2 A Sequence of Tutorial Examples of Sound Generation

Introduction

This chapter is intended to provide a training course in the use of Music V by discussing a series of examples ranging from simple to complex sound synthesis. It is written from the point of view of the user of Music V. Details of operation of the programs will be suppressed as much as possible. These can be found in Chapter 3. Because the programs will not be described here, many of the conventions of the computer score will seem arbitrary and must be temporarily accepted on faith.

For concreteness we will also arbitrarily assume values for certain parameters of the program, for example, a sampling rate of R = 20,000 Hz. Other parameters will be introduced as required. For the student's benefit, the parameters of the training orchestra are listed at the beginning of the problems for Chapter 2.

The material assumes that the student has a working knowledge of FØRTRAN programming. The programming examples will be written in FØRTRAN IV. It is also assumed that the student understands the general functioning of a computer—arithmetic, memory, input–output, and program. If necessary, these skills can be learned from books cited in the references at the end of Chapter 2.

This chapter is intended as training material and not as a reference

manual for Music V. Reference material is organized and presented in Chapter 3.

The Simplest Orchestra

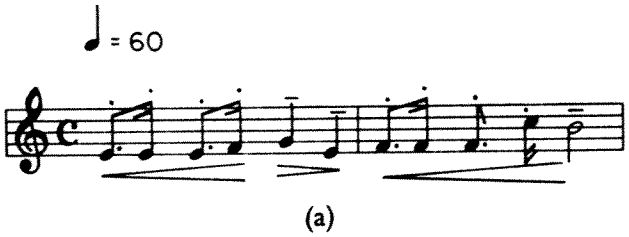
By way of introduction, an elementary orchestra and score are shown in Fig. 27. We shall start by describing the material and then explain the details of its operation. Figure 27a shows the conventional score of the few notes that will be synthesized. Figure 27b shows the block diagram of the simulated instrument that will play the score. It consists solely of an oscillator and an output box. The oscillator has two inputs; the amplitude of the output equals P5; the frequency is proportional to P6. The waveform of the oscillation is determined by stored function F2, which is sketched in Fig. 27c.

The records in the computer score, Fig. 27d have been numbered for reference in this discussion. Each record has a sequence of entries designated P1, P2, P3, etc. In the training orchestra, up to 30 entries (P1–P30) may be used. The entries are separated either by blank spaces or by a comma. Each record is terminated by a semicolon. A record may extend over several lines; conversely, several records may be put on one line.

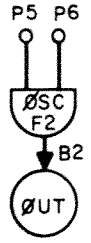
Records 1 through 4 define, for the computer, the instrument shown in Fig. 27b. INS 0 1; says that, at time 0 in the composition, instrument 1 will be defined. ØSC P5 P6 B2 F2 P30 ; says that the first unit generator in the instrument will be an oscillator, will have inputs P5 and P6, will use function F2 for its waveform, will store its output in I–Ø block B2, and will use P30 for temporary storage (which we will discuss later). ØUT B2 B1; says to take the samples in I-Ø block B2 and add them to the contents of block B1 in preparation for outputting these samples. END; terminates the instrument definition.

Record 5 defines the function F2 (Fig. 27c) and causes it to be generated and stored in the computer memory assigned to F2.

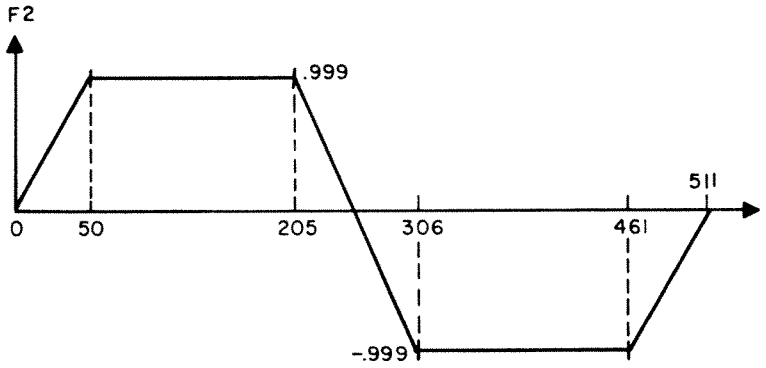
Notes 1 through 11 in the score are generated by records 6 through 16, respectively. In each of the records P1 (NØT) says the purpose of the record is to play a note. P2 gives the starting time of the note measured in seconds from the beginning of the composition. P3 (1) gives the instrument number on which the note will be played. P4 gives the duration of the note in seconds. Durations of staccato notes are written to produce more silence between successive notes than the corresponding silence for the legato notes. P5 gives the amplitude of the note as required by the instrument. In the training orchestra, amplitude can vary over the range 0 to 2047. Amplitudes are varied to



(a)



(b)



(c)

```
 1  INS 0 1 ;
 2  ØSC P5 P6 B2 F2 P30 ;
 3  ØUT B2 B1 ;
 4  END ;
 5  GEN 0 1 2 0 0 .999 50 .999 205 -.999 306 -.999 461 0 511 ;
 6  NØT 0 1 .50 125 8.45 ;
 7  NØT .75 1 .17 250 8.45 ;
 8  NØT 1.00 1 .50 500 8.45 ;
 9  NØT 1.75 1 .17 1000 8.93 ;
10  NØT 2.00 1 .95 2000 10.04 ;
11  NØT 3.00 1 .95 1000 8.45 ;
12  NØT 4.00 1 .50 500 8.93 ;
13  NØT 4.75 1 .17 500 8.93 ;
14  NØT 5.00 1 .50 700 8.93 ;
15  NØT 5.75 1 .17 1000 13.39 ;
16  NØT 6.00 1 1.95 2000 12.65 ;
17  TER 8.00 ;
```

(d)

Fig. 27. Elementary orchestra and score: (a) conventional score; (b) instrument block diagram; (c) waveform; (d) computer score.

correspond to the dynamic markings on the conventional score. P6 equals .02555 times the frequency of the note in cycles per second (hertz). The proportionality constant .02555 will be explained below. Record 17 terminates the composition at 8 sec.

Simple Unit Generators to Output, Add, and Multiply

Having introduced a simple orchestra and score from the user's standpoint, we will now describe in more detail the operation of a few simple unit generators. Although they are simple, these are the most frequently used building blocks for all instruments.

As we showed in Chapter 1, the acoustic output wave is produced by passing a sequence of numbers (samples) $s_0, s_1, \ldots, s_i$ through a digital-to-analog converter and driving a loudspeaker with the analog voltage from the converter. The first sample s_0 is the amplitude of the acoustic wave at the beginning of the composition at $t = 0$, where t is time. The second sample s_1 is the amplitude one sampling time later. We shall assume a sampling rate of 20,000 Hz for the training orchestra; hence s_1 is put out at $t = 1/20{,}000$ sec. $s_{40,000}$ is the amplitude at $t = 2$ sec. It is quite possible, though seldom useful, to specify the sample that controls the amplitude of the acoustic output at any 1/20,000 sec throughout the entire composition.

The purpose of the portions of the Music V program called "instruments" is to calculate all the s_i samples. For example, if a note is to be played from 3 sec to 4 sec in the composition, samples $s_{60,000}$ through $s_{80,000}$ must be computed. The nature of $s_{60,000}$ through $s_{80,000}$ determines the characteristics of the sound—its pitch, loudness, timbre, everything. The nature of the samples is, in turn, determined by the particular unit generators that are put together to form the instrument and by the numbers on the data records that control these generators.

A problem that must be solved by Music V is to keep track of time so as to "turn on" a given instrument program at the sample at which its note should begin, and to "turn off" the instrument at the sample at which its note should end. The starting sample is computed simply by multiplying the starting time of the note given in P2 by the sampling rate. The terminating time is P2 plus the duration P4, and the terminating sample number is the sampling rate times the terminating time. Because of the universal necessity for this control of time, P2 and P4 must always be used for starting time and duration in all records which specify notes.

A second problem facing Music V is to combine the numbers from all instruments that are playing simultaneously at a given time. The

digital-to-analog conversion process demands that the samples be output in sequence, s_1 followed by s_2, followed by s_3, and so forth. Thus the contribution of all instruments to a sample must be computed simultaneously. A way to accomplish this end, which has been used in earlier programs, is to calculate one number from each active instrument, combine these numbers (by addition), output the sample, and then proceed to the next sample. Music V operates in essentially this way, but for additional efficiency it calculates a block of numbers from each instrument instead of a single number. These blocks, called I-Ø blocks, are one of the fundamental data storage units in the program.

I-Ø Blocks

I-Ø blocks are short for unit generator input–output blocks. They can be used as storage locations for either inputs or outputs for unit generators, hence the designation input–output blocks. Blocks are designated B1 through B10 in the training orchestra. Block B1 has the special function of storing the numbers that will be sent to the digital-to-analog converter. All other blocks are equivalent in mono. (In stereo, blocks B1 and B2 are both reserved for output.)

The size of the block is a parameter of the orchestra. In the training orchestra, it has been set at 512. The maximum size of numbers in the I-Ø blocks is another program parameter. In the training orchestra it has been set at ± 2047 which is appropriate for a 12-bit digital-to-analog converter.

AD2 Generator

The simplest generator is the two-input adder, AD2. Its function is to combine two numbers by addition. It has two inputs and one output as shown in Fig. 28a. The equation of operation is

$$Ø_i = I1_i + I2_i$$

where I1 and I2 are the two inputs, Ø is the output, and i is the index of samples that starts at 0 at time $t = 0$. We must quickly add that this equation is computed only for those samples during which the instrument with AD2 is playing a note.

In the score, AD2 is put in an instrument by a statement such as

```
AD2  B2  B4  B3 ;
```

This example says: take the numbers stored in block B2, add them to those stored in block B4, and put the sum in block B3. The relation

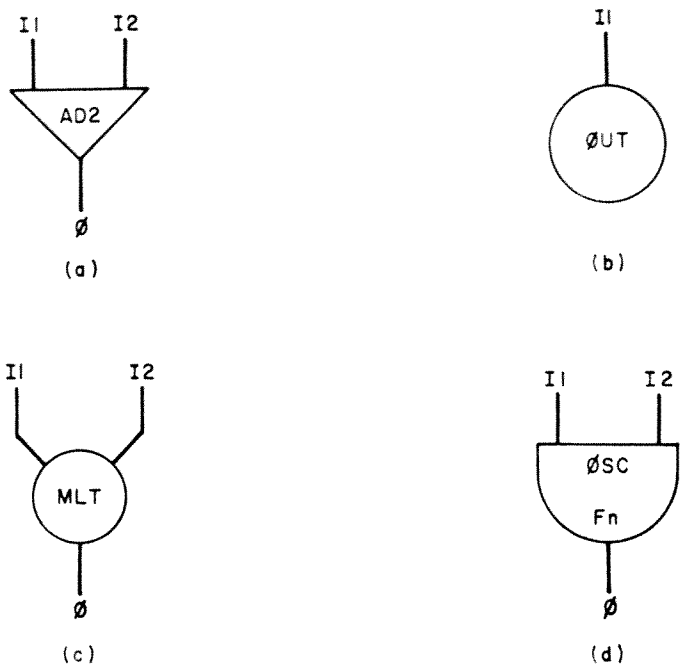


Fig. 28. Four simple unit generators: (a) AD2; (b) ØUT; (c) MLT; (d) ØSC.

between sample index i and the numbers in a given block at a given time need not worry the user; it is treated automatically by the program.

AD3 and AD4 also exist and form a sum of three and four inputs, respectively. The score statement evoking AD4 would be

AD4 B2 B3 B4 B5 B6 ;

where B2 through B5 are inputs and B6 the output.

ØUT Generator

The ØUT generator takes the numbers from an instrument and places them in the special I-Ø block B1 for subsequent outputting through the digital-to-analog converter. ØUT also combines the numbers with any other instrument simultaneously being played. ØUT is diagrammed in Fig. 28b. It is shown with one input. The output to B1 is not shown; it always goes to this block. The equation of operation is in FØRTRAN-like notation

$$\text{Acoustic output}_i = \text{acoustic output}_i + \text{I1}_i$$

This equation says: I1 is added to anything previously in the acoustic output block; by this simple means any number of instruments may be combined. The operation of addition is perfectly equivalent to the

way in which sound waves of several real instruments combine in the air.

In the score ØUT is evoked by a statement such as

```
ØUT B2 B1;
```

where B2 is the block containing the input, and B1 is the special block for acoustic output.

MLT Generator

The MLT generator multiplies two numbers together in a manner exactly analogous to the addition done by AD2. It is diagrammed in Fig. 28c. The equation of operation is

$$Ø_i = I1_i \cdot I2_i$$

where I1 and I2 are the two inputs and Ø is the output. In the score

```
MLT B2 B3 B4 ;
```

associates I1 with B2, I2 with B3, and Ø with B4. In general, the order of listing generator descriptions on the score is: inputs, outputs, special parameters.

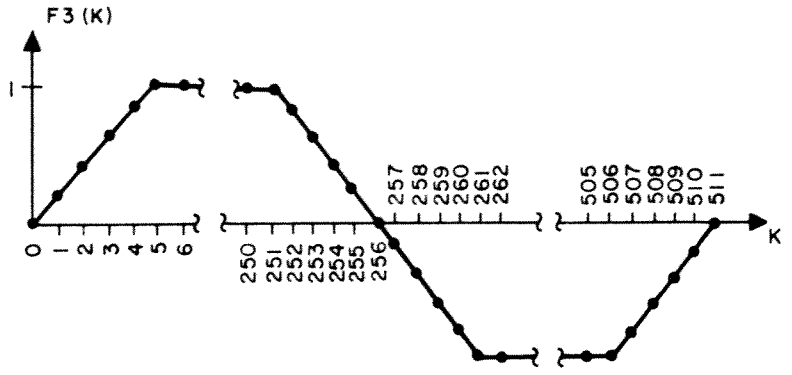
ØSC Generator[1]

By far the most important generator is the oscillator ØSC. It is the most frequently used and the most difficult to understand of the simple generators. Its importance is based on the prominence of oscillations in musical sounds and on its nature as a *source* of numbers. The generators previously described modify or output numbers that have been created elsewhere; ØSC is one of the few units that actually produce numbers.

The diagram of ØSC is presented in Fig. 28d. As will be shown, three quantities determine the output Ø: I1 controls the amplitude of the oscillation; I2 controls the frequency; and F_n, a stored function, is the waveform. F_n is exactly one cycle of the ØSC output; the purpose of the ØSC can be looked upon as repeating F_n at the desired frequency and amplitude.

F_n may be thought of as a continuous function of time, but in the computer it must be represented by a block of samples. In the training orchestra each function is represented by 512 samples. Figure 29 shows an example of a stored function F3. The waveform is a square wave with slightly slanted sides. The 512 points, F3(k) k = 0...511, are

[1] Also see Chapter 3, Section 6, for a basic discussion of ØSC.



F3(0) = 0	F3(250) = 1.0	F3(505) = −1.0
F3(1) = .2	F3(251) = 1.0	F3(506) = −1.0
F3(2) = .4	F3(252) = .8	F3(507) = −.8
F3(3) = .6	F3(253) = .6	F3(508) = −.6
F3(4) = .8	F3(254) = .4	F3(509) = −.4
F3(5) = 1.0	F3(255) = .2	F3(510) = −.2
F3(6) = 1.0	F3(256) = 0	F3(511) = 0
. . .	F3(257) = −.2	
	F3(258) = −.4	
	F3(259) = −.6	
	F3(260) = −.8	
	F3(261) = −1.0	
	F3(262) = −1.0	
	. . .	

Fig. 29. Function stored as 512 samples.

indicated as dots on the function. Actually only 511 numbers are independent since F3(0) = F3(511). The 512 numbers representing the function are listed below the function. These numbers are actually stored in 512 locations in the computer memory. The programs that calculate and store the numbers are called GEN routines and will be discussed later.

One may ask, why go to all the trouble of having a GEN program compute and store numbers and then have the ØSC program modify and repeat these numbers? Why not, instead, have the GEN programs repeatedly calculate exactly the desired numbers? The reason, the importance of which cannot be overemphasized, is efficiency. ØSC is a very fast number repeater. The GEN programs must be flexible and, hence, they are in comparison very slow.

By denoting a function F3, we imply that several stored functions are possible. In the training orchestra 10 functions, designated F1 through F10, are available.

The simplest ØSC program would simply repeat the 511 numbers in F3, one after the other: F3(0), F3(1),..., F3(511), F3(1),.... This would produce an oscillation whose peak amplitude would be 1 and whose frequency would be 20,000/511 = 39.14 Hz. That frequency is too low for most purposes. By repeating every other sample, F3(1), F3(3),..., F3(511), F3(2),..., one could produce a higher frequency, 78.28 Hz. In general, by repeating every nth sample of F3, one obtains a frequency of

$$\frac{20{,}000}{511} \cdot n \text{ Hz}$$

F3 is stored as samples, as is the output of ØSC: the process carried out by ØSC can be thought of as resampling F3 to obtain a desired frequency. A simple resampling that puts out every nth sample of F3 can produce only frequencies that are multiples of 39.14 Hz. Clearly these offer too limited a choice of frequencies.

The actual algorithm used in ØSC, which overcomes these limitations, is

$$S_{i+1} = S_i + I2_i$$
$$Ø_i = I1_i \cdot F_n([S_i]_{\text{Mod } 511})$$

where

i is the index of acoustic output samples;
S_i is a running sum which increases by $I2_i$ for each successive value of i; S_i is usually set to zero at the beginning of each note;
$[S_i]_{\text{Mod } 511}$ is $[S_i - n \cdot 511]$ where n is selected so that $[S_i]_{\text{Mod } 511}$ always falls between 0 and 511;
$I1_i$ is the amplitude input that multiplies the amplitude of F_n;
$I2_i$ is the frequency controlling input; and
$Ø_i$ is the output.

