An Acoustic Compiler for Music and Psychological Stimuli

By MAX V. MATHEWS

(Manuscript received November 3, 1960)

A program for synthesizing music and psychological stimuli on a digital computer is described. The sound is produced by three operations: (a) A compiler generates the programs for a set of instruments. (b) These instruments are "played" by a sequencing program at the command of a sequence of "note" cards which contain information analogous to that given by conventional music notes. (c) The computer output, in the form of numbers on a digital magnetic tape, is converted to audible sound by a digital-to-analog converter, a desampling filter, and a loudspeaker. By virtue of the general nature of the compiling program a great variety of instruments may be produced, and the instrument programs are quite efficient in terms of computer time. The "note" cards are arranged to minimize the effort necessary to specify a composition. Preliminary compositions indicate that exceedingly interesting music and useful psychological stimuli can be generated.

I. INTRODUCTION

General translating devices for rapid conversion of numerical data into a continuous analog signal[1] make it possible for a digital computer to produce interesting and useful sounds, among them music. In this way many of the mechanical and acoustic limitations of conventional instruments and sound sources can be overcome. This paper describes the third in a series of programs written for sound production, which achieves a much greater versatility than its predecessors[2] because it includes a compiler* which writes programs for various sound generators or instruments.

Since many who are interested in the musical aspects of this subject may not be familiar with computers, technical descriptions will be minimized and programming details omitted. In addition, it may be helpful to describe briefly the digital-to-acoustic converter to which the process

* A compiler is a program which writes other programs.

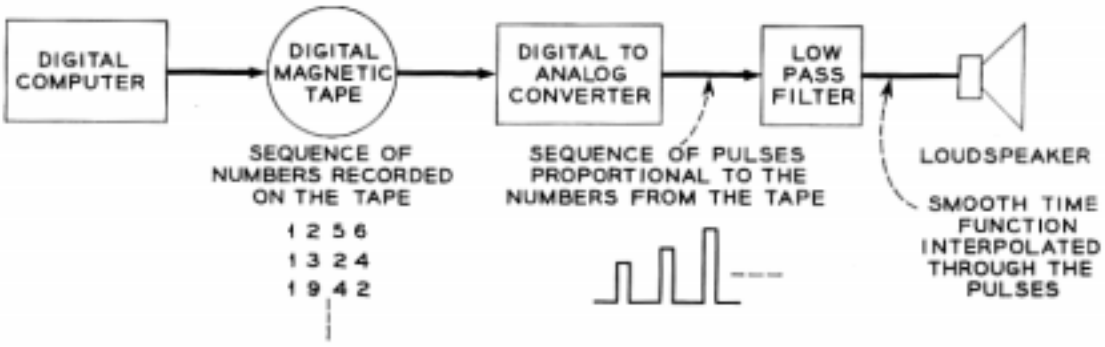


Fig. 1 — Digital-to-acoustic converter.

owes both its existence and generality. The conversion process is schematized on Fig. 1. The computer prepares a magnetic tape on which are written successive digitized samples of the acoustic output. These numbers are then converted by a digital-to-analog converter to pulses whose amplitude is proportional to the numbers. Finally, the pulses are smoothed by a low-pass filter to obtain the excitation for a loudspeaker. The maximum effective sampling rate of the present translator is 20,000 per second, permitting frequencies up to 10,000 cps to be produced.[3] Each sample is reproduced from a four decimal digit integer. Thus, the signal-to-quantizing* noise ratio is greater than 60 db. This ratio is as large as can be conveniently reproduced electronically. Within the limits of this frequency range and this signal-to-noise ratio the converter can theoretically reproduce any sound whatsoever, provided that an appropriate sequence of digital samples can be generated.

II. BASIS OF THE COMPILER

What is the basic objective of this sound generation procedure? It is not simply to produce sounds in the most general way. This generality could be achieved by having the composer list the 20,000 numbers per second which he wished converted to sound. However, such a process is impossibly tedious and, more important, does not effectively control the parameters which determine the psychological impact of the sound on the listener. The basic objective is then to find an economical way of defining and specifying these parameters while automatically supplying the numerical data through which these factors act. In addition, it is a practical necessity to select generating procedures which are economical of computer time.

In order to fulfill these objectives completely a great deal more must

* Quantizing noise is the error introduced by representing a continuous function with a number that can take on only integer values.

be learned about man as a listener. However, the following admittedly incomplete heuristics figured strongly in the design of the compiler:

1. Music can be considered a time sequence of acoustic events which might be called *notes*, although the connotations of this word are grossly inadequate for this usage. Several sequences (voices) are usually added together in all but the simplest pieces.

2. Individual notes are formed from approximately periodic functions. Their most important parameters are period, amplitude, duration, and wave shape. There are, however, notes not fitting this description which are coming into use, for example, those using random noise and those in which the pitch changes greatly over the duration of the note.

