ are zeros. The probability $p_N(z)$ or z errors in a character consisting of N bits can be expressed for a symmetric channel[1] where $p_1 = p_0$.

$$p_N'(z) = \binom{N}{z} p_1^z q_1^{(N-z)}. \tag{1}$$

For an asymmetric channel, if all N-bit symbols are assumed to be transmitted with equal probability,

$$p_N''(z) = \frac{1}{2^N}\left|\sum_{r=0}^{r=N}\binom{N}{r}\left[\sum_{x=0}^{x=z}\left\{\binom{r}{x} p_1^x q_1^{(r-x)}\right\}\right.\right.$$
$$\left.\left. \cdot \left\{\binom{N-r}{z-x} p_0^{(z-x)} q_0^{N-1-(z-x)}\right\}\right]\right| \tag{2}$$

and, for a Z channel, $p_0 = 0$ and $q_0 = 1$.

$$p_N'''(z) = \frac{1}{2^N}\left\{\sum_{r=z}^{r=N}\binom{N}{r}\binom{r}{z} p_1^z q_1^{r-z}\right\} \tag{3}$$

where

$$\binom{N}{x} \equiv \frac{N!}{x!(N-x)!}. \quad (4)$$

Simplifications can be made in (1)–(3), if systems of inherently high reliability are considered. That is, if

$$p_1 \ll 1, \qquad p_0 \ll 1. \quad (5)$$

Expanding $(1-p)^k$ in terms of a power series and retaining only the first term, (1)–(3), respectively, become

$$p_N'(z) = \binom{N}{z} p_1^z \quad (6)$$

and, if $p_1 = \alpha p_0$,

$$p_N''(z) = \frac{1}{2^N}\left\{\sum_{r=0}^{r=N}\binom{N}{r}\left[\sum_{x=0}^{x=z}\binom{r}{x}\binom{N-r}{z-x}\frac{p_1^z}{\alpha^{z-x}}\right]\right\} \quad (7)$$

$$p_N'''(z) = \frac{1}{2^N}\sum_{r=z}^{r=N}\binom{N}{r}\binom{r}{z}p_1^z. \quad (8)$$

ACKNOWLEDGMENT

The authors would like to thank the U. S. Department of Defense for permitting the publication of this paper. Valuable suggestions by C. L. Christiansen, R. J. Sippel, and P. J. Nelson contributed greatly to this paper.

A Compiler with an Analog-Oriented Input Language

M. L. STEIN†, J. ROSE‡, AND D. B. PARKER‡

INTRODUCTION

ANALOG computation is, for many problems, more convenient than digital computation but lacks the precision obtainable from a digital-computer solution. A compiler has been developed which, for problems involving differential equations expressible in the form

$$\dot{y}_i = f(y_1, y_2, \cdots, y_n), \qquad i = 1, 2, \cdots, n \quad (1)$$

combines to a considerable extent the desirable attributes of both types of computation. The compiler achieves this by deducing from a description of an analog setup diagram the differential equations which the analog computer solves, and then compiling a program for solving the equations.

Two drawbacks are avoided. The differential equations represented by a diagram are deduced with no attempt to simulate the analog computer, and a sophisticated integration procedure is used. Therefore, an accurate and relatively efficient digital-computer program is obtained. Even though the solution is obtained from the differential equations, the identity of the output of each element on the diagram is not lost, so that the results may be visualized more easily as the response to the physical system being studied.

The integration procedure used in the final program is the Gill[1] version of the fourth-order Runge-Kutta method. Since no starting procedure is required, and because the discontinuities introduced by nonlinear analog elements cause no difficulty, this method was adopted.

The compiler is not the first digital computer program with an input language related to differential analyzers; for example, there are the programs DIDAS[2] and DEPI.[3] However, the program which this paper describes differs from other similar programs of which the authors are aware in one or more of the following respects:

1) The input language is closely related to an extensively used electronic analog computer.
2) The user need not provide his own input and output program. The compiler provides a complete program ready to run.
3) Since the final program produced is in machine language, it is efficient in terms of execution time as compared to an interpretive program.
4) Rather than simulating a differential analyzer, the compiler deduces from a setup diagram the differential equations represented by the diagram.

In producing a program to solve the differential equations expressed by an analog setup diagram, the compiler uses a technique of increasing popularity.[4] This technique is the use of another processor as an inter-

† University of Minnesota, Minneapolis, Minn. The work of this author was supported in part by Convair-Astronautics.
‡ Convair (Astronautics) Div., General Dynamics Corp., San Diego, Calif.

[1] S. Gill, "A process for the step-by-step integration of differential equations in automatic digital computing machines," *Proc. Cambridge Phil. Soc.*, vol. 47, pp. 96–108; June 3, 1950.

[2] G. R. Slayton, "DIDAS," presented at the Twelfth Natl. Meeting of the Assoc. for Computing Machinery; June, 1958.

[3] F. H. Lesh and F. R. Curl, "DEPI, An Interpretative Digital-Computer Routine Simulating Differential-Analyzer Operations," Jet Propulsion Lab., California Inst. Tech., Pasadena, Calif., Memo. No. 20-141; March 22, 1957.

[4] *Communications of the Assoc. for Computing Machinery*, vol. 1, no. 7, p. 5; July, 1958.