The operation of ØSC can be understood geometrically by referring to Fig. 30. S_i is a ramp function whose slope is I2 units per sample of acoustic output. $[S_i]_{\text{Mod } 511}$ is the sawtooth function which is reset to zero each time S_i equals a multiple of 511. With a slope of I2, exactly 511/I2 samples are required for S_i to reach 511; hence the period of $[S_i]_{\text{Mod } 511}$ is exactly 511/I2 samples. At a sampling rate of 20,000 Hz, the frequency of $[S_i]_{\text{Mod } 511}$ is

$$\text{Freq} = \frac{20{,}000 \cdot I2}{511}$$

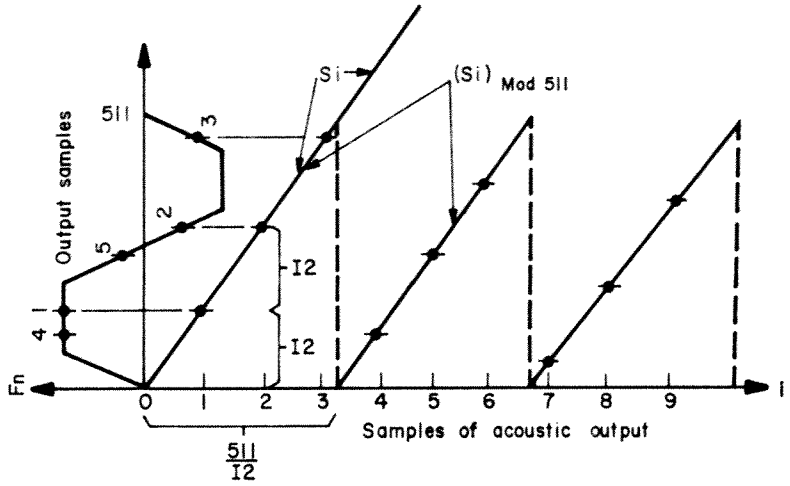


Fig. 30. Operation of ØSC.

This is the fundamental relation between the frequency of ØSC and I2. It can be written

$$\text{Freq} = 39.4 \cdot \text{I2}$$

or in case we want to solve for I2 for a given frequency

$$\text{I2} = .02555 \cdot \text{freq}$$

More generally,

$$\text{I2} = \frac{N_F}{R} \cdot \text{freq}$$

where N_F is the length of a stored function (N_F = number of samples − 1) and R is the acoustic sampling rate.

$[S_i]_{\text{Mod } 511}$ has the desired frequency but the wrong waveform—a simple triangle. $[S_i]_{\text{Mod}}$ is used to scan F_n as specified by the second ØSC equation. The scanning process is equivalent to projecting samples of $[S_i]_{\text{Mod } 511}$ to the left in Fig. 30 and sampling F_n as indicated. This process, along with a multiplication by I1, gives an output of the desired frequency, amplitude, and waveform.

Although $[S_i]_{\text{Mod } 511}$ lies between 0 and 511, it will not, in general, take integer values. Since $F_n(k)$ is sampled and stored only for integer values of k, some accommodation must be made. The simplest ØSC algorithm truncates $[S_i]_{\text{Mod } 511}$ to the next smaller integer value. More complex ØSC routines interpolate $F_n(k)$ between successive k's.

In the score ØSC would be called by a statement such as

```
ØSC P5 P6 B2 F2 P30 ;
```

where P5 is the amplitude input, P6 the frequency input, B2 the I-Ø block for output, F2 the stored function, and P30 is a vacant-note parameter location for storing the sum S_1. One of the note record parameters must be reserved for S_1. Since the initial value of S_1 is zero, the parameter need not be written; unwritten parameters are always set to zero at the beginning of each note.

Examples of Simple Instruments

Having now discussed the four simplest and most important generators, let us look at some examples of instruments constructed of these generators. For each instrument we will show the score cards which define the instrument and play a note or two. The instruments will require two or more stored functions. Although the GEN score cards that generate these functions are shown here we will postpone until later a detailed discussion of the GEN routines.

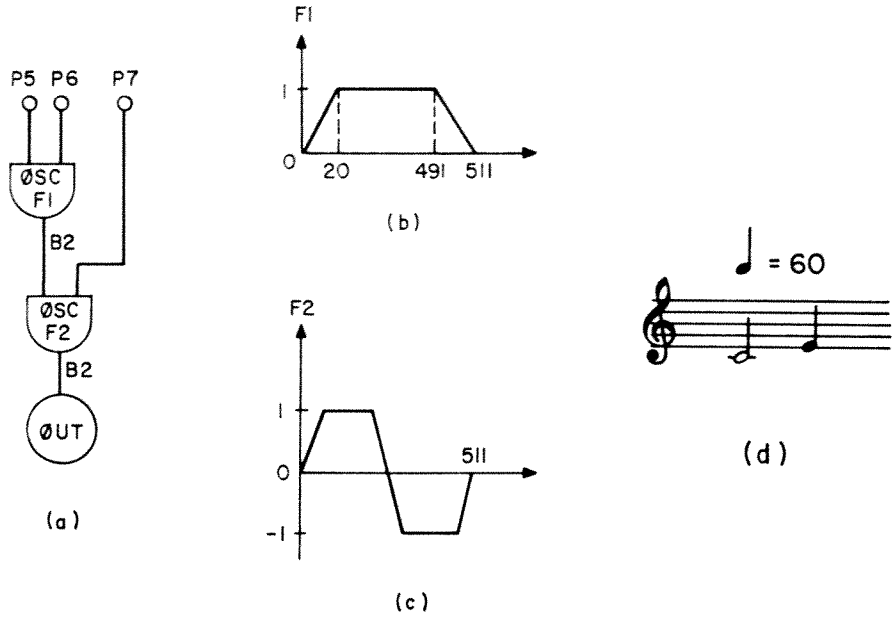
We will also postpone discussion of the conversion function which greatly simplifies writing scores of the notes. Consequently, our scores will be somewhat labored and should not be considered typical.

Instrument with Attack and Decay

The simplest instrument shown in Fig. 27 produces sounds by turning an ØSC on and off suddenly. The sudden transients might be heard as unwanted clicks. An instrument is shown in Fig. 31a with an envelope that gradually increases the sound amplitude at the beginning of the note and decreases the amplitude at the end.

The upper ØSC generates the desired envelope which forms the amplitude input for the lower ØSC. F1, the waveform function for the upper ØSC, is the desired envelope as sketched in Fig. 31b. The ØSC is used in a degenerate mode in that its frequency will be set at the value that permits it to go through exactly one cycle of oscillation during the note being played. Usually this is a very low frequency; however, unlike real oscillators, computer-simulated oscillators can produce low frequencies with ease and precision. The frequency-control equation for ØSC is

$$\text{P6} = \frac{511}{20{,}000} \cdot \text{freq}$$




```
1  INS 0 1 ;
2  ØSC P5 P6 B2 F1 P30 ;
3  ØSC B2 P7 B2 F2 P29 ;
4  ØUT B2 B1 ;
5  END ;
6  GEN 0 1 1 0 0 .99 20 .99 491 0 511 ;
7  GEN 0 1 2 0 0 .99 50 .99 205 -.99 306 -.99 461 0 511 ;
8  NØT 0 1 2 1000 .0128 6.70 ;
9  NØT 2 1 1 1000 .0256 8.44 ;
10 TER 3 ;
```

(e)

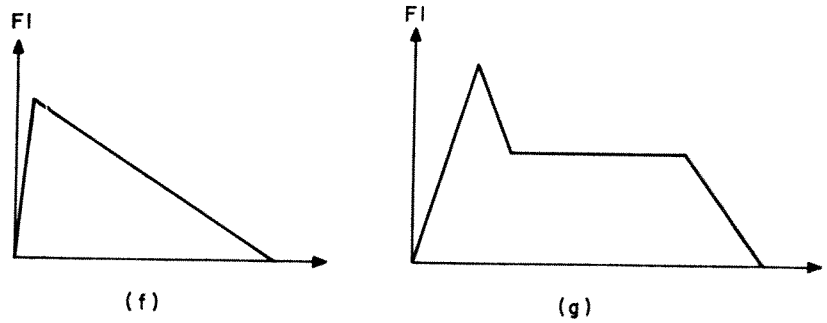


Fig. 31. Instrument with attack and decay: (a) block diagram; (b) envelope function; (c) waveform function; (d) conventional score; (e) computer score; (f) pianolike envelope; (g) brasslike envelope.

If we wish exactly one cycle of oscillation per note,

$$\text{Freq} = \frac{1}{\text{note duration}}$$

or

$$\text{P6} = \frac{511}{20{,}000 \times \text{note duration}}$$

$$= \frac{.02555}{\text{note duration}}$$

Thus, for the first note, whose duration is 2 sec, P6 equals .0128 (line 8 of score) and for the second note, whose duration is 1 sec, P6 equals .0256.

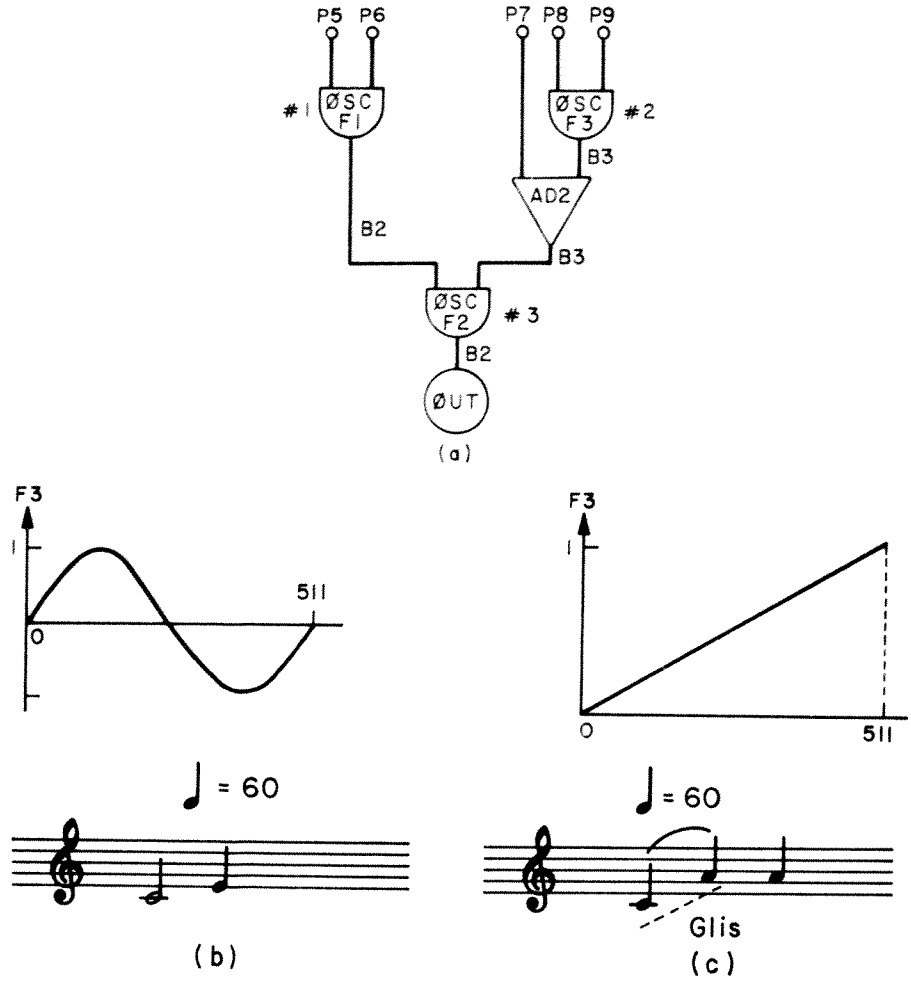
The envelope does much more than eliminate clicks. It is as important in the determination of timbre as the waveform. The attack time is especially important; percussive instruments have very short times (1 or 2 msec), stringed instruments having long times (50–200 msec). In addition, envelopes can have other shapes: the triangular shape shown as an alternate envelope on Fig. 31f is typical of a piano, and the envelope with initial overshoot in Fig. 31g is typical of a brass instrument.

The score, Fig. 31e, is similar to the score in Fig. 27. A few points should be mentioned. The instrument is named "1" and is referred to as "1" in P3 of the NØT cards. I-Ø block B2 is used for both the input and output of the lower ØSC. This is permissible since all the unit generators read their inputs before storing their outputs. However, as will be pointed out later, an I-Ø block must not be used for two different purposes at the same time. The upper ØSC uses P30 to store its S_i; the lower ØSC uses P29. In general, since the S_i's of different ØSC's are different, they must be kept in different locations.

Record 6 causes the generation of the envelope function by evoking GEN1; its operation is the same as in Fig. 27. The envelopes produced by an ØSC have the unfortunate characteristic that the whole envelope stretches and shrinks with the duration of the note. Thus the attack time and the decay time are proportional to duration; the second note in the score will have half the attack time of the first note. Usually this variation is undesirable since it changes the timbre of the note. Special attack and decay generators, which avoid this problem, will be taken up later.

Adding Vibrato

Vibrato, which we will define as a variation in pitch, adds much interest to tone color. In Fig. 32a ØSC #2 and AD2 have been appended to the simple attack and decay instrument to provide vibrato. They




```
 1  INS 0 2 ;
 2  ØSC P5 P6 B2 F1 P30 ;
 3  ØSC P8 P9 B3 F3 P29 ;
 4  AD2 P7 B3 B3 ;
 5  ØSC B2 B3 B2 F2 P28 ;
 6  ØUT B2 B1 ;
 7  END ;
 8  GEN 0 1 1 0 0 .99 20 .99 491 0 511 ;
 9  GEN 0 1 2 0 0 .99 50 .99 205 -.99 306 -.99 461 0 511 ;
10  GEN 0 2 3 1 1 ;
11  NØT 0 2 2 1000 .0128 6.70 .067 .205 ;
12  NØT 2 2 1 1000 .0256 8.44 .084 .205 ;
13  TER 3 ;

10' GEN 0 1 3 0 0 .999 511 ;
11' NØT 0 2 2 1000 .0128 6.70 4.55 .0128 ;
12' NØT 2 2 1 1000 .0256 11.25 0 .0256 ;
```

(d)

Fig. 32. Instruments with vibrato or glissando: (a) block diagram; (b) F3 and score for vibrato; (c) F3 and score for glissando; (d) computer score.

provide a time-varying frequency control to ØSC #3, thus producing a frequency variation in its output. This illustrates that the frequency control of an ØSC does not have to remain constant over a note, but can change in any desired way. P7 controls the average pitch. P8 determines the maximum variation in pitch. P9 determines the rate of variations, which for typical instruments might be 4 to 8 changes per second. The waveshape F3 of ØSC #2 determines the way in which frequency changes with time. The exact shape is usually not critical and a sine wave, as shown, is usually satisfactory.

In the first score card, Fig. 32d, the instrument is named "2" and referred to as such in the P3 fields of the NØT records. An additional I-Ø block B3 is required by the instrument. Block B2 must hold the output of ØSC #1 until ØSC #3 has used it as amplitude input. Consequently, ØSC #2 and AD2 has to use B3 to hold the frequency input for ØSC #3. However, after ØSC #3 has completed its computation, both B2 and B3 are available for other uses; in this case B2 was used to hold the output of ØSC #3.

The order of computation is the order in which generators are written in the score. It is essential to maintain the right order. In the example, ØSC #1 must be written ahead of ØSC #3 since it provides an input to ØSC #3. ØSC #2 must be written ahead of AD2, and AD2 must be ahead of ØSC #3 for the same reason. ØSC #1 could be in any order with respect to ØSC #2 and AD2.

The two GEN1 functions (records 8 and 9) are the same as before. Record 10 calls upon GEN2 to provide a sine wave for F3. P2 = 0 says to compute F3 at t = 0 with respect to the acoustic output. P3 = 2 says to call upon GEN2; P4 = 3 says to compute F3; P5 = 1 says to compute the fundamental with amplitude of 1; P6 = 1 says that there is only one harmonic (i.e., the fundamental).

In note records 11 and 12, P7, P8, and P9 concern pitch and hence are of special interest. The rest of the parameters are the same as in Fig. 31. P7 determines average pitch. Thus for the first note C_{262}

$$P7 = 262 \times .02555 = 6.70$$

P8 is set equal to 1% of P7 so that the maximum frequency deviation will be 1% of the center frequency. Thus the frequency will change from 259.4 Hz to 264.6 Hz. A 1% vibrato is quite large; ¾% is more typical of actual players. However, there is much individual variation in vibrato. P9 determines the number of complete cycles of change per second, which we have set at 8. Thus

$$P9 = 8 \times .02555 = .205$$

With a change in F3, and the meaning of P7, P8, and P9, the same instrument can also be used for glissando. An F3 consisting of a straight "interpolating" function appropriate for glissando is shown in Fig. 32c. P9 now becomes

$$\text{P9} = \frac{.0255}{\text{duration of note}}$$

and causes ØSC #2 to produce one cycle per note (the same as ØSC #1). P7 is set at

$$\text{P7} = .0255 \times \text{initial note frequency}$$

and P8 at

$$\text{P8} = .0255 \times (\text{final note frequency} - \text{initial note frequency})$$

The action of AD2 and ØSC #2 with F3 is such that at the beginning of the note B3 will contain .0255 × initial note frequency, and at the end of the note it will contain .0255 × final note frequency.

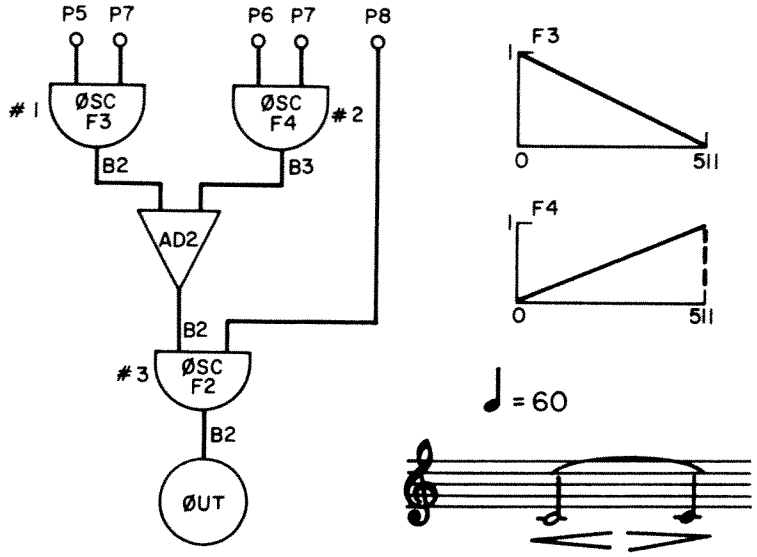
Substitution of cards 10′, 11′, and 12′ into the score in place of cards 10, 11, and 12 will produce the glissando sample shown. Note that for the second note (A_{440}), which has a constant frequency, P8 is equal to 0 since the initial and final frequencies are the same. P6 and P9 have the same values, and hence P9 could be eliminated if the instrument were redefined.