3. The ear is sensitive to a number of nuances which must be introduced to obtain interesting timbres. These effects include a wide range of attack and delay characteristics, which strongly affect *timbre*; frequency modulation or *vibrato*; and amplitude modulation or *tremolo*.

The basic form of the generating program is a scheme for producing sequences of sounds on individual instruments, whose outputs can be combined so as to effect several voices. The instruments are formed by combining a set of basic building blocks called *unit generators*, appropriate combinations of which can produce sounds of almost any desired complexity or simplicity. Such an approach has many advantages, the most obvious perhaps being that novel characteristics may be introduced by compiling new instruments. Of equal importance is that composing and computing effort be minimized for simple instruments. The cost of the compiling philosophy is some additional programming and mathematical complexity in forming the instruments and the substantial work (now completed) of writing a compiling program. But this price is small compared to the advantages gained.

The compiling program was greatly simplified by the use of macro instructions, which specify a sequence of computer instructions by means of a single statement. In this way each unit generator can be specified by a single macro statement.

In order to speed the computer operation certain basic functions which specify characteristics such as wave shape, attack, and decay are generated only once and stored in the computer memory, where they serve as references for the unit generators. The functions may be generated by Fortran subprograms.* By utilizing Fortran, a great variety of functions can easily be programmed.

* Fortran is an automatic coding procedure for the IBM 704 and 7090 computers which makes possible the simple generation of most well-known mathematical functions. A subprogram is a subsidiary program.

III. THE MECHANICS OF GENERATION

The first step in producing a musical piece is to punch a set of cards* which specify the instruments in the orchestra. Most of these cards contain a single macro instruction. These instrument cards are then fed into the computer, together with the compiling program, and the computer punches a card deck, which is the music-generating program or *orchestra*.

A sequence of note cards or *score* must now be prepared. These give the parameters such as pitch, duration, and amplitude for the notes which are to be generated. The orchestra, any Fortran subprograms required by it, and the note cards are now inserted in the computer. The numerical samples of the acoustic output are written by the computer on an output magnetic tape. This tape is then converted to a sound via the high-speed data translator[1] and a loudspeaker.

As a convenient alternative, numbers on the output tape may be copied directly from an input tape that is placed on another of the computer tape machines. This tape, for instance, might have been produced by some previous music-making attempt, and this procedure would permit modifications of a composition without regeneration of the entire piece.

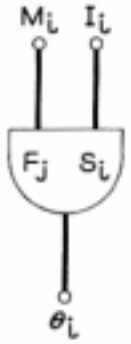


Fig. 2 — Unit generator.

IV. COMPILING THE ORCHESTRA

Various kinds of unit generators are available for forming instruments. However, the one used most frequently generates quasiperiodic functions, as typified by sustained notes. This unit is diagrammed in Fig. 2 and produces samples θ_i according to the relations

* Communication between programmer and computer is carried out by punched cards. Up to 80 digits or alphabetic characters may be impressed on each card and read by either man or machine.

$$\theta_i = M_i \cdot F_j([S_i]_{\text{mod } 512}),$$

$$S_{i+1} = S_i + I_i,$$

where i is the index specifying sample sequence. The index i starts at zero at the beginning of each note and terminates at a value determined by the duration of the note. Sequencing, which controls note duration, will be discussed later. The function F_j is defined for an argument x, where $0 \leqq x < 512$ and is one of 20 functions ($j = 1, 2, \cdots, 20$) which may be stored in the computer memory. As illustrated in Fig. 3, $[S_i]_{\text{mod } 512}$ acts as a triangular scanning function which, if I_i is constant, results in θ_i having a period equal to $512/I_i$ samples and a wave shape determined

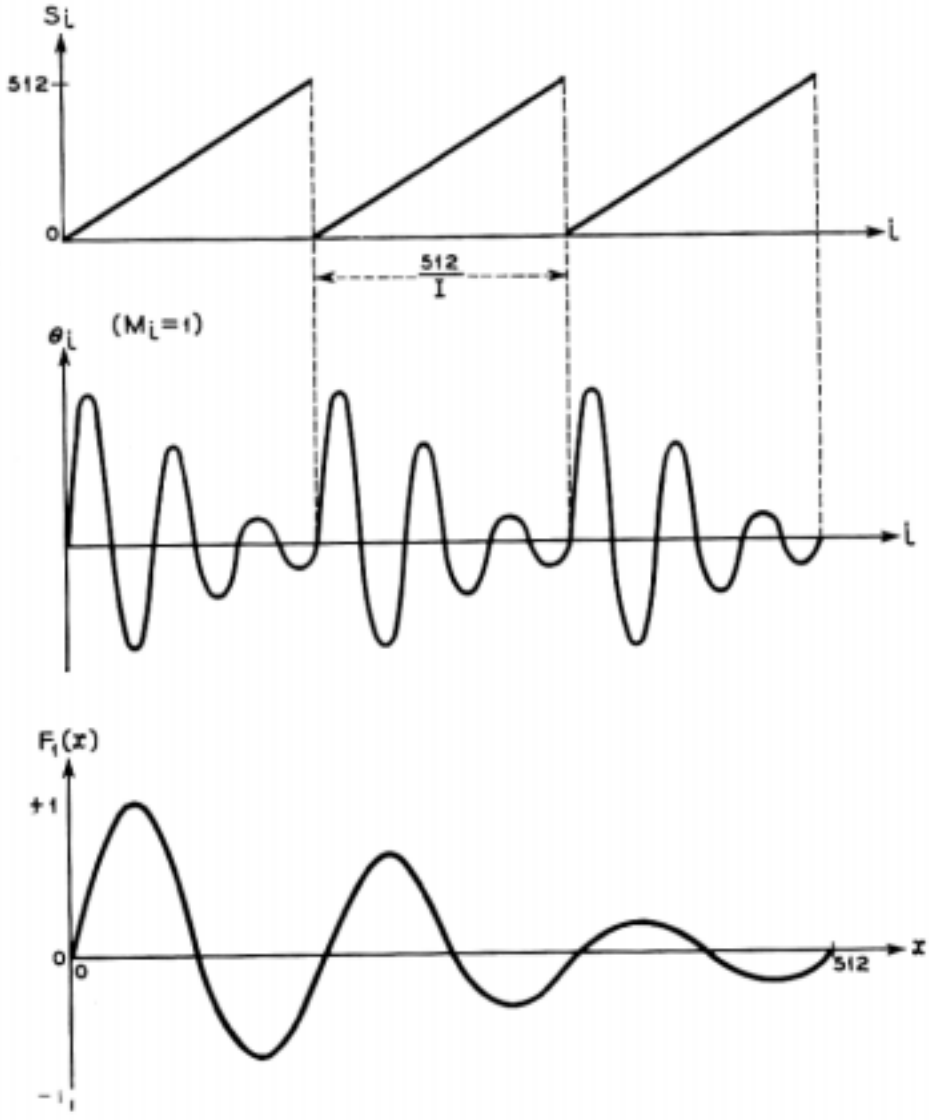


Fig. 3 — Quasiperiodic generation.

by F_j. A varying I_i produces a frequency-modulated output. The generated function is multiplied by M_i, which thus produces amplitude modulation. By this means, attack and decay characteristics can be introduced. To summarize, three functions determine θ_i, these being M_i, I_i, and F_j. In addition, one initial condition, S_0, is involved and is often set to zero at the beginning of each note.

The output θ_i may be added into the final acoustic output. Here the addition defines the process by which outputs of several instruments are combined. On the other hand, θ_i may be used as any input to another unit generator.

The simplest instrument is illustrated in Fig. 4. A periodic note with wave shape determined by F_1, amplitude determined by C1, and frequency by C2 is produced by generator 1U1. The generator 1U2 adds θ into the acoustic output. The attack and decay are instantaneous, which will result in perceptible clicks in the sound.

A more complex instrument with controllable attack and decay may be constructed as in Fig. 5. Here a new generator 2U1 and a function F_2 are added to the structure; 2U1 produces an attack and decay characteristic according to F_2 which amplitude modulates the periodic output of 2U2, while C1 and C2 again specify amplitude and frequency of the note. The new parameter C3 is set so 2U1 generates one period per note. (C3 = 512/duration of note in samples.)

An instrument with attack and decay and vibrato is shown in Fig. 6. Generators 3U1 and 3U2 have been added; 3U2 is an adder whose output is the sum of its inputs. Thus the center frequency of the tone is again specified by C2, the frequency deviation is controlled by C4 and the rate of vibrating (throb rate) by C5, and F_3 determines the wave shape of the frequency variation.

Instruments of even greater versatility can be easily developed by

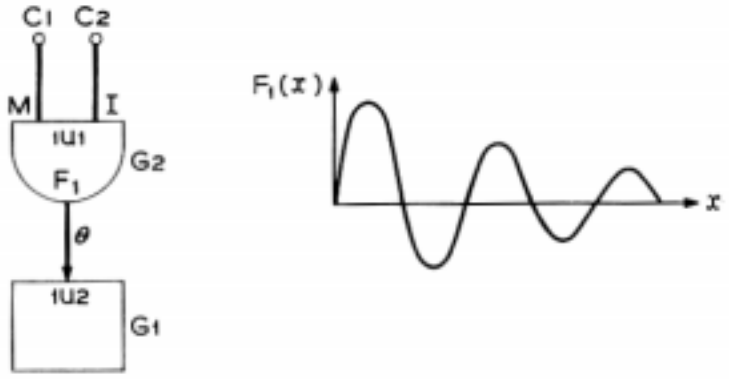


Fig. 4 — Simplest instrument.