The glissando obtained in this way has a linear change of frequency in hertz. This means that the musical intervals will change faster at the beginning of the slide than at the end. Although a linear change of musical intervals might be preferable, this glissando has been much used and seems perfectly satisfactory. During most slides, listeners are insensitive to the precise time course of the pitch.

Instrument with Swell and Diminuendo

In the glissando instrument, ØSC #2 and AD2 form a linear interpolating unit which generates a frequency control that goes from initial to final frequency. If we apply the interpolating unit to the amplitude control on an ØSC, we can obtain a continuously changing amplitude for crescendos and decrescendos. An instrument with this feature is shown in Fig. 33.

In order to simplify the score, we have complicated the interpolater with an extra oscillator ØSC #2. The glissando instrument required writing the initial frequency in P7 and the (final-initial) frequency in P8. The swelling instrument is arranged so the initial amplitude is




```
 1  INS 0 3 ;
 2  ØSC P5 P7 B2 F3 P30 ;
 3  ØSC P6 P7 B3 F4 P29 ;
 4  AD2 B2 B3 B2 ;
 5  ØSC B2 P8 B2 F2 V1 ;
 6  ØUT B2 B1 ;
 7  END ;
 8  GEN 0 1 3 .999 0 0 511 ;
 9  GEN 0 1 4 0 0 .999 511 ;
10  GEN 0 1 2 0 0 .99 50 .99 205 -.99 306 -.99 461 0 511 ;
11  NØT 0 3 2 0 2000 .0128 6.70 ;
12  NØT 2 3 1 2000 0 .0256 6.70 ;
13  TER 3 ;
```

Fig. 33. Instrument with swell and diminuendo.

written in P5 and the final amplitude in P6. ØSC #1 and ØSC #2 both generate one cycle per note of waveforms F3 and F4, respectively. F3 goes linearly from 1 to 0 over the course of a note and is multiplied by the initial amplitude in ØSC #1. Similarly, F4 goes from 0 to 1 and is multiplied by the final amplitude. Thus the output of AD2 will proceed linearly from the initial amplitude to the final amplitude.

Records 11 and 12 in the score play what amounts to a single note made up of two notes tied together. The first note swells from 0 to maximum amplitude, the second decays back to zero. Amplitude controls in P5 and P6 are obvious. P7 is set to produce one cycle per note in both ØSC #1 and ØSC #2.

One peculiarity is introduced by the structure and use of the instrument. We want the two notes to blend into each other with no break between notes. To achieve this, we have omitted the usual attack and decay ØSC. However, the waveform ØSC #3 must also produce a continuous output over the juncture. If we were to store the sum for the ØSC in an unused note parameter (P30, for example), it would be reset to zero at the beginning of each note, a sudden change of phase between notes would result, and a click might be introduced. To avoid this transient, the sum is stored in variable V1. The training orchestra provides space for 200 variables, denoted V1 through V200. These variables may be changed by either the instruments or the score, but they are not reset at the beginning of a note. Consequently, storing the sum of ØSC #3 in V1 assures that it will never be reset and that the oscillator will proceed continuously between all notes. However, this instrument will be limited to playing only one voice.

There are many other uses for variables, as we will see in the next example.

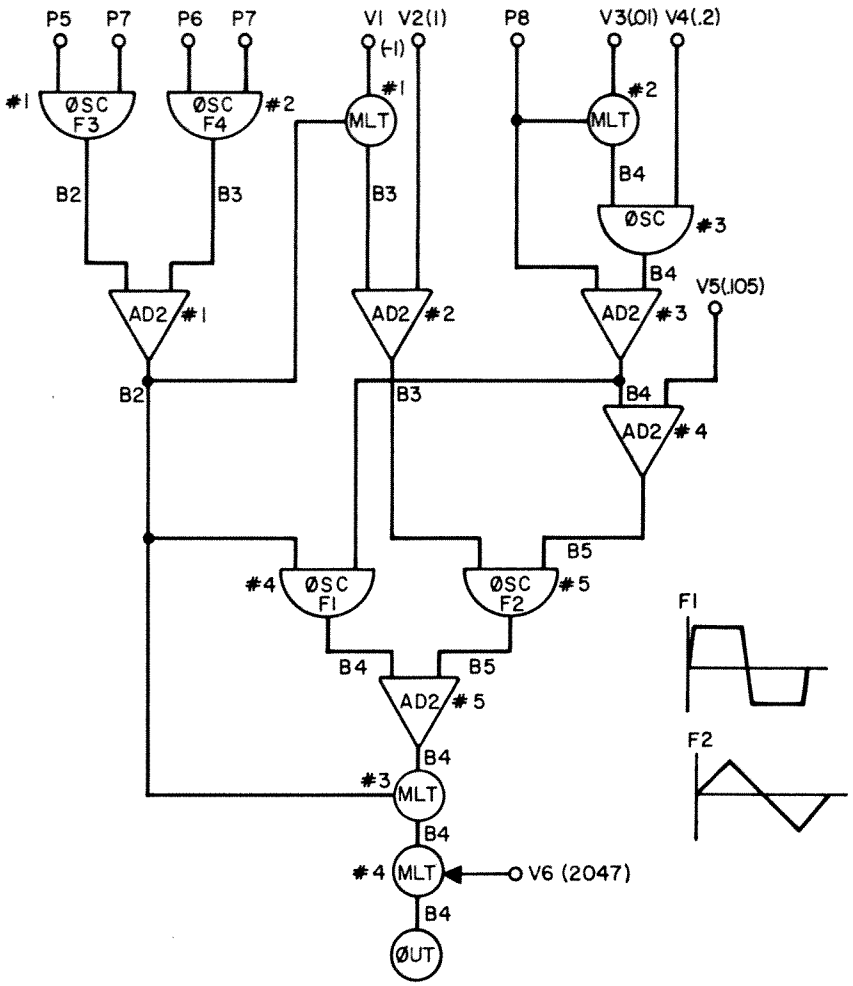
Instrument that Varies Waveform with Amplitude

We conclude these examples of simple instruments with a not-so-simple one. It has been shown that one of the factors that contribute interest to the timbre of real instruments is a change in spectrum with the intensity of the sound. Usually the loud sounds have more high-frequency components than the soft sounds. Figure 34 shows an instrument that is able to change spectrum with amplitude.

The instrument is an elaboration of the swell and diminuendo instrument shown in Fig. 33, and it uses the same parameters on the note records. ØSC #1, ØSC #2, and AD2 #1 form a linear interpolation unit with P5 as the initial amplitude and P6 the final amplitude. These inputs range from 0 to 1 with 1 as the maximum output. We will call the instantaneous amplitude Amp_1. Amp_1 is stored in block B2. MLT #1 and AD2 compute B3 according to the relation

$$B3 = 1 - Amp_1 = Amp_2$$

Oscillator ØSC #4 is controlled by Amp_1, and ØSC #5 by Amp_2. Thus when Amp_1 is 0, Amp_2 is equal to 1, and all the output comes from ØSC #5; when Amp_1 is 1, Amp_2 is equal to 0, and all output comes from ØSC #4. At intermediate values of Amp_1, intermediate portions of output come from ØSC #5 and ØSC #4. In this way the waveform of F2 in ØSC #5 controls the spectrum at low amplitudes, and the waveform of F1 in ØSC #6 controls at high amplitudes.




```
 1  INS 0 4 ;
 2  ØSC P5 P7 B2 F3 P30 ;
 3  ØSC P6 P7 B3 F4 P29 ;
 4  AD2 B2 B3 B2 ;
 5  MLT B2 V1 B3 ;
 6  AD2 B3 V2 B3 ;
 7  MLT P8 V3 B4 ;
 8  ØSC B4 V4 B4 F5 P28 ;
 9  AD2 P8 B4 B4 ;
10  AD2 B4 V5 B5 ;
11  ØSC B3 B5 B5 F2 V7 ;
12  ØSC B2 B4 B4 F1 V8 ;
13  MLT B2 B4 B4 ;
14  MLT B4 V6 B4 ;
15  ØUT B4 B1 ;
16  END ;
```

Fig. 34. Instrument that varies waveform with amplitude.

The amplitude of the sum of ØSC #4 and ØSC #5 is relatively independent of Amp_1. The normal dependence is restored by MLT #3. The output of MLT #3 ranges from −1 to +1; MLT #4 increases this range to −2047 to +2047, the normal amplitude range.

Frequency control of ØSC #4 and ØSC #5 is a vibrato circuit plus AD2 #4, which makes ØSC #5 4 Hz higher in frequency than ØSC #4. A slight divergence adds richness to the tone. The amplitude of the vibrato is automatically set at 1% of the center frequency of the tone by MLT #2. This is an expensive way of controlling amplitude, and better ways will be discussed when CØNVT functions are considered. The frequency of vibrato is set at about 6 Hz by V4.

The instrument requires six constants as inputs. These are stored in V1 through V6: V1 = −1, V2 = 1, V3 = .01, V4 = .2, V5 = .105, and V6 = 2047. The record that stores these constants is

```
SV3 0 1 −1 1 .01 .2 .105 2047;
```

P1 and P2 say to set variables in Pass III at time 0. P3 says to start with variable 1 and continue with 2, 3, etc., to the end of the data. P4–P9 give the six numbers to be set in V1–V6. New variables can be set at any time, as previously set variables can be changed, with other SV3 cards. Times of settings and changes are all controlled by P2.

We will not write a score for this instrument since, except for setting variables, little new is involved. A reasonable choice for F1 and F2 is sketched in Fig. 34. The harmonics of F1 decrease at about 6 dB per octave; those of F2 at 12 dB per octave. Thus the instrument is likely to have higher-frequency energy at high output amplitudes. Other more interesting examples of F1 and F2 could be devised.

CØNVT Function to Process Note Parameters

Scores for the instruments thus far discussed contain many affronts to a lazy composer, and all composers should be as lazy as possible when writing scores. For example, computing the frequency control of an oscillator as

I2 = .02555 × frequency in hertz

is a tedious process. Instead, one would like to write the notes of a scale directly, such as the numbers 0–11 for a 12-tone scale.

A FØRTRAN routine named CØNVT is called at the end of Pass II; it can apply the full power of FØRTRAN to convert the note parameters as written by the composer into a new set of parameters, which are the

inputs to the instruments. As will be clear from the examples below, the nature of CØNVT depends on the instruments used with it. Consequently, no universal CØNVT program is or can be supplied with Music V; instead the composer must write his own for each orchestra he defines. Let us explore the possibilities with the simple attack and decay instrument designed in Fig. 31.

We shall assume that the composer would like to write frequency directly in hertz and would like to write amplitude on a decibel scale rather than on a linear scale. Furthermore, the note duration is already written in P4; it is an indignity to have to write P6 (= .02555/duration). Hence we will assume that the composer will write

P5 = amplitude of note in decibels with 66 dB corresponding to a maximum amplitude of 2000

P6 = frequency of note in hertz

With these inputs CØNVT must compute[2]

P5 = 10.0**(P5/20.0)

P7 = 511.0 * P6/(sampling rate)

and

P6 = 511.0/(P4 * sampling rate)

A program to achieve these conversions is given below along with annotated comments.

	Text	Notes
	SUBRØUTINE CØNVT	
	CØMMØN IP, P, G	1
	DIMENSIØN IP(10), P(100), G(1000)	
	IF (P(1) − 1.0) 102, 100, 102	2
100	IF (P(3) − 1.0) 102, 101, 102	3
101	P(5) = 10.0 ** (P(5)/20.0)	4
	P(7) = 511.0 * P(6)/G(4)	
	P(6) = 511.0/(P(4) * G(4))	
	IP(1) = 7	5
102	RETURN	
	END	

Notes

1. The data-record parameters P1–P100 have been placed by Pass II in P(1)–P(100). The IP array contains some pertinent fixed-point

[2] Equations relating to programs will usually be written in a FØRTRAN-like notation.

constants; in particular, IP(1) = number of parameters in the data record. G is a general memory array for Pass II.

2. This statement checks to see whether the data record pertains to a note (rather than a GEN or something else). The numerical equivalent of NØT is 1. Chapter 3, Section 3 lists the numerical equivalent of all the operation codes.
3. This statement checks to see whether instrument #1 is referred to by the NØT record. Other instruments would usually require other CØNVT functions.
4. These statements perform the desired conversions. The sampling rate is always kept in variable G(4). Thus in calculating P(6) and P(7) we have divided by G(4) rather than by the number 20,000. This is desirable because sampling rate is often changed and, if CØNVT always refers to G(4) to obtain the current rate, it will not have to be reassembled with each change of rate. Instead only G(4) need be modified, and this is a simple change which we will discuss shortly.
5. CØNVT has added one parameter P(7); thus the word count IP(1) must be changed to 7. The possibility of *generating additional parameters with CØNVT* is most important and attractive since the composer does not have to write these parameters. In addition, Pass I and Pass II do not have to process and sort these additions, which increases efficiency.

With this CØNVT function the score lines to play the two notes on Fig. 31d (equivalent to lines 8 and 9 on Fig. 31c) are

```
NØT 0 1 2 60 262 ;
NØT 2 1 1 60 330 ;
```

Now let us construct a somewhat more complicated CØNVT function for instrument 2 in Fig. 32. We will again use P5 as amplitude in decibels. Frequency will be specified in terms of an octave, and a 12-tone note within the octave by P6 and P7, P6 giving the octave and P7 the step within the octave. Thus, for example

Note	P6	P7
C_{131}	−1	0
C#	−1	1
D	−1	2
...		

B	−1	11
C_{262}	0	0
C#	0	1
...		
C_{524}	1	0

The vibrato controls will be eliminated from the NØT record. Instead, we will assign two Pass II variables, G(50) to control the percent frequency variation and G(51) the rate of vibrato.

The equations which must be programmed into CØNVT are

Frequency = 262.0 * (2 ** (P6 + P7/12.0))
P5 = 10.0 ** (P5/20.0)
P6 = 511.0/(P4 * sampling rate)
P7 = 511.0 * frequency/sampling rate
P8 = 511.0 * frequency * G(50)/(sampling rate * 100)
P9 = 511.0 * G(51)/sampling rate

Most of the equations are self-explanatory. The note frequency is computed in hertz from the logarithmic scales embodied in P6 and P7 by the first relation. The factor 100 is put in the denominator of P8 because G(50) is a percentage.

Vibrato control is a good example of the use of Pass II memory in a composition. Except for the first few variables, numbers in the G array may be used for any purpose desired by the composer. Numbers are placed in the array by an SV2 record, which is analogous to the SV3 record that was previously used to set a Pass III variable. Thus

```
SV2 0 50 .5 6 ;
```

would set G(50) = .5 and G(51) = 6 at t = 0.

The program to carry out the computations follows.

```
                                                        Text                                    Notes
      SUBRØUTINE CØNVT
      CØMMØN IP, P, G
      DIMENSIØN IP(10), P(100), G(1000)
      IF (P(1) - 1.0) 102, 100, 102
100   IF (P(3) - 2.0) 102, 101, 102
101   P(5) = 10.0 ** (P(5)/20.0)
      P(7) = 511.0 * 262.0 * (2.0 ** (P(6) + P(7)/12.0))/G(4)      1
      P(6) = 511.0/(P(4) * G(4))
      P(8) = P(7) * G(50)/100.0                                    2
      P(9) = G(51) * 511.0/G(4)                                    3
      IP(2) = 9
102   RETURN
      END
```

Notes

1. This statement calculates frequency, multiples it by the appropriate constant of proportionality, and stores it in P(7).
2. This statement computes the maximum vibrato deviation. The properly scaled frequency is already available in P(7) and hence must only be multiplied by G(50)/100.0.
3. This statement sets the rate of vibrato. The constant of proportionality, 511.0/G(4), is the same as any other ØSC frequency control. G(51) will be the vibrato frequency in hertz.

A score for this instrument to replace lines 11 and 12 on Fig. 32 is

```
SV2  0 50 1  6 ;
NØT 0  2 2 60 0 0 ;
NØT 2  2 1 60 0 4 ;
```

Once G(50) and G(51) are set, any number of notes may be written with the same vibrato. On the other hand, the vibrato constants may be changed at any time by a subsequent SV2 record. For example, the deviation could be reduced and the rate increased at t = 15 sec by the record

```
SV2 15 50 .5 8 ;
```

One may ask, why use Pass II variables in CØNVT rather than Pass III variables or Pass I variables which also exist? The answer is, CØNVT is a Pass II subroutine and can only make use of information available in Pass II.

A final example of a CØNVT subroutine will provide a convenient score language for the glissando instrument in Fig. 32. As shown, the initial frequency of a note must be written in P7 and the (final − initial) frequency in P8. We shall eliminate the arithmetic to calculate (final − initial). Instead the score card will have

P5 = amplitude in decibels

and

P6 = final frequency in hertz

The initial frequency of each note will be defined as the final frequency of the preceding note. The initial frequency of the first note will be read into the program with an SV2 card into G(50).

In order to use this simple form the program must remember the final frequency of each note. G(50) will also be used for this purpose.

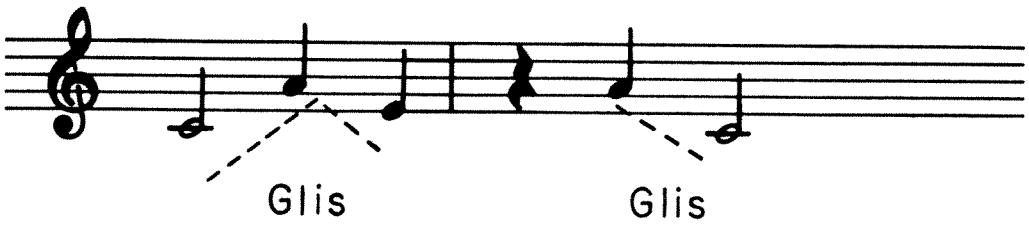
The program to achieve these objectives follows.