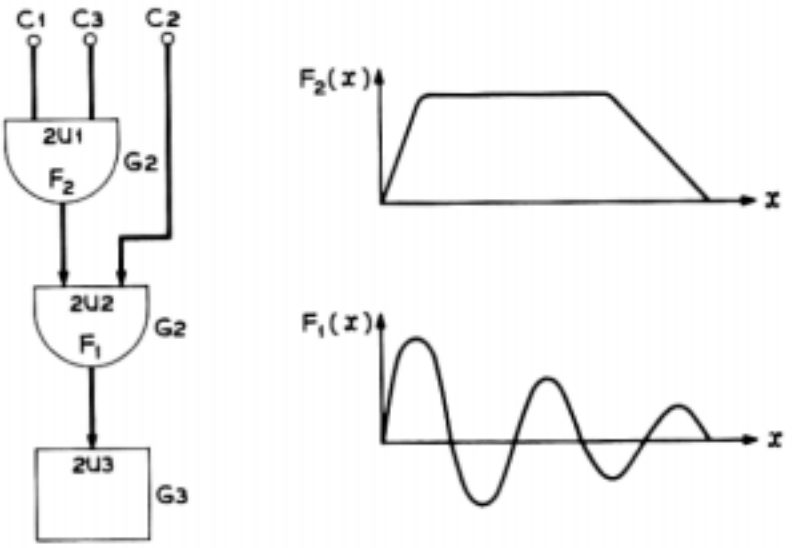


Fig. 5 — Instrument with attack and decay.

putting attacks on the vibrato generator, or adding glissando, or in many other ways. A list and brief description of some of the unit generators which may be used is included in the Appendix.

The punching of the cards from which the instruments are compiled can be illustrated as in Table I, using the cards of I3, the instrument in Fig. 6.

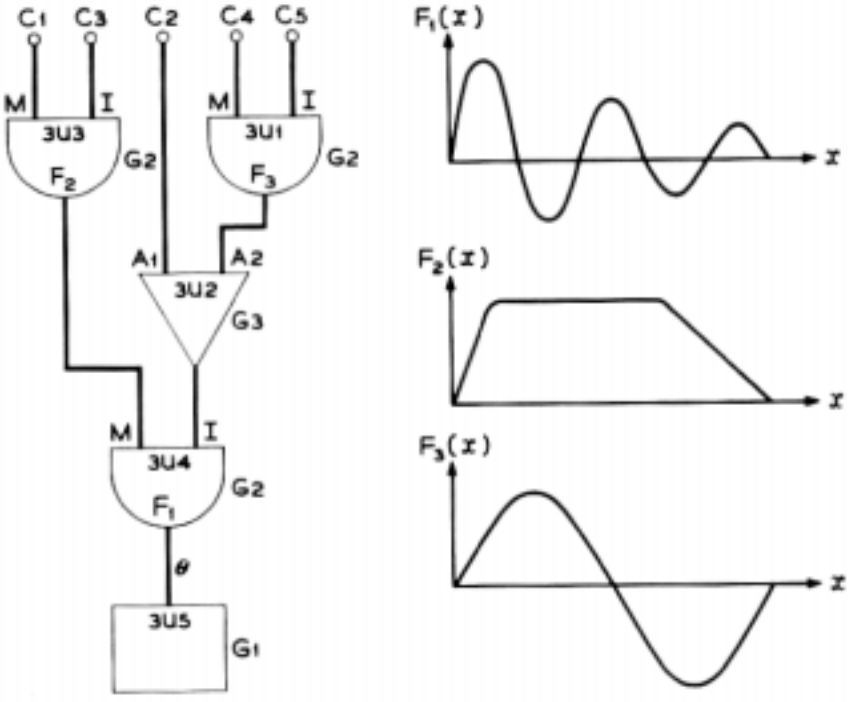


Fig. 6 — Instrument with attack, decay, and vibrato.

TABLE I — DEFINITION OF AN INSTRUMENT

Card Columns		
8 9 10 ...	16 ...	72
M A C	G2, 3U1, F3, (3U2,A2), x, x	
M A C	G3, 3U2, (3U4,I), x, x	
M A C	G2, 3U3, F2, (3U4,M), x, x	
M A C	G2, 3U4, F1, (3U5,θ), x, x	
M A C	G1, 3U5	
M A C	S1, I3, ((3U3,M,C1)(3U3,I,C3) $	
E T C	(3U2,A1,C2)(3U1,M,C4)(3U1,I,C5) $	
E T C	(3U3,S,PO))	

Comments concerning this example:

1. The MAC is a general title designating a macro instruction. Each of the first five macros specifies one unit generator.

2. The Gn in columns 16 and 17 specifies the type of generator, G1 being an output unit, G2 a semiperiodic generator, and G3 an adder.

3. The nUm (3U1 for example) designates the instrument number by n and the number of the unit generator in the instrument by m. Each instrument must have a different number, and the unit generators are numbered sequentially in each instrument.

4. The rest of the unit generator designation varies depending on the type of generator, but in general it specifies where the output of the generator is placed and provides the option of specifying inputs to the generator as constants. Thus (3U2,A2) on the first card shows that the output of 3U1 forms the A2 input of 3U2; F3 indicates that function F_3 is called on by the generator and the terminating x's allow the possibility of providing fixed inputs. For example, if the vibrato frequency were fixed at, say, 8 cps instead of being varied by C5, then a constant equal to $8 \times 512/10{,}000 = 0.4096$ (assuming a 10,000 sample-per-second rate) could be included by the statement:

MAC G2, 3U1, F3, (3U2,A2), x, 0.4096B17*

By specifying all fixed constants in the instrument definitions, the number of parameters which must be written in the score is minimized.

5. In the computer the computation of the sample proceeds from one generator to the next in the order in which they are listed on the cards. Hence, for example, if 3U2 uses as an input the output of 3U1, then 3U2 must be listed after 3U1. In addition, to execute the program, the first

* The B17, appended to the number, specifies the decimal point in the computer memory.

generator in each instrument must be number one (nU1) and the last generator must be of a terminating type, G1.

6. The final macro MAC S1, I3, $\cdots$ compiles instructions which set the parameters C1 through C5 at the beginning of each note. The values of these parameters are obtained from the score in a manner which will be discussed later. The macro can be interpreted in the following way: S1 is the name of the macro which is concerned with setting parameters, I3 refers to instrument 3. Each of the subparentheses sets one parameter; for example, (3U3,M,C1) means set the M input of 3U3 to the value determined by the C1 conversion function. As many subparentheses as desired may be inserted in the macro. If necessary several cards may be used by terminating each card with $ and starting the next card with ETC. The final subparenthesis (3U3,S,PO) uses a special parameter PO which is zero and is used to set the initial value of S in 3U3 to zero. Although it is not done in this instrument, the function to which a given generator refers can also be set by the score. For example, (3U4,F,P6) would cause the function number of 3U4 to be set equal to the sixth parameter on the note cards of the score.

7. The control of the instruments may be summarized by a rule which says that each input or parameter of the unit generators must be either (a) the output of some other generator, or (b) defined as a constant, or (c) set from the score at the beginning of each note. This rule can be used as a check on the correctness of the instrument compilation.*

V. WRITING THE SCORE

After the orchestra program has been compiled it is inserted into the computer, together with any necessary subprograms and the score. The score is also punched on cards, which perform one of four general functions. These are to control the Fortran subprograms for generating the F_j functions, to set the time scale or tempo of the piece, to punctuate the piece with measures and a termination, and to specify the sequence of notes and rests.

Usually each note is specified by one card which gives the duration of the note, the instrument on which it will be played, and all parameters required by the instrument. The card has the format shown in Fig. 7. The OP code, consisting of three alphabetic letters in the first three columns, specifies the function of the card. For example, RST means

* An algorithm to check the correctness could be included in the compiler but has not yet been developed.

rest and a blank (unpunched) OP code indicates a note. The blank code was chosen to save effort, since note cards are by far the most frequently used. The remainder of the card contains space for up to 12 numbers, P1 through P12. P1 gives the instrument number and P2 the duration. The sequence of notes for each instrument is determined by the sequence of note cards. Rests may be inserted between the notes where desired. The note sequences for each instrument are treated separately, so that note cards for different instruments may be interleaved. Thus if two notes are to be sounded together it is only necessary that the sum of the durations of the preceding notes and rests for each of the two instruments be equal.

The control of sequence by note duration introduces the possibility of a duration error in one note causing all subsequent notes in that instrument to be incorrectly positioned with respect to the other instruments. To mitigate this penalty the composition is also divided into

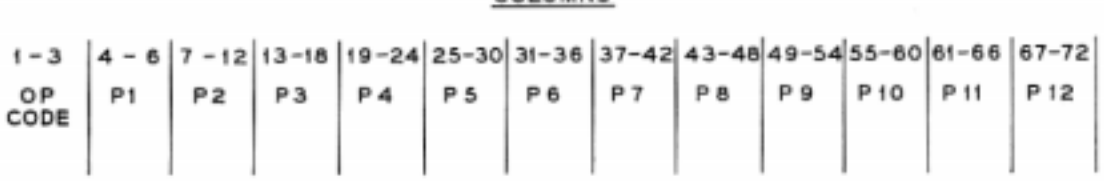


Fig. 7 — Score card.