	Text	Notes
	SUBRØUTINE CØNVT	
	CØMMØN IP, P, G	
	DIMENSION IP (10), P(100), G(1000)	
	IF (P(1) – 1.0) 102, 100, 102	
100	IF (P(3) – 2.0) 102, 101, 102	
101	P(5) = 10.0 ** (P(5)/20.0)	
	P(7) = G(50) * 511.0/G(4)	1
	P(8) = (P(6) – G(50)) * 511.0/G(4)	2
	G(50) = P(6)	3
	P(6) = 511.0/(P(4) * G(4))	4
	P(9) = P(6)	
	IP(2) = 9	
102	RETURN	
	END	

Notes

1. This statement sets the initial frequency, which was stored in G(50).
2. This statement computes the (final – initial) frequency.
3. This statement stores the final frequency in G(50) to become the initial frequency of the next note.
4. This and the following statement set the frequency inputs of ØSC #1 and #2 to 1 cycle per note.

Figure 35 shows a brief score for the instrument. Only the NØT cards and the SV2 cards are shown. Record 1 sets the initial frequency




```
1  SV2 0 50 262 ;
2  NØT 0 2 2 60 440 ;
3  NØT 2 2 1 60 330 ;
4  NØT 3 2 1 60 330 ;
5  SV2 4.5 50 440 ;
6  NØT 5 2 1 60 400 ;
7  NØT 6 2 2 60 262 ;
```

Fig. 35. Score for glissando instrument using a CØNVT subroutine.

to 262 Hz. Records 2, 3, and 4 generate an initial glissando. An arbitrary choice is made. The frequency slide is completed at t = 3, and the third note is held at E 330. Record 5 resets the initial frequency to 440 for the beginning of the second glissando. The change is done at t = 4.5 in the middle of a rest and hence is inaudible. A different arbitrary choice for glissando is made for the last two notes. During the first note ($5 < t < 6$), the frequency changes from 440 to 400. During the second note ($6 < t < 8$) the frequency shift continues from 400 to 262.

The flexibility of one method for obtaining frequency slides has been demonstrated. The use of Pass II variables as a memory for the CØNVT subroutine is important. Powerful logic may be programmed in this way.

Additional Unit Generators—RAN, ENV, FLT

Next to be discussed are the generator of random signals RAN, the band-pass filter FLT, and the envelope generator ENV. Although a few other generators exist and one can easily design his own generators, the three described here plus the stereo output box are sufficient for most purposes.

Random Signal Generator—RAN

RAN is a source of random signals with controlled amplitude and spectrum. The spectrum is low pass and contains energy from zero frequency to a cutoff frequency determined by one of the inputs (I2). (As will be shown later, RAN may be combined with an oscillator to obtain a band-pass signal.)

The diagram of RAN is shown in Fig. 36a. I1 controls the amplitude of Ø; I2 controls its bandwidth. The equation of operation is

$$Ø_i = I1_i \cdot R_i(I2_i)$$

where $R_i(I2_i)$ is a low-pass random function whose amplitude varies from -1 to $+1$ and whose cutoff frequency is approximately $(R \cdot I2)/1024$ where R is the sampling rate. The amplitude of $Ø_i$ varies from $-I1_i$ to $I1_i$. The approximate spectrum of Ø is shown in Fig. 36b. The cutoff frequency is not abrupt, and there are lobes of energy above $(R \cdot I2)/1024$ Hz. Neither does the main passband have a flat top. Even with these deficiencies, the generator is very useful.

The key to RAN is the generation of the random function R_i. This is obtained by generating a sequence of independent random numbers N_i, which are uniformly distributed over the interval -1 to $+1$. These

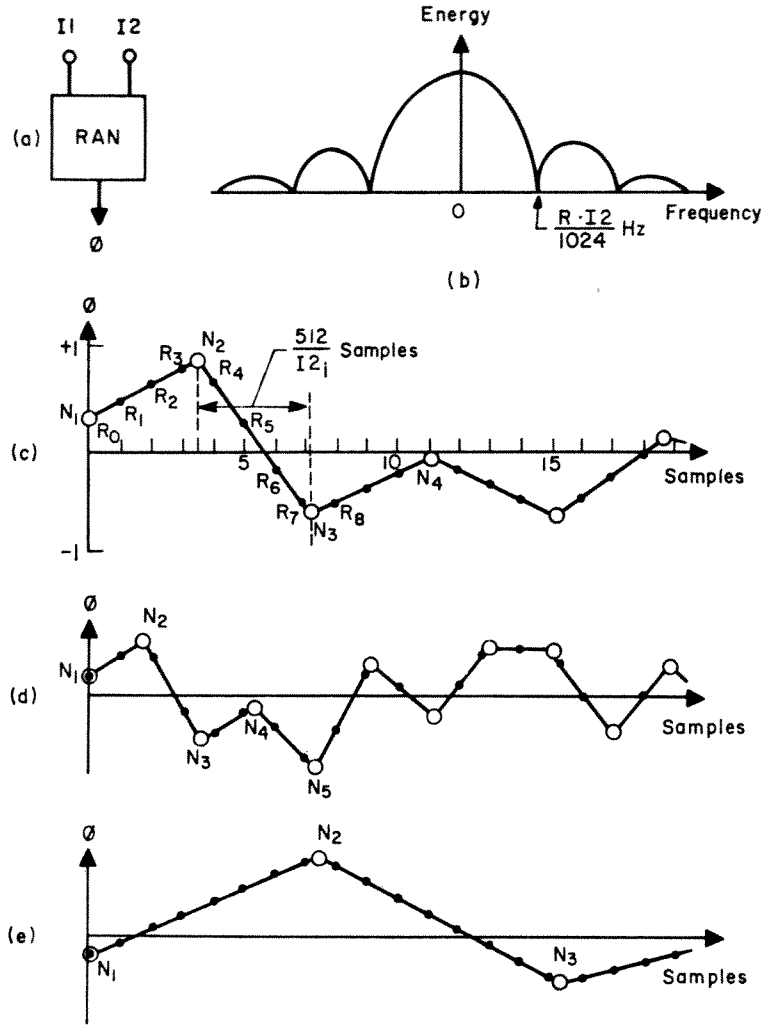


Fig. 36. Random generator—RAN: (a) block diagram; (b) spectrum of Ø; (c) medium-frequency random function; (d) high-frequency random function; (e) low-frequency random function.

numbers are connected by straight lines to form the continuous functions shown in Fig. 36c. R_i consists of samples of the line function, the sampling interval being chosen so that $512/I2_i$ sampling intervals fall between each N_i ($512/I2_i$ does not need to be an integer). The algorithm for sampling the line function is similar to the sum in ØSC.

A medium-frequency function with I2 $\cong$ 128 is shown in Fig. 36c. It wiggles at a moderate rate and at R = 20,000 Hz would have a cutoff of about 2500 Hz.

A high-frequency function with I2 $\cong$ 256 and a cutoff of 5000 Hz is shown in Fig. 36d. The N_i independent random numbers occur more frequently here, giving the function a more jagged appearance and a higher-frequency spectrum. The maximum useful value of I2 is 512.

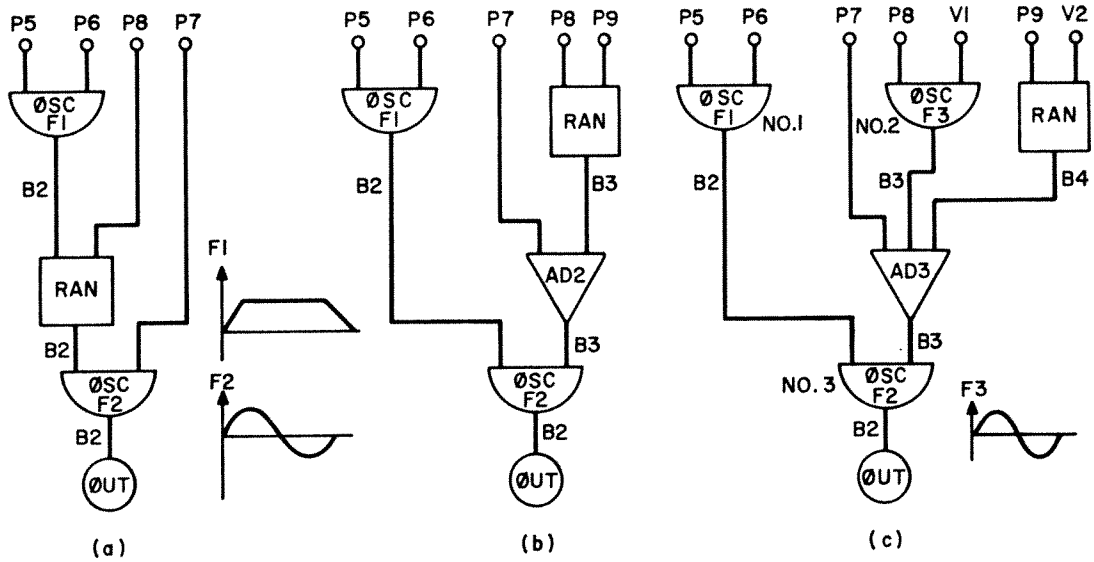
This will produce an independent number for each sample and will achieve a cutoff of R/2, which is the highest frequency representable by R samples per second.

A low-frequency function with $I2_i \cong 64$ and a cutoff of 1250 Hz is shown in Fig. 36e. It is clearly the smoothest of the three functions.

A score record to evoke RAN is

```
RAN P5 P6 B2 P30 P29 P28 ;
```

where P5 is the I1 input, P6 the I2 input, B2 the output; P30 is an unused note parameter for the storage of the sum; and P29 and P28 are two other temporary storage locations.




```
(a) INS 0 1 ;
    ØSC P5 P6 B2
      F1 P30 ;
    RAN B2 P8 B2
      P29 P27 P26 ;
    ØSC B2 P7 B2
      F2 P28 ;
    ØUT B2 B1 ;
    END ;
```

```
(b) INS 0 2 ;
    ØSC P5 P6 B2
      F1 P30 ;
    RAN P8 P9 B3
      P29 P27 P26 ;
    AD2 P7 B3 B3 ;
    ØSC B2 B3 B2
      F2 P28 ;
    ØUT B2 B1 ;
    END ;
```

```
(c) INS 0 3 ;
    ØSC P5 P6 B2
      F1 P30 ;
    ØSC P8 V1 B3
      F3 P29 ;
    RAN P9 V2 B4
      P28 P26 P25 ;
    AD3 P7 B3 B4 B3 ;
    ØSC B2 B3 B2
      F2 P27 ;
    ØUT B2 B1 ;
    END ;
```

Fig. 37. Examples of instruments with random generator: (a) amplitude-modulated band-pass noise; (b) frequency-modulated band-pass noise; (c) periodic plus random vibrato.

Three useful instruments involving RAN are shown in Fig. 37. Instrument 1 produces a band-pass noise by amplitude modulation of an ØSC with RAN. Both the center frequency and the width of the pass-band are controlled by note parameters, P7 determining the center frequency and P8 the width.

The top ØSC produces the initial attack and decay with function F1. The bottom ØSC has a sinusoidal waveform F2, and without RAN it would produce a single-frequency sinusoid at R·P7/511 Hz. By virtue of the multiplication inherent in the amplitude input to ØSC, the sinusoid is multiplied by the output of RAN. Thus the output of RAN is modulated by the sinusoid and according to the convolution theorem (Appendix B) the band-pass spectrum sketched in Fig. 38 is achieved.

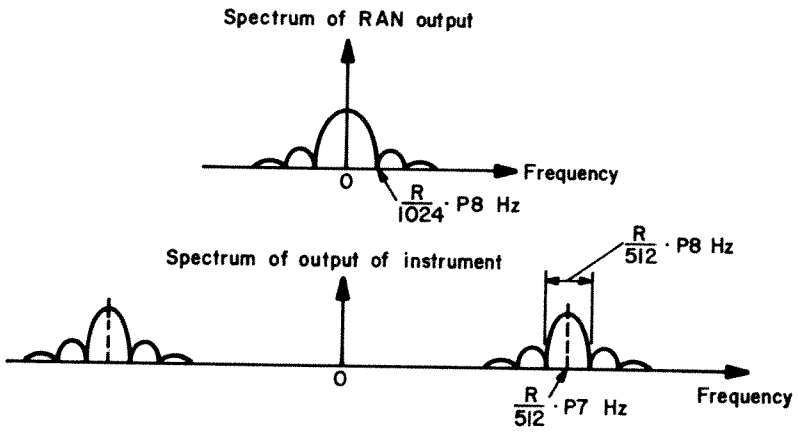


Fig. 38. Spectrum of instrument that generates amplitude-modulated band-pass noise.

The modulation can be looked upon as shifting the (low-pass) spectrum of RAN and centering it about the frequency of ØSC.

If F2 is a complex wave with harmonics, the modulation will generate a replica of the RAN spectrum centered about each harmonic, or a multiple-band noise. Because the auditory effect is usually muddy and unpleasant, most instruments use a sinusoidal F2.

The center frequency of the passband is simply the frequency of ØSC—R·P7/511. The bandwidth is two times the cutoff frequency of RAN or R·P8/512. The factor 2 comes from the negative frequencies in the RAN spectrum which are shifted into positive frequencies by the modulation. The CØNVT function appropriate to this operation is

$$\text{P7} = \frac{511}{\text{R}} \cdot \text{center frequency}$$

and

$$P8 = \frac{512}{R} \cdot \text{bandwidth}$$

It is often desirable to make the bandwidth a fixed percentage of the center frequency. This corresponds to a fixed musical interval about the center frequency. The CØNVT equation is simply

$$P8 = k \cdot P7$$

and P8 must not be written in the score.

Very narrow bands of noise can be generated by small values of P8. In fact, for P8 = 0, a zero bandwidth or pure sinusoid is produced. Narrow-band noises produced by amplitude modulation reveal the way in which they are generated; they sound like a sinusoid with a fluctuating amplitude. This sound is often not what the composer desires or expects from such noises; however, it is an essential characteristic of amplitude modulation and cannot be avoided with this instrument.

INS 2 (Fig. 37b) generates frequency-modulated noise with a band-pass spectrum. The center frequency of the band is again controlled by P7 as $P7 \cdot R/511$. The rest of the characteristics of the spectrum are not as easy to estimate as in the case of the amplitude modulated noise. $P8 \cdot R/511$ is the maximum instantaneous deviation of frequency of ØSC. Frequency-modulation theory says that the width of the noise band will be somewhat greater than $2 \cdot P8 \cdot R/511$. For most purposes $2 \cdot P8 \cdot R/511$ is a useful estimate of bandwidth.

P9 determines the rate at which the frequency of ØSC deviates. Its effect on the spectrum is hard to compute precisely. Experience has indicated that in order to produce "smooth" sounding noise, P9 should be about five times P8. CØNVT is a convenient place to set both P8 and P9.

At very small bandwidths, INS 2 sounds like a sine wave with a small random variation in frequency.

INS 3 (Fig. 37c) shows an excellent vibrato circuit devised by J. C. Tenney. The frequency variation contains a periodic component supplied by ØSC #2 and a random component supplied by RAN. A useful set of parameters is

$$P8 = P9 = .0075 * P7$$

to give $\frac{3}{4}$-percent periodic and $\frac{3}{4}$-percent random variation

$$V1 = 8 * 511/R$$

for an 8-Hz periodic vibration rate and

$$V2 = 16 * 511/R$$

for a 16-Hz random bandwidth. The random bandwidth tends to be substantially greater than the periodic frequency.

Envelope Generator—ENV

The use of ØSC as an envelope generator is satisfactory in some applications, but it makes the attack and decay times proportional to the total note duration. Important aspects of timbre depend on the absolute attack time. With ØSC, these will change from long notes to short notes. The difference may be enough to give the impression of two different kinds of instruments.

A special generator ENV has been programmed to sweep away this limitation. It allows separate control of attack time, steady-state duration, and decay time. In order for ENV to be effective, a special CØNVT function must be written for ENV. The computations in CØNVT are at least as complex as those in ENV.

An instrument using ENV is illustrated in Fig. 39. ENV has four inputs I1–I4 and requires one function. I1 determines the amplitude of the output, and I2, I3, and I4 the attack time, the steady-state time, and the decay time, respectively. The function F1 is divided into four equal sections, the first determining the shape of the attack, the second the shape of the steady state, and the third the shape of the decay. The last section is not used and should be zero to allow for any round-off error involved in scanning the first three parts.

The output $Ø_i$ may be written

$$Ø_i = I1_i * \text{function (scanned according to I1, I2, and I3)}$$

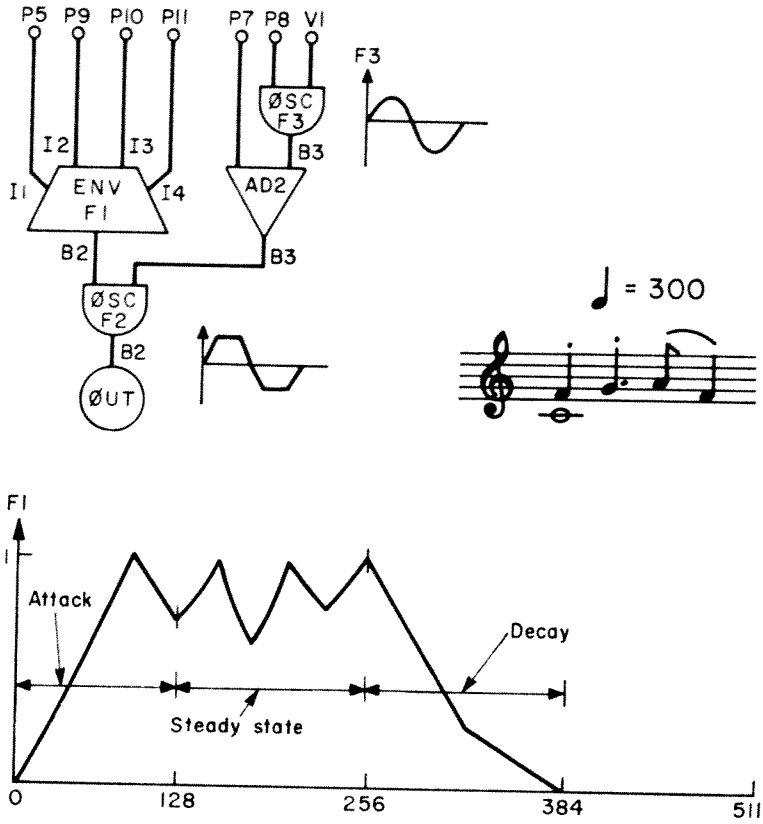
The first quarter of the function is scanned at a rate of I2 locations per sample, the second quarter at a rate of I3 locations per sample, and the third quarter at a rate of I4 locations per sample. Consequently, CØNVT should compute

$$I2 = \frac{128}{\text{attack time} * \text{sampling rate}}$$

$$I3 = \frac{128}{\text{steady-state time} * \text{sampling rate}}$$

and

$$I4 = \frac{128}{\text{decay time} * \text{sampling rate}}$$




```
 1  INS 0 4
 2  ENV P5 F1 B2 P9 P10 P11 P30 ;
 3  ØSC P8 V1 B3 F3 P29 ;
 4  AD2 P7 B3 B3 ;
 5  ØSC B2 B3 B2 F2 P28 ;
 6  ØUT B2 B1 ;
 7  END ;
 8  GEN 0 1 1 0 0 96 1 128 .7 150 1 175 .6 200 1 225 .7 256 1
      320 .3 384 0 511 0 ;
 9  SV2 0 50 .050 .100 ;
10  SV3 0 1 .15 ;
11  NØT 0 4 .1 54 349 ;
12  NØT .2 4 .1 54 392 ;
13  NØT .5 4 .13 54 440 ;
14  NØT .6 4 .2 54 349 ;
15  NØT 0 4 .8 54 262 ;
```

Fig. 39. Envelope generator ENV for attack and decay. Instrument #4.