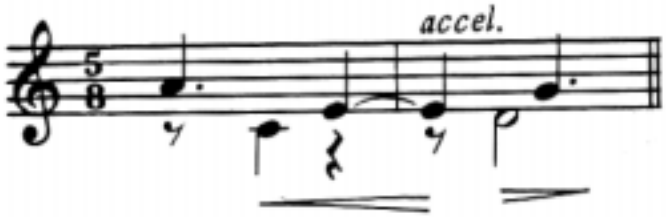
arbitrary units called *measures*. A measure can contain any number of notes up to a maximum of 100. Durations are always computed from the beginning of the current measure, so that a mistake will affect only one measure. In general, the end of the measure marks a break in the notes of all instruments. However, special provision, by means of a slur, has been made for the rare cases where a note must be carried over from the end of one measure to the beginning of the next.

The significance of the numbers P3 through P12 on the cards depends on the particular instrument. The parameters defined by the instrument (C1 through C3 for Fig. 5 example) refer to another set of Fortran subprograms (called CVTO1 through CVTO3). Each of these subprograms can use any or all of the numbers P2 through P12 as arguments of a function to compute one parameter inserted into the instrument. These subprograms can be any of the enormous variety of functions that are specifiable by Fortran; thus exceedingly flexible conversion is possible. The additional complexity of this conversion between score-card parameters and instrument parameters is justified because it allows the composer to write in psychologically meaningful numbers. The burden of

converting from psychological to physical parameters is then carried by the computer.

The types of conversions which are usually employed, as well as the details of a score, are probably best presented by a short but liberally annotated example.

Suppose we wish to generate two measures which in conventional music notation would be written



with instrument 1 (Fig. 4) playing the upper voice and instrument 2 (Fig. 5) playing the lower voice.

Before proceeding we must decide what wave shape function F1 and what attack and decay function F2 we desire and obtain subprograms to generate these. It is unfortunately beyond the scope of this paper to discuss Fortran programming, so for the present let us consider that we have purchased two subprograms, say GEN10 and GEN11, from some competent Fortran programmer. These, when called upon by the score, will produce the damped sinusoid and the attack function illustrated on Fig. 5.

We will also need to obtain from this programmer three conversion functions CVTO1, CVTO2, and CVTO3 with which to set the parameters in our instruments, and we may well choose these functions so as to simplify our task of score writing. The function CVTO1 sets the amplitude of the note, and it is desirable to write the score in a logarithmic rather than linear scale, since the former much more closely approximates the ear's loudness scale. Hence, let us request that

$$\text{CVTO1} = 10^{\text{P3}/20}$$

and use P3 as amplitude control in decibels.

Assuming that the composition is to be played with an even-tempered frequency scale, we can easily obtain a conversion which will let us write frequencies in the form 2.0 through 2.11, where the 2 refers to the octave and the .0 through .11 to the 12 tones within the octave. For this purpose

$$\text{CVTO2} = \frac{512.0}{10{,}000.0} \times 32.70 \times 2^{(\text{I[P5]}+\text{F[P5]}/0.12)},$$

where I[P5] means the integer part of P5, F[P5] means the fractional part of P5, the sampling rate is 10,000 per second, and P5 = 0.0 refers to C three octaves below middle C having a frequency of 32.70 cps. Middle C, for example, would be 3.0 and A above middle C, 3.9.

The remaining conversion function CVTO3 causes the attack generator 2U1 to produce one cycle per note, and thus must be

$$\text{CVTO3} = \frac{512.0}{\text{P2}}.$$

Notice that this function operates on the duration P2 and requires no additional parameters on the note card.

Having obtained the necessary Fortran functions, we can now write the score as shown in Table II.

Comments concerning this example:

1. These two cards cause functions F_1 and F_2 to be generated. P1 determines the generating subroutine to be called and P2 the function to be generated. If desired, P3 through P12 may be used as parameters by the generating routine, although such was not done here.

2. This card sets the time scale so that a P2 of 1 produces 1000 samples, or one-tenth second at a 10,000 sample-per-second rate. This is the duration of an eighth note. The time scale can be reset at any point in the composition and is reduced to 750 for the second measure to accomplish the accelerando.

3. The initial rest for instrument 2 is produced by this card. How-

TABLE II — EXAMPLE OF A COMPOSITION

OP Code	P1	P2	P3	P4	P5	Comments (numbers refer to comments in text)
GEN	10	1				1
GEN	11	2				1
TME			1000			2
RST	2	1				3
	1	3	50		3.9	4
	1	2	55		3.4	4
	2	2	53		3.0	4
MES						5
TME			750			
RST	2	1				6
		4	60		3.2	6
	1	2			3.4	
	1	3			3.7	
MES						
TER						7

ever, no cards are needed for the terminal rest at the end of the first measure for instrument 2. The length of the measure is defined as the maximum sum of the durations of the notes and rests for any instrument. Automatic rests are inserted for instruments not played and between the last note of any instrument and the end of the measure.

4. These cards produce the three notes in the first measure. The instrument numbers and durations are given by P1 and P2. The amplitudes, given in decibels by P3, are arranged to effect a crescendo as requested by the score. The frequencies are specified by P5 in the 12-tone units previously defined.