The attack time AT and decay time DT will in general be constants. CØNVT calculates the steady-state time SS as the duration P(4) minus the attack and decay times

$$SS = P(4) - AT - DT$$

Thus the steady-state time varies with duration. For short notes there may be no steady state, and the attack and decay times may have to be shortened so that their sum does not exceed the duration. All this must be done by CØNVT.

The data record to evoke ENV is

```
ENV I1, F, Ø, I2, I3, I4, S ;
```

S is a sum that must be assigned temporary storage in some unused note parameter.

In the example shown in Fig. 39, Pass II variables V50 and V51 contain the attack and decay times, respectively. These are set with the SV2 record. The vibrato rate is kept in Pass III V1 and is set with SV3 record. The attack and decay function F1 is computed with GEN1. The attack portion has a slight overshoot for added sharpness. The steady-state portion has two cycles of quaver. The decay portion has two line segments to approximate an exponential.

The NØT records contain amplitude in decibels in P5 and the frequency in hertz in P6.

The instrument requires inputs P5 and P7 through P11, as shown on the diagram. The CØNVT program to compute these inputs from the NØT record is listed and annotated below.

	Text	Notes
	SUBRØUTINE CØNVT	
	CØMMØN IP, P, G	
	DIMENSIØN IP(10), P(100), G(1000)	
	IF (P(1) − 1.0) 105, 100, 105	
100	IF (P(3) − 4.0) 105, 101, 105	
101	COR = 1.0	1
	SS = P(2) − G(50) − G(51)	
	IF (SS) 102, 103, 103	2
102	CØR = P(4)/(G(50) + G(51))	3
	P(10) = 128.	
	GØ TØ 104	
103	P(10) = 128./(G(4) * SS)	4
104	P(9) = 128./(G(4) * G(50) * COR)	5
	P(11) = 128./(G(4) * G(51) * COR)	
	P(5) = 10.0 ** (P(5)/20.0)	6
	P(7) = 511.0 * P(6)/G(4)	
	P(8) = .0075 * P(7)	
	IP(1) = 11	
105	RETURN	
	END	

Notes

1. CØR will correct attack and decay times for short notes where the steady state does not exist.
2. Checks to see if steady state time SS is positive.
3. Steady state is negative. CØR is set to reduce attack and decay times proportionally so that

   AT + DT = duration

   P(10) is set at 128 so that steady-state time will equal one sample, which is the minimum possible steady state.
4. Computation of P(10) for positive steady-state times.
5. Computation of P(9) and P(11) for either positive or zero steady-state times. CØR will be less than 1 for zero steady-state times.
6. The usual computation of amplitude and frequency control. Vibrato amplitude is set at $\frac{3}{4}$ percent of center frequency.

The NØT records (11–15) play the five notes sketched on the staff. Two additional capabilities of the program are inherent in these records. Instrument #4 is used to play up to three voices simultaneously. The second voice is a sustained C_{262}. The third voice occurs because the slurred notes overlap slightly, with the note from record 6 extending into the beginning of the note from record 7. Pass III can play multiple simultaneous voices on any instrument. As many as 30 voices can be played in the training orchestra.

The score records are not written in ascending sequence of action times, in that the C_{262} is written last and starts at t = 0. The order of these records is immaterial, since they will be sorted into the proper ascending sequence of action times in Pass II.

Filter—FLT

One of the more difficult sound-processing operations is filtering. A unit generator that operates as a band-pass filter is shown in Fig. 40. The filter may be used to introduce formants or energy peaks at specified frequencies into sound waves. Such formants are characteristic of many instruments.

The filter is calculated by means of a difference equation. In terms of the diagram shown in Fig. 40, the equation is

$$Ø_i = I1_i + I2_i \cdot Ø_{i-1} - I3_i \cdot Ø_{i-2}$$

I1 is the input to the filter, Ø the output, and I2 and I3 determine the frequency and bandwidth of the passband.

More specifically, the difference equation approximates a 2-pole filter with a pole pair at $-D \pm j\,F$ Hz on the complex frequency plane

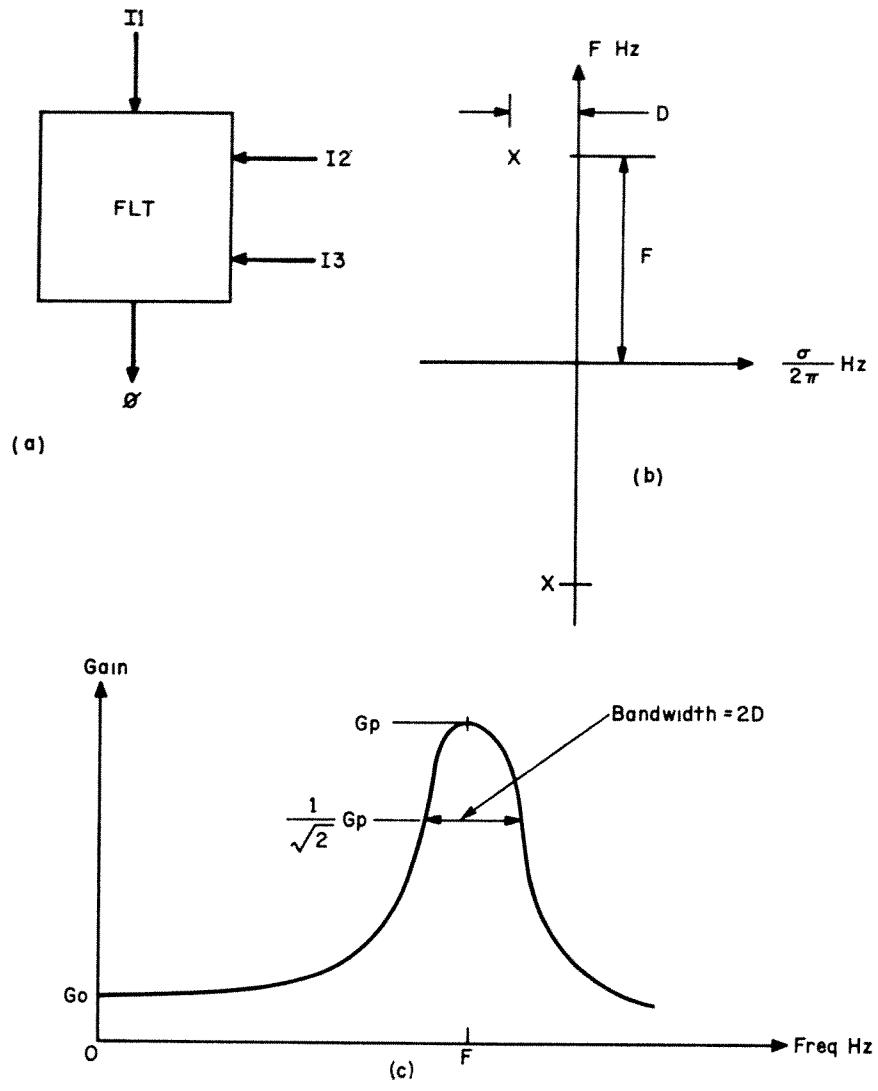


Fig. 40. Band-pass filter—FLT. (a) Diagram: I2 = $2e^{-2\pi D/R} \cos 2\pi F/R$; I3 = $e^{-4\pi D/R}$; (b) poles in complex plane; (c) curve of gain vs. frequency.

as shown in Fig. 40. The approximate gain of the filter is also shown as a function of frequency in Fig. 40, where the peak occurs at F Hz and the bandwidth at the half-power points is 2D Hz. The approximation holds for $F \gg D$. I2 and I3 are determined as functions of F and D by the relations

$$I2 = 2\, e^{-2\pi D/R} \cos 2\pi F/R$$

and

$$I3 = e^{-4\pi D/R}$$

where R is the sampling rate. CØNVT may be conveniently used to compute I2 and I3 from F and D.

The main problem in using the filter is obtaining a reasonable amplitude of output. The dc gain G_0 is given by the equation

$$G_0 = \frac{1}{1 - I2 + I3}$$

and the peak gain is approximately

$$G_p = G_0 \cdot \frac{F}{2D} \qquad \text{for } F \gg D$$

Both gains may be either less than or much greater than unity, depending on F and D. Narrow bandwidths produce high peak gains.

The amplitude of the output depends both on the amplitude of the input and on its frequency composition. A sinusoid near frequency F will be multiplied by G_p. A low-frequency sinusoid will be multiplied by G_0. A complex signal must be decomposed into individual harmonics, the gain for each harmonic computed separately, and the resulting amplified harmonics reassembled at the output. This process is usually impractical, and one approximates the gain as something between G_0 and G_p. Often the approximation is poor and it is necessary to adjust the amplitudes and recompute the samples to avoid overloading or underloading the output. For this reason filters should be used sparingly.

The score record for FLT is

FLT I1, Ø, I2, I3, P_i, P_j

where P_i and P_j are two unused note parameters in which $Ø_{i-1}$ and $Ø_{i-1}$ are stored.

Composing Subroutines—PLF

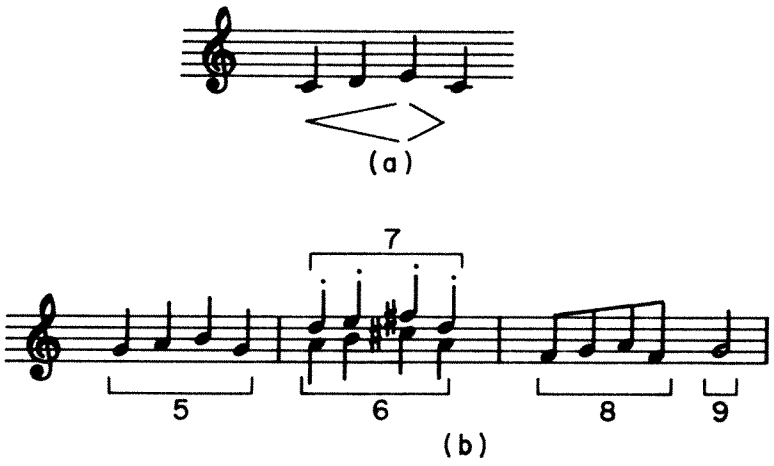
Our tutorial discussion of what might be called the basics of sound generation is now complete. We are ready to take up compositional subroutines that will permit the generation of note parameters by the computer. These are some of the most interesting but difficult directions in which computer sound generation can be developed. Advanced applications point toward complete pieces composed by computer. However, long before these goals are achieved, PLF subroutines will be useful in saving the human composer from much routine work.

So far, for each note to be played, the composer has had to write a line of score starting with NØT.... PLF subroutines will now be developed which write these NØT records. Furthermore, one score record that evokes a PLF subroutine can generate many NØT's.

Although PLF programs can do other things besides compute NØT records, these records are of overriding importance and are the reason for creating PLF's. Moreover, Pass I itself is justified because it serves to contain the PLF subroutines.

Let us develop an example to demonstrate and teach the possibilities and practices of PLF programs. The example will allow storing a group of N notes in Pass I memory. A call to a PLF1, which we shall write, will insert these N notes anywhere in a composition and modify the notes by an arbitrary frequency shift, by an arbitrary tempo shift, and by specifying the instrument on which they will be played.

Figure 41 illustrates the use of PLF1. The first four score records store the four-note pattern in Pass I variable storage, using variables 10 through 43. Like the other passes, Pass I has a general storage array D(2000), which contains 2000 locations in the training orchestra. The




```
1  SV1 0 10 0 1 52 0 ;
2  SV1 0 20 1 1 56 .167 ;
3  SV1 0 30 2 1 60 .333 ;
4  SV1 0 40 3 1 56 0 ;
5  PLF 0 1 10 40 0 1 .583 4 ;
6  PLF 0 1 10 40 4 1 .750 4 ;
7  PLF 0 1 10 40 4 1 1.167 5 ;
8  PLF 0 1 10 40 8 .5 .417 4 ;
9  PLF 0 1 10 10 10 2 .583 4 ;
```

(c)

Fig. 41. PLF note-generating example: (a) pattern; (b) conventional score; (c) computer score.

PLF1 routine will assume that a note is described by four numbers

Starting time in seconds
Duration in seconds
Amplitude in decibels
Frequency in logarithmic units

in four successive D locations. Successive notes in a pattern will be separated by 10 locations in D, so that the first note goes into D(10), the second into D(20), etc., as is accomplished by the SV1 records.

The logarithmic frequency scale is introduced here. The composer will write scale numbers which are related to frequency by the equation

$$\text{Scale} = \log_2\left(\frac{\text{frequency in hertz}}{262.0}\right)$$

This is probably the most useful scale for compositional algorithms. Middle C_{262} is 0, C_{512} is $+1$, etc.; the even-tempered half step is an increment of $\frac{1}{12}$. Thus

$C_{262} = 0$
C# = .083
D = .167
D# = .250
E = .333
F = .417
G = .583
G# = .667
A = .750
A# = .833
B = .917
$C_{512} = 1.000$

The even-tempered standard musical intervals are

half step = $\frac{1}{12}$ = .083
full step = $\frac{1}{6}$ = .167
minor third = $\frac{1}{4}$ = .250
major third = $\frac{1}{3}$ = .333
fourth = $\frac{5}{12}$ = .417
fifth = $\frac{7}{12}$ = .583

Frequency transposition can be done simply by adding a constant to the scale steps of a pattern. Multiplication corresponds to increasing or decreasing the size of the intervals in a pattern. Scales other than 12-tone can be represented with equal facility. The logarithmic scale is

so powerful and appropriate that we will use it almost exclusively from here on.

In Fig. 41 score records 5 through 9 evoke the PLF subroutine that is to be presented. The P fields in these records have the following significance.

- P1 calls a PLF subroutine
- P2 is not used since action time has no importance in Pass I
- P3 identifies the subroutine number (PLF1)
- P4 gives the D location of the first note in the pattern
- P5 gives the D location of the last note in the pattern
- P6 gives the time in seconds at which the pattern should begin
- P7 gives the duration scaling of the pattern; .5 = play at double speed; 2 = play at half speed
- P8 gives the logarithmic interval to shift the frequency of the pattern. For example, P8 = .583 corresponds to shifting the theme up by a fifth
- P9 gives the instrument number on which the pattern should be played

P1 through P3 have the same significance for all PLF routines. The rest of the P's depend entirely on the particular subroutine to be written.

The conventional score for the notes produced by records 5 through 9 is shown in Fig. 41 with the notes coming from a given record identified. Record 5 produces the first four notes in which the pattern is shifted up by a fifth. Records 6 and 7 produce two copies of the pattern playing in fourths. The upper voice is played on instrument 5 which is assumed to yield a staccato timbre. Record 8 plays the pattern at double speed. Record 9 plays the first note of the pattern at half speed.

In order to write a PLF program we will have to know something of the operation of Pass I. It reads the score records in the order in which they appear in the score. The SV1 records cause data to be stored in the D(2000) memory. A NØT record would simply cause the record to be sent on to Pass II. This is accomplished by placing the NØT data in the P(100) array and calling a communication routine WRITE1, which writes out the P array on a file that will later be read by Pass II. For bookkeeping purposes, the number of parameters in the record is kept in another Pass I location IP(1) and is automatically written out by WRITE1. The function of the PLF routine is to generate NØT records and to write them out exactly as Pass I would have done with a record in the score.

How is the PLF routine brought into action? When Pass I reads a PLF score record it calls a subroutine PLFn, in which n is in P3. The rest of the data on the score record is in the P array where it can be used by the subroutine.

The annotated PLF1 routine to perform the computations we have described follows.

	Text	Notes
	SUBRØUTINE PLF1	1
	CØMMØN IP, P, D	
	DIMENSIØN IP(10), P(100), D(2000)	
	NS = P(4)	2
	NE = P(5)	
	TS = P(6)	
	DS = P(7)	
	FS = P(8)	
	IP(1) = 6	3
	P(1) = 1.0	4
	P(3) = P(9)	5
	DØ 100 I = NS, NE, 10	6
	P(2) = TS + DS * D(I)	7
	P(4) = DS * D(I + 1)	8
	P(5) = D(I + 2)	9
	P(6) = (2.0 ** (D(I + 3) + FS)) * 262.0	10
	CALL WRITE1(10)	11
100	CØNTINUE	
	RETURN	
	END	

Notes

1. This CØMMØN and DIMENSIØN statement locates the three essential arrays, IP, P, and D for PLF1. The Pass I definition of these arrays must agree with the definition in the subroutine.
2. These statements take parameters P4–P8 from the PLF data record and store them in the PLF subroutine. Since the P(100) array will be used to output NØT records, the PLF parameters must be removed from it.
3. The word count of the NØT records is set at 6. We will assume that we are generating notes for an instrument of a type shown in Fig. 39. The six fields are

   P1 NØT
   P2 Action time in seconds

P3 Instrument number
P4 Duration in seconds
P5 Amplitude in decibels
P6 Note frequency in hertz

4. The programs convert all alphabetical symbols into numerical equivalents in the initial reading routine in Pass I. All subsequent processing is done on the numbers. The equivalence of NØT is 1.0, which is set by this statement.
5. This statement sets the instrument number into P(3). P(1) and P(3) are constant for all the notes in the pattern and hence can be set once at the beginning.
6. This DØ loop is executed once for each NØT in the pattern. The storage of the pattern in the D array is inherently defined by the loop, the first note beginning at D(NS), the last note beginning at D(NE), the notes being 10 locations apart.
7. This statement computes the starting time of the note as the PLF time shift TS, plus the duration scale DS, times the starting time relative to the beginning of the pattern D(I).
8. This statement computes the scaled duration of the note.
9. This statement transfers the amplitude of the note from the pattern to the P(100) array. No modification of amplitude is necessary.
10. This statement adds the frequency transposition FS to the pattern frequency D(I + 3), and converts the sum from a logarithmic to a linear frequency scale in hertz.
11. This statement calls for writing out the completed NØT record.

In this example it is already possible to see many labor-saving advantages in PLF. Although the pattern is atypically short, 17 notes are produced by only five data records, far fewer than are needed to write a separate NØT record for each note.

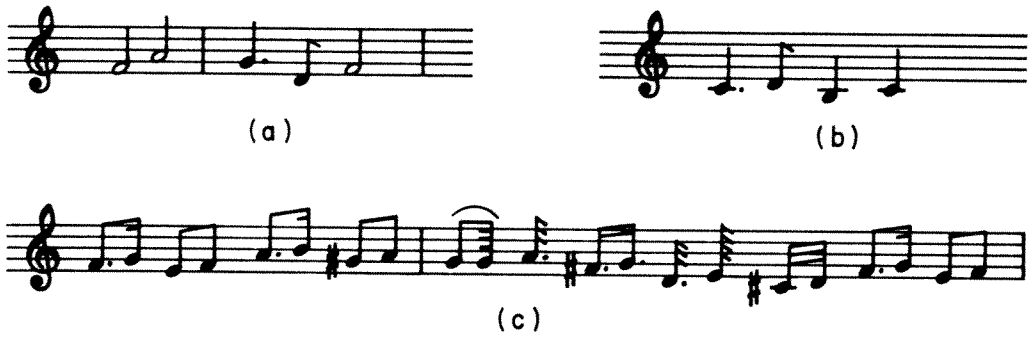
Next let us discuss a slightly more complicated and considerably more interesting PLF routine. It takes the product of two themes, in a sense. Each note in the first theme is replaced by the entire second theme. The second theme is scaled for duration; its total duration exactly equals the duration of the note it replaces. The log frequencies of the second theme are increased by the log frequencies of the replaced note, so that the second theme is centered about each note of the first theme. Amplitudes are similarly treated. The process is reminiscent of some "theme and development" styles. The second theme can be considered an ornament applied to the first theme.

An example is shown in Fig. 42, where the first theme, the second theme, and the product are written in musical notation. Typically, theme 2 is short and compact in frequency range. However, this is not a requirement. We could also form the product

Theme 2 × theme 1

Our multiplication algorithm is not commutative, and

Theme 1 × theme 2 ≠ theme 2 × theme 1




```
 1  SV1 0 10 0 2 50 .417 ;
 2  SV1 0 20 2 2 53 .750 ;
 3  SV1 0 30 4 1.5 56 .583 ;
 4  SV1 0 40 5.5 .5 59 .167 ;
 5  SV1 0 50 6 2 62 .417 ;
 6  SV1 0 60 0 1.08 4 0 ;
 7  SV1 0 70 1.5 .375 2 .167 ;
 8  SV1 0 80 2 .75 0 -.083 ;
 9  SV1 0 90 3 .75 -2 0 ;
10  PLF 0 2 10 50 60 90 0 4 ;
```

(d)

Fig. 42. Multiplication of two themes: (a) theme 1 times (b) theme 2 gives (c) product via PLF2; (d) computer score.

The score for the example is also shown in Fig. 42. Lines 1–5 define theme 1, lines 6–9 define theme 2, and line 10 calls PLF2 to generate the product. The calling sequence is

P4 D location of first note of first theme
P5 D location of last note of first theme
P6 D location of first note of second theme
P7 D location of last note of second theme
P8 Starting time of product theme
P9 Instrument number

We will assume that the D array arrangement and the instrument are the same as were used for PLF1.

The annotated FØRTRAN program follows.

	Text	Notes
	SUBRØUTINE PLF2	
	CØMMØN IP, P, D	
	DIMENSIØN IP(10), P(100), D(2000)	
	NB1 = P(4)	1
	NE1 = P(5)	
	NB2 = P(6)	
	NE2 = P(7)	
	TS = P(8)	
	IP(1) = 6	
	P(1) = 1.0	
	P(3) = P(9)	
	DØ 101 I = NB1, NE1, 10	2
	START = TS + D(I)	3
	DS = D(I + 1)/(D(NE2) + D(NE2 + 1))	
	DØ 100 J = NB2, NE2, 10	2
	P(2) = START + DS * D(J)	4
	P(4) = DS * D(J + 1)	
	P(5) = D(J + 2) + D(I + 2)	
	P(6) = (2.0 ** (D(J + 3) + D(I + 3)) * 262.0	
	CALL WRITE1(10)	
100	CØNTINUE	
101	CØNTINUE	
	RETURN	
	END	

Notes

1. This group of statements moves the PLF parameters from the P(100) array into the subroutine and sets the unchanging parts of the NØT parameters in P(100) in preparation for writing NØT records.
2. The program contains two nested DØ loops. The outer loop is executed once for each note in theme 1, the inner DØ cycles for each note in theme 2.
3. These two statements compute the starting time shift and the duration scaling for a repetition of theme 2. START is the beginning time of a note in theme 1. DS is computed so that the last note in theme 2 will end at the ending time of the note in theme 1.
4. This and the following three statements compute the starting times, durations, amplitudes, and frequencies for the notes in theme 2

which are replacing a single note in theme 1. Amplitudes in decibels and frequencies on a logarithmic scale are simply added. Frequencies are converted to hertz.

These two examples give only a slight indication of the range of objectives that may be programmed with PLF routines. The routines are not limited to generating NØT records. They may also be used to manipulate the information stored in the D array of Pass I. A powerful application is a set of PLF routines, each of which effects a different transform on a set of notes stored in the D array. Since the result of each transform must be in the same form as its input, several transforms may be successively applied. Finally, a last PLF writes out the NØT records. A composition using these subroutines would consist of the description of some thematic material, plus a long sequence of PLF calls to manipulate this material.

The PLF routines provide one of the most exciting areas for further development in the entire Music V structure. Not only do they promise the most interesting possibilities, but they also offer the greatest challenges to the composer's creativity.

Compositional Functions

The note-generating subroutines that have just been demonstrated can be greatly strengthened by defining information in certain ways which we call compositional functions. Compositional functions can be used to provide a new language to describe sounds, called a graphic score. Although graphic scores can be used to represent the notes in a conventional score, the notation is completely different. It is more powerful in the sense that many sounds that are impossible to notate conventionally can be readily described by a graphic score. Moreover, in the synthesis of sounds, the graphic scores can be "read" easily by note-generating subroutines. This section can only lay the foundation for graphic scores, but further information is given in the references.

A compositional function is a function defined over an entire section of a composition. It is used to control some always present parameter such as loudness or tempo. Compositional functions should not be confused with the stored function used to describe waveshape or envelope. The stored functions are generated, stored, and used in Pass III. Compositional functions are described and used in the first two passes. Both their mode of description and their use differ from those of stored functions.

Metronome Function

Let us start by considering the metronome function which is built into Pass II and can be evoked if desired. So far, the starting times and durations of notes have been written in numbers which were interpreted as seconds. Thus a note

```
NØT 2 4 1 54 .167 ;
```

starts at 2 sec from the beginning of the section and lasts for 1 sec. With a metronome function, P2 and P4 are interpreted in beats; the note starts at the beginning of the second beat of the section and lasts for one beat. The relation between beats and seconds is given by the metronome function, which is in standard metronome marking of beats per minute. Thus, for example, if the metronome function is 180, the note would start $\frac{2}{3}$ sec from the beginning of the section and would last $\frac{1}{3}$ sec.

The metronome function need not be constant, but can change abruptly or gradually during a section to introduce accelerandos or retards. The operation can be illustrated by an example shown in Fig. 43.

The conventional score for 14 quarter notes lasting 14 beats is shown at the top, together with tempo marking. The NØT cards to encode this are as follows.

```
NØT   0   4   .8   60   0    ;
NØT   1   4   .8   60   .167 ;
NØT   2   4   .8   60   .333 ;
NØT   3   4   .8   60   0    ;
NØT   4   4   .8   60   0    ;
NØT   5   4   .8   60   .167 ;
NØT   6   4   .8   60   .333 ;
NØT   7   4   .8   60   .417 ;
NØT   8   4   .8   60   .583 ;
NØT   9   4   .8   60   .750 ;
NØT  10   4   .8   60   .917 ;
NØT  11   4   .8   60   .583 ;
NØT  12   4   .8   60   .583 ;
NØT  13   4   .8   60   .750 ;
```

The metronome function is evoked and is stored in Pass II variable storage by the records

```
SV2 0 50 0 60 4 60 8 120 11.9 120 12 30 14 30 ;
SV2 0 2 50 ;
```

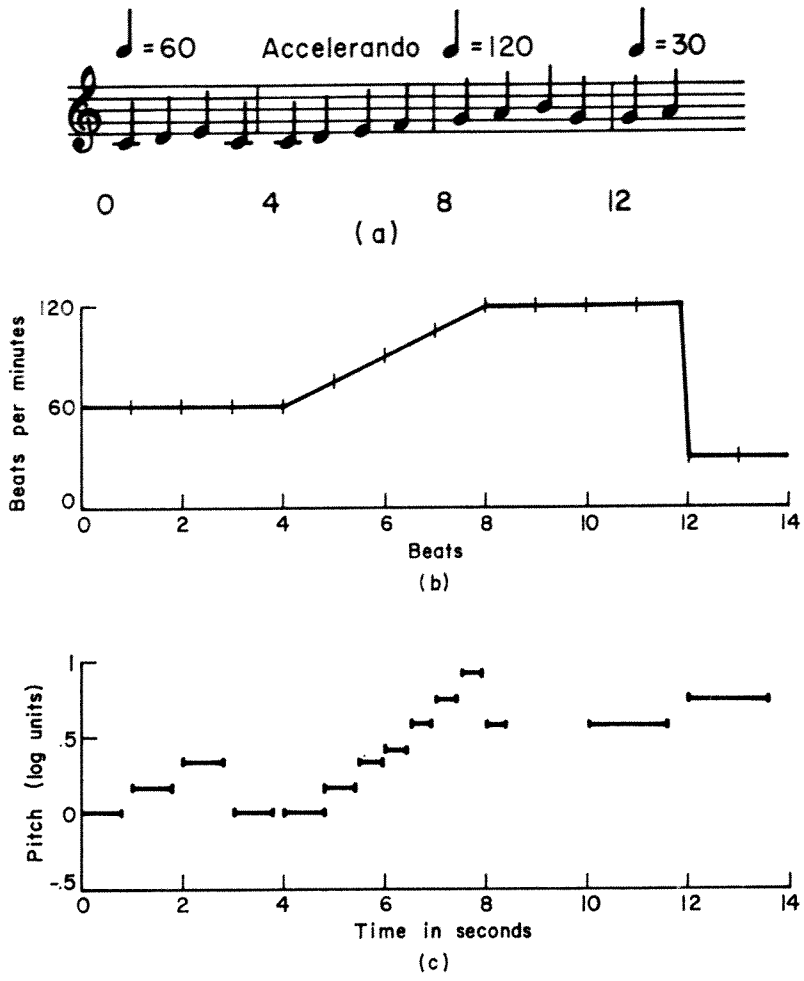


Fig. 43. Metronome function: (a) music score; (b) metronome marking function; (c) graphic score.

The first record describes the function, and P4 gives the initial abscissa (0), P5 the ordinate at that abscissa (60), P6 the next abscissa (4), P7 the next ordinate (60), etc. The abscissa is in beats and the ordinate in metronome marking—beats per minute. Successive points on the function are connected by straight lines, as shown in Fig. 43. As many segments as desired may be used by putting more points into the SV2 function. The abscissa points need not be uniformly arranged.

The second record tells the Pass II program that a metronome function is being used and that it starts in variable 50, that is to say, in G(50).

The graphic score in Fig. 43 shows the notes resulting from the metronome function being applied to the score. The pitch of each note, plotted against the time it occurs, is shown by the horizontal bars. Pitch is given on a logarithmic scale, 0 = middle C, +1 = C_{512}. Time

is in seconds. Such a graphic score has proved to be an effective way of displaying many computer note sequences.

The first four notes, at a tempo of 60, occupy the first 4 sec. The second measure is played at an increasing tempo from 4 to 6.5 sec. The third measure at a tempo of 120 lasts from 6.5 to 8.5 sec. The last two notes, at a tempo of 30, go from 10 to 14 sec.

The computation relating metronome function to a note's starting time and duration consists in sampling the metronome function at the beginning of the note. These sampling points are indicated by ticks on the function. Thus the sixth note has a tempo value of 75. The duration and starting time of the note are defined as

$$\text{Duration} = \text{P4} \times \tfrac{60}{75} \text{ sec}$$
$$= .8 \times \tfrac{60}{75} = .64 \text{ sec}$$

and

$$\text{Starting time} = \text{starting time of previous note}$$
$$+ (\text{P2 of note} - \text{P2 of previous note}) \cdot \tfrac{60}{75}$$
$$= 4 + (5 - 4) \cdot \tfrac{60}{75} = 4.8 \text{ sec}$$

Because the metronome function for the 13th note is sampled at 12 beats, its value is 30, and a rather long silence occurs between the 12th and 13th notes. Such a silence is inherent in the algorithm. It is seldom objectionable; also such large changes in tempo seldom occur.

If several voices are playing simultaneously, the same metronome function is applied to all.

Metronome functions have proved to be powerful tools for inserting accelerandos and ritardandos. Without them, the calculation of starting times and durations of gradually changing note sequences can be very tedious. In addition, they enable the composer to write in terms of conveniently defined beats, rather than seconds, which often turn out to be unwieldy decimals.

A Note-Generating Subroutine and Graphic Score

The metronome function is built into Pass II. Let us now write a subroutine PLF3, which will generate a voice from a complete, if elementary, graphic score. A sample score is shown in Fig. 44. Four functions are used to describe the voice—duration, duty factor, pitch, and amplitude. All are functions of time, which in this example goes from 0 to 13 sec.

The duration function in Fig. 44a gives the time from the beginning of one note to the beginning of the following note. The first note starts

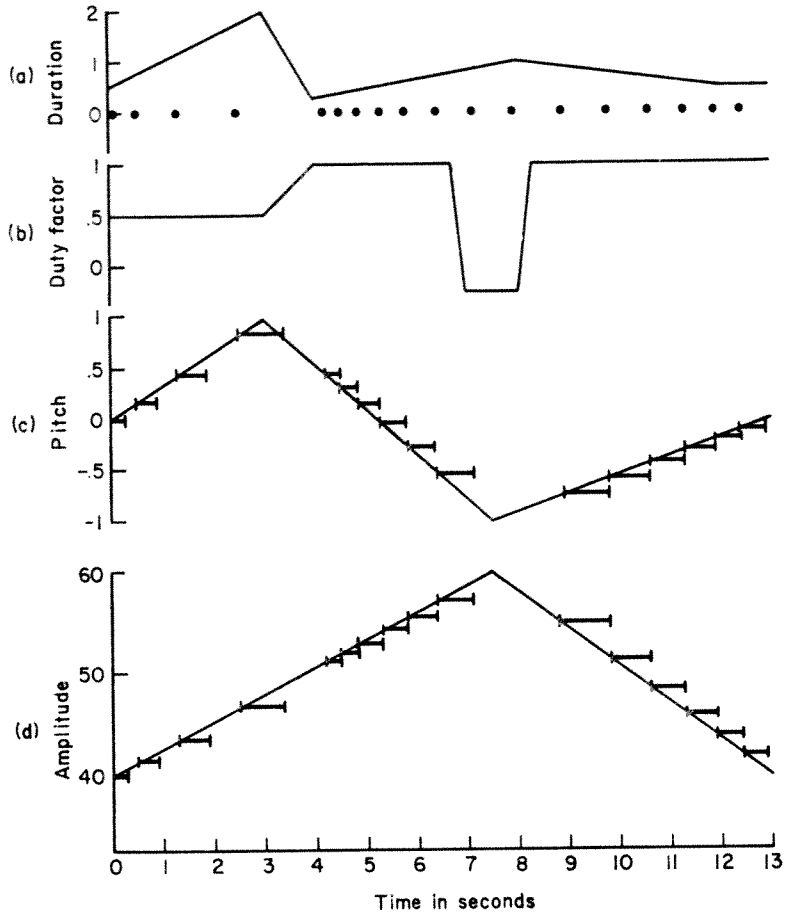


Fig. 44. Graphic score and resulting notes: (a) duration; (b) duty factor; (c) pitch; and (d) amplitude.

at t = 0, where the value of the duration function is .5 sec. The second note starts at .5 sec, where the value of the duration function is .8 sec; the third note starts at 1.3 sec (1.3 = .5 + .8), and so on. In other words, the duration function is sampled at the beginning of each note to obtain the interval to the beginning of the next note. The actual sampling times are shown as dots along the abscissa of Fig. 44a. Although there are more advanced ways of representing durations graphically, this representation is easy to program and is useful for a number of purposes.