5. This card terminates the measure.

6. These cards play instrument 2 in the second measure. Notice that the instrument number, P1, is not repeated on the second card. An automatic repeating feature is built into the score-reading program, so that if any parameter is left blank it is repeated from its previous value. Thus, for example, the amplitude of 60 db (P3) is not punched on the last two note cards. This feature is of great value in eliminating a quantity of redundant parameters which otherwise would have to be copied from card to card.

7. This card terminates the composition. It must be preceded by a MES card in order to generate the second measure.

This example provides at least a brief illustration of most of the functions of the score. The two most important omissions are the slur OP code which enables a note to be continued from one measure to the next and the "set" OP code which allows parameters from several cards to set one instrument. These are infrequently used functions and do not justify the space necessary to describe them adequately.

VI. SPEED OF COMPUTION

The time required to synthesize a piece of music depends directly on the number of unit generators involved and the number of samples in the piece. Thus, simple instruments can produce samples rapidly, complex instruments more slowly. For example, on the IBM 7090, the Fig. 4 instrument with two unit generators requires about 0.2 millisecond for each sample. With a rate of 20,000 samples per second, 4 seconds of computer time would be needed to generate each second of music. The Fig. 6 instrument requires 0.5 millisecond per sample, or 10 seconds computer time per second music. The computation cost to produce the music may typically come to as much as $100 per minute of music. Fortunately, computation costs are steadily decreasing.

VII. SOME PROGRAMMING DETAILS

For the benefit of programmers who may be interested, the operation of the compiling and score reading programs will be outlined very briefly. The compiling program at present consists of a symbolic assembly program for the IBM 7090 which has provisions for macro instructions. The instruments are closed subroutines assembled from these instructions. Consequently, it is quite possible to insert basic machine language instructions into the instruments simply by interspacing these with the macros. This ability to fall back on basic machine language is always desirable in a compiler.

The first instruction in any instrument bears the symbolic address nU1, where n is the instrument number. In playing the instrument, control is transferred to this point by the main program. Control is returned to the main program by the last unit generator (because of its type, G1) after one sample of acoustic output has been generated.

At the beginning of each note certain parameters in the instrument must be set. This is done by another closed subroutine, called the "setter," which is assembled along with the instrument. This subroutine delivers the parameters of the note card to the appropriate Fortran subprograms and stores the parameters computed by these programs in the instruments. The flexibility of the macro compiler is essential here, since various numbers and various types of parameters must be accommodated for the different instruments. The first instruction in the "setter" subroutine is designated In, where n is the instrument number.

The main score-reading and sound-generating program is assembled along with the instruments, so that all symbolic addresses are common to both. The main program operates in two phases, the first of which is a card-reading phase, which is terminated by a MES measure card. All the note cards in the first measure are read and their parameters stored in memory.

At the end of a measure, a sorting process must be carried out to put all the note cards into the time sequence in which the events occur in the measure, since time sequence with several instruments does not necessarily correspond to card sequence. After the sort, the setting subroutines are called to set the parameters in the instruments playing in the first time interval, and all these instrument subroutines are called N times, where N is the number of samples in the first interval. The process is repeated for the second interval, etc., until the measure is completed.

The cards for the following measure are then read, and the cycle continues until a termination card, TER, ends the composition.

The program, as written, is a compromise between efficiency, flexibility, and simplicity. The writing time was about a man-month, which is short for a compiler. This speed is attributable to the versatility of the macro assembly program. By using stored functions and setting instrument parameters only at the beginning of notes, a rather efficient program was achieved. The flexibility rests mainly on the ease with which new unit generators can be defined with new macros, and on the possibility of inserting any desired machine language instructions. So far, the program seems adequate for its objectives.

VIII. RESULTS AND CONCLUSIONS

The program has been used to generate a wide variety of sounds and sequential signals. These include musical compositions; sets of test tones to study attack, decay, and vibrato; control signals for a speaking machine; stimuli for a study of absolute pitch perception; and a set of random signals of various bandwidths and frequencies for listening tests.

The musical compositions demonstrated both the facility with which scores can be written and the range of sounds which can easily be produced. The most striking effects are the continuous modulation from one instrument type into another, precisely controlled vibratos with attack and decay of the vibrato rates, the rapid sweep of frequency over many octaves in a single note, frequency as well as amplitude attacks on notes, and the representation of a melodic line by the sum or difference of the frequencies of two voices.

The compositions affirmed that a deeper understanding of how sounds are perceived is necessary before we can effectively use the new instruments that can be compiled. However, the program itself is proving an excellent tool in carrying out studies. For example, a systematic variation of attack and decay times showed the predominant influence of attack in timbre. Similar examinations were carried out for vibrato and random signals.