The rhythm is represented by the duration function. The style of playing—legato or staccato—is represented by the duty-factor function, Fig. 44b, which gives the proportion of the interval between the starting times of successive notes that is occupied by sound. The first note has a duty factor of .5. Its length will be .25 sec (.5 duty factor × .5-sec interval to start of the second note). A duty factor of .25 produces an

exceedingly staccato note sequence. A duty factor of 1 produces a smooth legato. A duty factor greater than 1 causes overlap of successive notes. Like the duration function, the duty-factor function is sampled. The duty factor for each note is the value of the duty-factor function at the starting time of the note. Duty-factor function will be somewhat arbitrarily used for one other purpose. A negative duty factor will indicate a rest—the note will be omitted entirely.

The pitch and amplitude functions, Fig. 44c and d, are also sampled at the starting time of each note to obtain pitch and amplitude. Pitch is written in our logarithm scale, $C_{262} = 0$, $C_{512} = 1$, etc. Amplitude is written in decibels. A graphic representation of the actual notes that are generated are shown as horizontal bars superimposed on the pitch and amplitude functions. Pitches can be read from the pitch scale, amplitudes from the amplitude scale, durations and starting times from the beginning and ending times of either the pitch or amplitude bars. Because of the sampling process, the left end of each bar starts on the pitch or amplitude function.

The functions shown graphically in Fig. 44 must be represented numerically in the computer memory in Pass I. The convention that represented the metronome function will again be employed; successive breakpoints will be given by their abscissa and ordinate values. The data will be stored in the Pass I D(2000) array by SV1 cards. Thus the duration, duty factor, pitch, and amplitude functions in Fig. 44 are described by the following four records

```
Duration:    SV1 0 50 0 .5 3 2 4 .3 8 1 12 .5 13 .5 ;
Duty factor: SV1 0 65 0 .5 3 .5 4 1 6.7 1 7 -.25 8 -.25
               8.3 1 13 1 ;
Pitch:       SV1 0 85 0 0 3 1 7.5 -1 13 0 ;
Amplitude:   SV1 0 95 0 40 7.5 60 13 40 ;
```

The functions start in D(50). The amount of memory occupied by a function depends on the number of breakpoints. Successive functions have been arbitrarily spaced by sufficient multiples of five so as not to overlap.

A FØRTRAN function CØN has been provided to read the graphic functions. The statement

```
Z = CØN (D, N, T)
```

sets Z equal to the value, at time T, of the function that starts at D(N).

Thus, for example,

```
Z = CØN (D, 95, 3.0)
```

would set Z equal to 48, which is the value of the amplitude function at 3 sec. The values are computed by interpolating a straight line between the breakpoints that surround T (0 and 7.5 in the specific example). CØN must search the D array to find these breakpoints. It is essential not to ask for values of the function outside the range of breakpoints that have been defined. Otherwise CØN may never terminate its search.

We can now write a subroutine PLF3 to generate notes from graphic scores. The data record to call this routine is

```
PLF 0 3 TS END NA NP NDR NDF IN ;
```

where TS is a time shift giving the starting time of the sequence of notes to be produced by PLF3; END is the duration of the sequence; NA, NP, NDR, and NDF give the starting points in the D array of the amplitude, pitch, duration, and duty-factor functions, respectively. IN gives the instrument number.

An example to produce the notes shown in Fig. 44 is:

```
PLF 0 3 0 13 95 85 50 65 4 ;
```

We shall assume that the program writes out records in the form (which has been used frequently)

```
NØT TS IN D AMP PITCH ;
```

where AMP and PITCH are in decibels and log units.

The annotated PLF3 subroutine follows.

Text	Notes
SUBRØUTINE PLF3	
CØMMØN IP, P, D	
DIMENSIØN IP(10), P(100), D(2000)	
TS = P(4)	
END = P(5)	1
NA = P(6)	
NP = P(7)	
NDR = P(8)	
NDF = P(9)	
P(1) = 1.0	2
P(3) = P(10)	
IP(1) = 6	
T = 0.0	3

```
100  DR = CØN(D, NDR, T)                                    4
     IF(T + DR − END) 101, 101, 104                          5
101  P(2) = T + TS                                           6
     P(4) = DR * CØN(D, NDF, T)                              7
     IF(P(4)) 103, 103, 102                                  8
102  P(5) = CØN(D, NA, T)                                    9
     P(6) = CØN(D, NP, T)
     CALL WRITE1(10)
103  T = T + DR                                             10
     GØ TØ 100                                              11
104  RETURN
     END
```

Notes

1. These statements extract the essential information for the PLF3 from the P array.
2. These statements set the constant parts of the P array in preparation for writing out NØT records, and they set the word count.
3. T is the starting time of the next note to be generated (not including the time shift TS). It is set initially at zero and computed as a running variable and is increased by the interval between successive notes after each note is generated. T is also the variable used to specify abscissa values in CØN.
4. DR is the interval between successive notes as obtained by CØN from the duration function.
5. This statement checks to see whether the starting time of the next note is greater than the ending time, END. If so, the current note is not generated and PLF3 is terminated.
6. The time shift TS is added to T to obtain the starting time of the note.
7. The duration of the note is computed as the interval times the duty factor.
8. This statement checks for a rest. If the duration is zero or negative, owing to a zero or negative duty factor, no NØT is written out and the program proceeds to the next note.
9. These statements compute the rest of the NØT parameters and write out the NØT record.
10. This statement adds T to the starting time of the next note.
11. This statement transfers control in order to generate the next note.

Several features of the operation of PLF3 may be pointed out in Fig. 44. The first four notes are staccato, having large silences between

notes. The next six notes are legato, with no silent intervals between notes. Two possible notes have been omitted to form a rest.

In terms of the number of notes generated, PLF3 is very efficient. One PLF3 call produced 16 notes. It could just as well have produced 1600. In contrast to conventional scores, the notation for duration has the advantage that a second's worth of fast notes requires no more effort to describe than a second's worth of slow notes. Also ritardandos and accelerandos are easy to describe by lines with increasing or decreasing slopes, as illustrated from 4 to 8 and from 8 to 13 sec. Such tempo changes can have striking acoustical effect.

Pass II Subroutines—PLS Pitch-Quantizing Example

To complete the discussion of NØT-generating subroutines and note-manipulating subroutines, we will write one second-pass subroutine. Pass II routines cannot be used to generate additional notes since all the NØT records have been carefully sorted in increasing order of action times, and the addition of more NØT records would disrupt the ordering. However, PLS routines can change the values of note parameters (except action times). Since notes of all voices are sorted together, it is convenient for PLS to embody relations involving several voices at a particular time. For example, PLS could well be used to adjust the pitch intervals between voices.

We will not attempt quite as complicated an example as interval control. Instead we will control the pitches of a single voice so that they fall exactly on the steps of a previously specified scale. Such a process makes sense when applied to the output of the PLF3 routine that was presented in the preceding section. The pitches so generated are samples of a continuous pitch function and can fall anywhere. Sometimes it is desirable to limit the possible pitches to a prespecified set or scale. The scale need not correspond to any known or standard musical scale, such as a just scale or a 12-tone scale. An octave can be divided into any number of intervals; the intervals can be even tempered (equal) or unequal in size.

Figure 45 gives an example of the output of the routine to be written. It is applied to the notes generated by the PLF3 program. The pitch function and notes from Fig. 44 have been redrawn in Fig. 45. For the scale the octave is divided into five equal intervals, as shown in Fig. 45. Since pitch is in logarithmic units, these units correspond to equal musical intervals. The PLS routine will adjust the pitches of the notes generated by PLS to the closest scale step. The pitches generated by

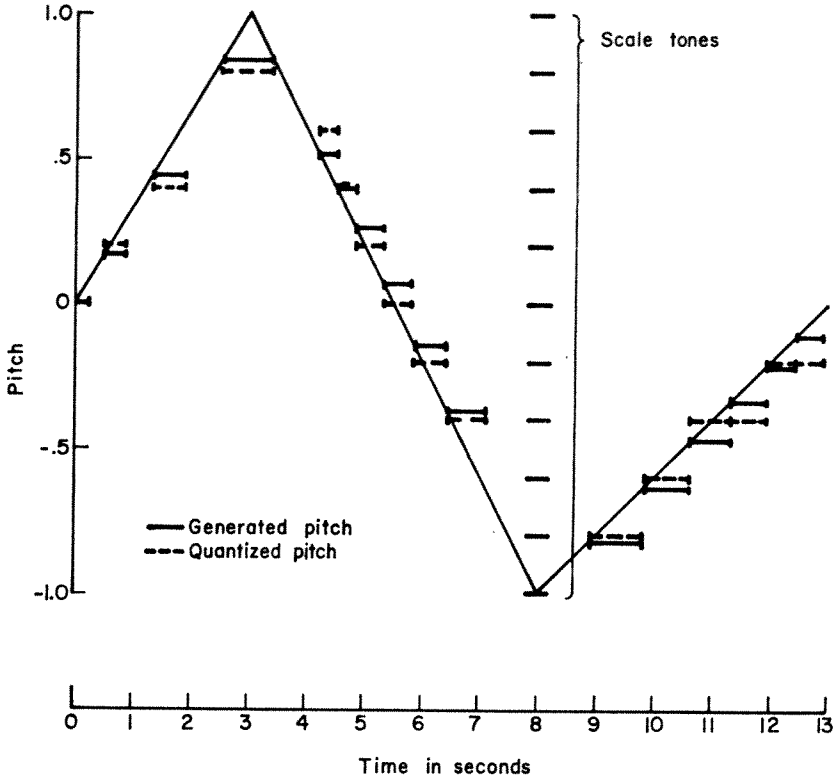


Fig. 45. Pitch quantizing by a PLS routine.

PLF are shown as solid horizontal bars, and the adjusted pitches are shown as dashed bars. The adjustment may be either up or down, depending on which scale step is closest. The process of adjustment is called pitch quantizing.

Note that the first and sixth notes happened to fall exactly on a scale step and require no quantizing. Also the last two pairs of notes become pairs of repeated notes as a result of quantizing. Quantizing tends to produce repeated notes if the scale steps are large and the change in pitch between successive notes is small.

In order to write a PLS routine, it is necessary to understand a few details of the operation of Pass II. All the data records in a section are read into a large array D(10,000), 10,000 locations long in the training orchestra. An array I(1000) is computed by sorting so that

I(1) = the address in D of the beginning of "first" data record where "first" means smallest action time

I(2) = the address in D of the beginning of "second" data record

etc.

For example if the thirteenth data record is

NØT 19 2 4 60 .167 ;

and is stored starting at D(109), then

I(13) = 109

and

D(109) = 6 (Word count)
D(110) = 1 (NØT ≡ 1)
D(111) = 19
D(112) = 2
D(113) = 4
D(114) = 60
D(115) = .167

After I(1000) is computed, the program goes through the data records in order of increasing action times, executing any PLS routines, storing any SV2 data in the G(1000) Pass II data array, and writing out NØT records with the aid of the CØNVT subroutine.

A PLS function can modify any NØT records with action times greater than the action time of the PLS function. It cannot affect NØT records with action times less than the PLS function, since these will already have been written before PLS is executed. The Pass II memory G(1000) will contain the numbers from any SV2 cards with action times less than the action time on the PLS function; it will not contain any data from SV2 cards whose action times are greater than on the PLS function.

The scale will be stored in Pass II memory by a SV2 statement giving the number of steps in the scale, followed by the pitches of these steps. Thus the scale used in Fig. 45 is inserted in the memory by the record

SV2 0 100 11 −1 −.8 −.6 −.4 −.2 0 .2 .4 .6 .8 1 ;

These data will go into G memory at time 0.

The PLS1 function will be called at action time 0 by the statement

PLS 0 1 100;

where P3 = 1 indicates PLS1 and the 100 gives the starting point of the scale in the G array.

An annotated program to carry out the pitch quantizing follows.

	Text	Notes
	SUBRØUTINE PLS1	
	CØMMØN IP, P, G, I, T, D	1
	DIMENSIØN IP(10), P(100), G(1000), I(1000), T(1000), D(10,000)	
	I1 = IP(2)	2
	IN = IP(3)	3
	NQ = D(I1 + 4)	4
	NB = NQ + 1	
	NL = NQ + IFIX(G(NQ))	
	DØ 103 J = 1, IN	5
	ID = I(J)	
	IF(D(ID + 1) − 1.0) 103, 100, 103	6
100	FREQ = D(ID + 6)	7
	MIN = 1,000,000.0	8
	DØ 102 K = NB, NL	
	IF(ABS(FREQ − G(K)) − MIN) 101, 102, 102	
101	MIN = ABS(FREQ − G(K))	
	QFREQ = G(K)	
102	CØNTINUE	
	D(ID + 6) = QFREQ	9
103	CØNTINUE	
	RETURN	
	END	

Notes

1. This common statement and the subsequent dimension statement describe the main data arrays in Pass II and must agree with the corresponding statements in the Pass II main program. IP gives certain miscellaneous constants, P is the communication array from which data records are read and written, G is general variable storage, I indexes the D array in action-time order, T contains action times and is primarily used in the sorting process, D contains the data records.
2. When PLS is called, IP(2) contains the address in the D array at which the PLS data record is located. In this case if

   IP(2) = 27

   then

   D(27) = 4 (Word count)
   D(28) = 10 (PLS ≡ 10)
   D(29) = 0
   D(30) = 1
   D(31) = 100

3. IP(3) contains the number of data records in D. The main DØ loop in the PLS routine will examine I(J) for J = 1 to IP(3).
4. This and the subsequent two statements determine NB and NL as the first and last locations of the scale description in the G array. A DØ loop will test these locations.
5. The main DØ loop examines all data statements in order of ascending action times. ID in the subsequent statement is the D address of the data statement.
6. This statement checks to see if the data statement is a NØT record. If not, it is skipped; if so, the pitch variable P6 is quantized.
7. FREQ is set equal to the pitch P6.
8. This and the subsequent statements to 102 determine the scale step that is closest to FREQ. MIN is initially set to a very large value. The absolute value of (FREQ − each scale step) is compared with MIN and if it is smaller than MIN, MIN is reset to that value. In this way MIN ends being the smallest interval and QFREQ ends being the closest frequency.
9. This statement resets the pitch D(ID + 6) to the closest scale step.

The PLS routines tend to be both longer and logically more complicated than the PLF routines. The steps in the example just discussed are typical. Actually, they were not all necessary for the problem at hand. The pitches could have been quantized by the PLF routine as they were generated. Even if the quantizing were done in Pass II, it would not have been necessary to go through the D array in order of action times. However, for slightly more complicated operations, such as quantizing the intervals between voices, all the Pass II steps are essential.

Another simplification in the program consists in writing out the scale for all the octaves in which it is to be used. In many cases, only one octave is written out; the actual pitches are translated to this octave before being quantized; and the quantized pitches are translated back to their original octave. The possibilities open to the composer are almost endless.

Interactions Between Instruments

The final process to be considered in this chapter involves interactions between instruments. The desirability for such interactions arises from the limitations of the "note concept," which defines sounds as having starting and ending times. Sometimes it is desirable to produce continuous sounds that change from time to time in controlled ways. As

we shall show, this can be done by using the output of an instrument as an input to another instrument. The first instrument is degenerate in the sense that it produces no acoustic output. Instead it plays a series of notes that generate a long and frequently changing modulation function for the second instrument. The second instrument may play only a single long note whose sound is varied by the parameter supplied by the first instrument.

A typical and important use of interactions is amplitude control to produce swells and diminuendos as notated on the conventional score in Fig. 46. Such a modulation is unwieldy to program with the apparatus previously described. Although we can draw continuous amplitude

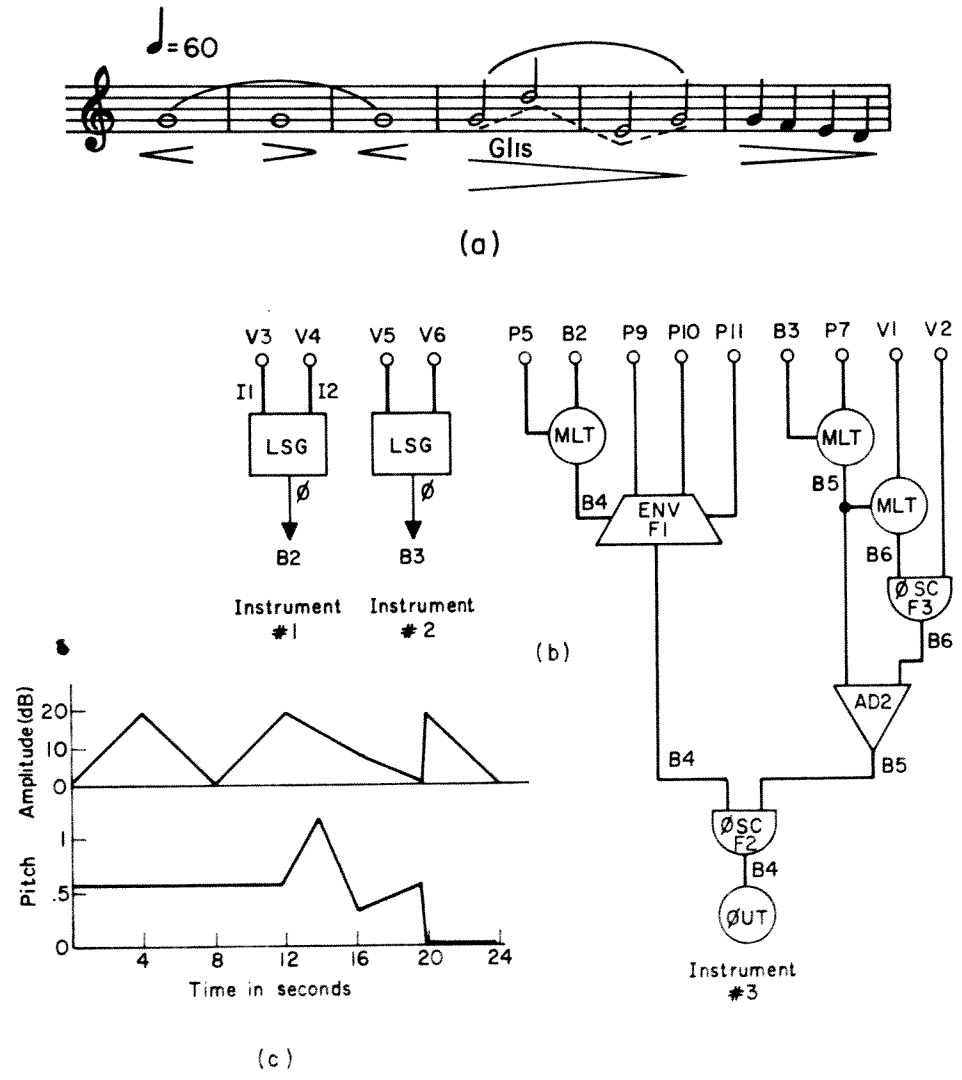


Fig. 46. Interconnected instruments for amplitude and glissando control: (a) musical score; (b) block diagrams of instruments; (c) continuous control functions.

functions, as is done in Fig. 44, these are sampled at the beginning of each note and that amplitude is held constant for the duration of the note; this effect is clearly not the desired objective of Fig. 46. Furthermore, since the inputs to instruments are set at the beginning of each note, the sampling effect is hard to avoid. In Fig. 46a, measures 4 and 5 also call for a combination of glissando with amplitude variation. The last measure applies a continuously changing amplitude control to a sequence of notes. As an example of interacting instruments, we shall produce these effects.