The program proved a convenient way of producing a random sequence of 12-tone chords in which the notes in a chord were all in octave relationship. These chords were used as psychological stimuli to test the feasibility of teaching absolute pitch. Other applications for generating psychological stimuli have been suggested.

The control signals to a speech synthesizer are fundamentally functions of time which must be flexibly specified. The music program, although conceived for producing a sequence of notes, proved ideal for this purpose. It probably would also be possible to synthesize an entire speaking machine as a complex instrument.

An important question is, "How easily can the professional composer make use of the program?" Probably the compilation of the instruments of an orchestra requires programming skill beyond that of most musicians, although a mathematically minded one would easily learn the technique. However, writing a score for an existing orchestra can be systematized to such an extent that almost anyone can create a composition. Consequently, it seems quite feasible for a musician without mathematical training to carry out his wishes with the aid of only a little programming help.

The compiler has great inherent flexibility in that new unit generators can easily be added to the group available for compiling instruments. An example is the random signal generator which was only recently programmed. In addition, the Fortran subprograms contribute to the versatility. Because of this ability to change and grow, we believe the compiler will be valuable for the production of computer music and stimuli for some time to come. We expect programs such as this, together with the cheaper, faster computers which are promised, to result in computer-generated sounds becoming of increasing utility.

APPENDIX

Unit Generators for Acoustic Compiler

A brief description of the most frequently used unit generators is given here. New generators are often added, so the list is not complete.

Output Unit — G1.

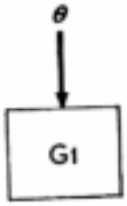
Call statement:

MAC G1, nUm.

Input designation: θ.

Function: To add the number stored in θ to the acoustic output and transfer control from the instrument to the main sequencing program. An output unit *must* form the last generator in every instrument and must not be used in any other position.

Diagram:



Periodic Function Generator — G2.

Call statement:

MAC G2, nUm, A, (pUq,B), C, D.

A = designation of stored function F_j;
(pUq, B) = location of output;
C = fixed designation of M input;
D = fixed designation of I input.

Input designation:

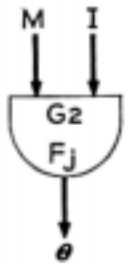
M = amplitude modulation input;
I = frequency control input;
S = initial value of S_i.

Function:

$$\theta_i = M_i F_j\{[S_i]_{\text{mod } 512}\};$$

$$S_{i+1} = S_i + I_i.$$

Diagram:



Adders — G3, G4, G5.

Call statement:

MAC G3, nUm, (pUq,B), C, D

MAC G4, nUm, (pUq,B), C, D, E

MAC G5, nUm, (pUq,B), C, D, E, F.

(pUq,B) = location of output;
C, D, E, F = fixed designation of inputs A1, A2, A3, A4 respectively.

Input designation: A1, A2, A3, A4

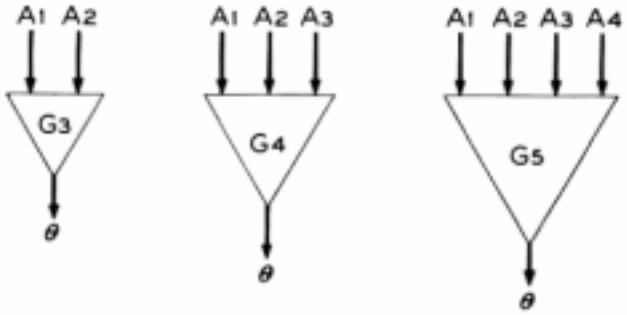
Function:

G3 θ = A1 + A2

G4 θ = A1 + A2 + A3

G5 θ = A1 + A2 + A3 + A4

Diagram:



Random Signal Generator.

Call statement:

MAC RAND, nUm, A, B, pUq, C.

A = fixed designation of M input;
B = fixed designation of I input;
pUq, C = location of output.

Input Designation:

M = amplitude modulation input;
I = frequency control input.

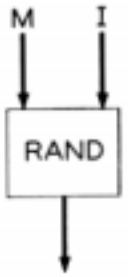
Function:

$$\theta_i = M_i R_i[I_i],$$

where $R_i[I_i]$ is a random variable whose bandwidth is controlled by I_i. The bandwidth is approximately equal to

$$\frac{\text{sampling rate}}{2} \times \frac{I_i}{512}.$$

Diagram:



REFERENCES

1. David, E. E., Jr., Mathews, M. V., and McDonald, H. S., A High-Speed Data Translator for Computer Simulation of Speech and Television Devices, Proc. Western Joint Computer Conf., March 1959.
2. Mathews, M. V., and Guttman, N., Generation of Music by a Digital Computer, Proc. Third International Congress on Acoustics, 1959, Elsever Publ. Co., Amsterdam, to be published.
3. Shannon, C. E., A Mathematical Theory of Communications, B.S.T.J., **27**, 1948, pp. 379, 623.