The apparatus for interactions is the input–output blocks B1–B10, which are shared by all instruments. The output of instrument A may be left in a block for subsequent use by instrument B. This requires that A be computed before B. Pass III computes the instruments in order of their numbers,[3] any that are #1 first, then any that are #2, and so forth. Hence by making A a lower numbered instrument than B, the proper order can be guaranteed. An additional requirement is that the block used for communication cannot be used for other purposes which would overwrite the output of A before B uses it. Also, in contrast to most instruments, A can generate only one voice at a time.

A special unit generator LSG, which rapidly computes functions formed from straight-line segments, is useful in instruments that generate control functions. Two such generators are used for instruments 1 and 2 in Fig. 46 to produce amplitude- and frequency-control signals, respectively. The operation of LSG is simply

$$I1_i = I1_i + I2_i$$
$$Ø_i = I1_i$$

or in other words I1 is incremented by I2 for each sample and Ø is equal to I1. Because only addition is involved, the process is rapid. I1 will be set to a desired initial value and I2 to the slope of a linear function that starts at I1. I1 and I2 can be reset at any time, thus changing the value of Ø and the slope abruptly. In instrument #1, Pass III variables 3 and 4 are used for I1 and I2. These will be set with SV3 records which are generated by a Pass I subroutine PLF4 to achieve a particular amplitude-control function. Instrument #2 produces the same effect for pitch. The outputs of instruments 1 and 2 are put in blocks B2 and B3 where they form inputs to instrument 3.

Instrument 3 is a modification of the envelope instrument which was

[3] As of February 21, 1968, this feature was not yet programmed in Music V. However, it seems both desirable and easy to insert.

developed in Fig. 39, and it uses the CØNVT function for that instrument. The additional amplitude function B2 is multiplied by the normal amplitude input P5. The continuous amplitude-control function is written in decibels (as shown in Fig. 46c), and B2 is the exponential transformation

$$B2 = 10 ** \left(\frac{\text{continuous amplitude function}}{20}\right)$$

Thus, the decibels of the normal amplitude function and the continuous function are additive. If the continuous function is 10 dB and the normal function is 50 dB, the resulting sound will be at 60 dB.

The normal frequency input P7 is multiplied by the additional frequency-control function B3. The continuous pitch function (also shown in Fig. 46c) will be written in our standard logarithmic scale, and B3 will be the exponential transformation

B3 = 2 ** continuous pitch function

Thus a continuous pitch function of 0 produces no change in pitch, a continuous pitch function of 1 produces a one-octave upward shift, and so forth. The computation of V3–V6 to achieve both the exponential conversions and the proper increments will be done by a PLF4 subroutine.

Input V1 specifies the proportion of frequency shift in the vibrato, proportionality being controlled by a multiplier. Such Pass III multiplication is essential rather than multiplication by the CØNVT function, because frequency can vary over a note.

The annotated PLF4 program is given below. The pitch and amplitude functions will be stored as Pass I variables in the usual notation. The functions shown in Fig. 46c are stored by the statements

```
SV1 0 50 0 0 4 20 8 0 12 20 19.99 0 20 20 24 0 ;
SV1 0 70 0 .583 12 .583 14 1.167 16 .333 19.99 .583 20 0
  24 0 ;
```

The calling record for PLF4 is

```
PLF 0 4 TS END FA FP ;
```

where TS is the starting time of the control functions, END is the duration of the control functions, FA is the starting variable of the amplitude function, and FP is the starting variable of the pitch function. For the example

```
PLF 0 4 0 24 50 70 ;
```

is the specific calling record.

PLF4 generates a sequence of SV3 records to form the inputs to instruments 1 and 2, and generates two NØT records to activate these instruments from 0 to 24 sec.

	Text	Notes
	SUBRØUTINE PLF4	
	CØMMØN IP, P, D	
	DIMENSION IP(10), P(100), D(2000)	
	TS = P(4)	1
	END = P(5)	
	NA = P(6)	
	NP = P(7)	
	I = NA	2
	IP(1) = 5	
	P(1) = 4.0	
	P(3) = 3.0	
100	P(4) = 10.0 ** (D(I + 1)/20.0)	3
	P(5) = (10.0 ** (D(I + 3)/20.0) − P(4))/((D(I + 2) − D(I)) * D(4))	
	P(2) = TS + D(I)	4
	CALL WRITE1(10)	
	IF (D(I + 2) − END) 101, 102, 102	5
101	I = I + 2	6
	GØ TØ 100	
102	I = NP	
	P(3) = 5.0	
103	P(4) = 2.0 ** D(I + 1)	
	P(5) = ((2.0 ** D(I + 3) − P(4))/((D(I + 2) − D(I)) * D(4))	
	P(2) = TS + D(I)	
	CALL WRITE1(10)	
	IF(D(I + 2) − END) 104, 105, 105	
104	I = I + 2	
	GØ TØ 103	
105	IP(1) = 4	8
	P(1) = 1.0	
	P(2) = TS	
	P(3) = 1.0	
	P(4) = END	
	CALL WRITE1(10)	
	P(3) = 2.0	
	CALL WRITE1(10)	
	RETURN	
	END	

Notes

1. These statements extract the essential information for PLF4 from the P array.
2. These statements prepare to write SV3 records for V3 and V4. P(1) is 4 for SV3. P(3) = 3.0 designates V3 as the first variable. One pair of V3 and V4 values will be written for each segment of the amplitude function. I = NA will set the initial value of the equations starting at 100 for the first segment.
3. This and the subsequent line calculate the initial value and slope for the first segment. The slope is in units per sample. D(4) is the sampling rate.
4. The time of the SV3 card is the beginning time of the first segment plus TS.
5. This statement terminates the amplitude function at the end of the current segment if D(I + 2) ≥ END.
6. I is incremented by 2 and control is transferred to 100 to continue with the next segment.
7. These statements write out SV3 records for variables 5 and 6 to produce the pitch control. The process is exactly analogous to amplitude control.
8. The rest of the program writes out two NØT records

```
NØT TS 1 END
NØT TS 2 END
```

that play two notes on instruments 1 and 2 which start at TS and have duration END.

The score records to produce the Fig. 46 output are given below. The definition of the instruments and the Pass III stored functions are omitted since they are completely standard.

```
SV1 0 50 0 0 4 20 8 0 12 20 19.99 0 20 20 24 0 ;
SV1 0 70 0 .583 12 .583 14 1.167 16 .333 19.99 .583 20 0
  24 0 ;
PLF 0 4 0 24 50 70 ;
NØT 0 3 11.8 40 262 ;
NØT 12 3 7.8 40 262 ;
NØT 20 3 .8 40 392 ;
NØT 21 3 .8 40 349 ;
NØT 22 3 .8 40 330 ;
NØT 23 3 .8 40 294 ;
```

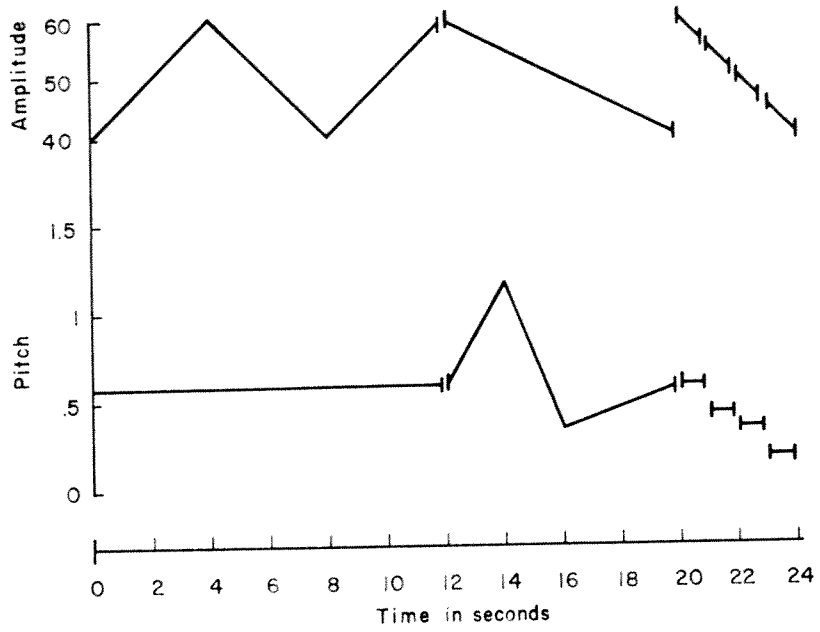


Fig. 47. Graphic score with continuous changes in amplitude and pitch.

The graphic score of the resulting sound is shown in Fig. 47. Beginnings and ends of individual notes are indicated by short vertical bars on the amplitude and pitch curves. An attack or decay will be produced by the envelope generator at these times. Amplitude and pitch changes occur continuously and independently of note boundaries.

Parting Exhortations to the Student

The tutorial examples are now complete. However, the student's task—harnessing the computer to his objectives—has just begun. A mere reading of the examples is not sufficient to master their content. The examples are a far from complete description of the Music V program; the Music V program as written will produce only a fraction of the wanted and achievable computer sounds.

Programming skills come only with practice. The problems accompanying this chapter provide some possibilities for practice. Their solution by the student is greatly recommended. If limited time forces a choice between reading the chapter and working the problems, working the problems is to be preferred.

Most of the problems are based on the material given in the chapter. In some cases, more details about the Music V program must be obtained. These can be found in the Music V Handbook which

forms the next part of this book. The handbook is intended to be a complete description of the program, arranged and indexed for references. Information about Music V that has been given in this chapter is also presented in the handbook, and is usually easier to find there. The student should become accustomed to answering his questions from the handbook as soon as possible.

Although it is not necessary to read the entire handbook to make use of its information, anyone who plans to make extensive use of Music V should read the sections describing the operations of the various parts of the programs. Detailed block diagrams as well as verbal descriptions of operations are included. Reading the handbook is a helpful preparation for reading the programs themselves.

Music V is written almost entirely in FØRTRAN. Consequently, it is practical to read the programs and understand their operation. Such understanding is essential if major modifications of the programs are to be made. The advanced user will want to make such modifications; Music V was written with this objective in mind. Hence the student's final teacher, and the final arbiter of questions about the operation of Music V, is the programs. Such is the nature of computer programs.

Annotated References by Subject

Computers in General

J. Bernstein, *The Analytical Engine* (Random House, New York, 1964). A non-mathematical and elementary introduction to computers and what they can do.

A. Hassitt, *Computer Programming and Computer Systems* (Academic Press, New York, 1967). A discussion of programming from an elementary to an advanced viewpoint.

Fortran Programming

S. C. Plumb, *Introduction to Fortran* (McGraw-Hill, New York, 1964).

E. I. Organick, *Fortran IV Primer* (Addison-Wesley, Reading, Mass., 1966).

S. V. Pollack, *A Guide to Fortran IV* (Columbia University Press, New York, 1965).

These are three self-instructional texts that teach Fortran.

Graphic Scores

M. V. Mathews and L. Rosler, "Graphic Scores," *Perspectives of New Music 6*, No. 2 (1968). A detailed article illustrating one technique for composing with the aid of a computer.

Problems for Chapter 2

Parameters of Training Orchestra

Sampling rate—20,000 Hz
Function block length—512
Number of functions—10

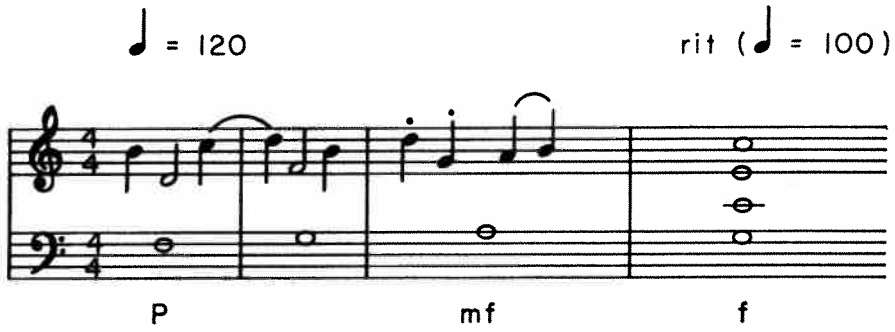
Range of functions——$-1 < F < +1$
I-Ø block length—512
Number of I-Ø blocks—10
Range of unit generator inputs and outputs——2047 to +2047
Maximum number of note parameters—30
Number of Pass III variables—200
Maximum number of voices—30
Pass II G array length—1000
Pass I D array length—2000

Even-Tempered Scale

Note	Frequency in hertz	Logarithmic pitch
C	262	0
C#	277	.083
D	294	.167
D#	311	.250
E	330	.333
F	349	.417
F#	370	.500
G	392	.583
G#	415	.667
A	440	.750
A#	466	.833
B	494	.917

Introductory Score-Writing Problem

1. Using the orchestra defined in Fig. 27 write the computer score for the following conventional score.



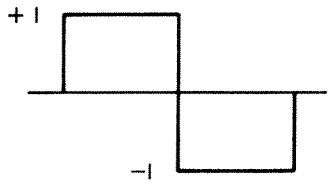
Assume that in amplitude, p ≈ 50, mf ≈ 150, and f ≈ 500. Assume that staccato notes sound for .5 the nominal time occupied by the note (for example, a staccato quarter note at a tempo of 120 would sound for .25 sec). Legato notes sound for .8 of their nominal time, and slurred notes for 1.1 of their nominal time. (Remember that a Music V instrument can play

more than one note at a time. The limit in the training orchestra is 30 simultaneous voices.)

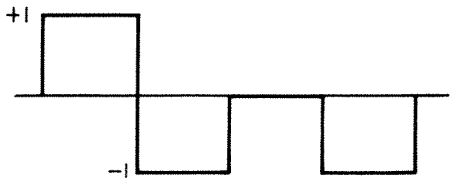
Simple Unit Generators

2. Write out the samples $F_n(j)$ $j = 0...511$ for the following stored functions. To shorten your answers, use ... to indicate a sequence of identical samples.

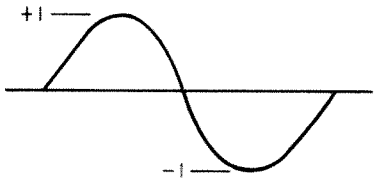
(a) F4, a symmetrical square wave with amplitude +1 or −1



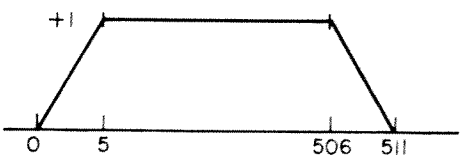
(b) F5, a triple pulse wave as shown



(c) F6, a sine wave of peak amplitude 1 (write only the first 20 samples)



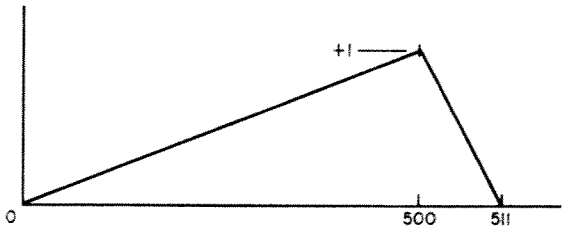
(d) F7, an attack function with shape



3. For an oscillator with

I1 = 500
I2 = 50.35

and a function F1



calculate S_i, $[S_i]_{\text{Mod } 511}$, $F([S_i]_{\text{Mod } 511})$

and $\emptyset_i$ for $i = 0 \ldots 15$

Assume $[S_i]_{\text{Mod } 511}$ is rounded to the next lower integer in looking up values of F1. Calculate the truncation error in $\emptyset_i$ due to rounding for $i = 0 \ldots 15$.

4. Instrument 1 consists of only one oscillator

ØSC P5 P6 B2 F4 P30 ;

where F4 is the symmetrical square wave defined in problem 2a. Write the samples B2(1)...B2(20) generated by the following notes

(a) NØT 0 1 .001 1000 250 ;
(b) NØT .002 1 .001 1000 50 ;
(c) NØT .004 1 .001 500 128.3 ;
(d) NØT .006 1 .001 1000 600 ;
(e) The numerical frequency of the last note is

$$\frac{600}{511} \cdot 20{,}000 = 23{,}500 \text{ Hz}$$

This frequency is much greater than half the sampling rate. What is the apparent period of B2(1)...B2(20)? This period (about 6 samples) results from foldover.

5. Instrument 1 shown in Fig. 27 uses the symmetrical square wave of problem 2a for its stored function. Write the output samples $S_0, S_1, \ldots, S_{60,000}$ resulting from the following score. Abbreviate your answer by designating blocks of zero samples by ...

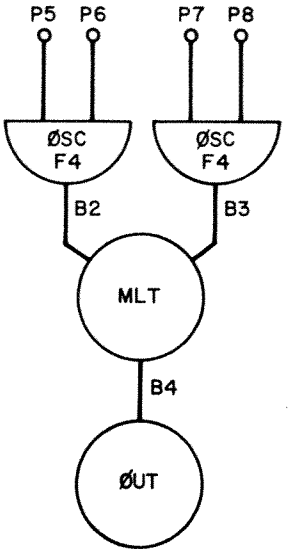
NØT 0 1 .0005 1000 60 ;
NØT .5 1 .0006 500 200 ;
NØT 1 1 .0002 100 10 ;
NØT 2 1 .001 500 70 ;
NØT 2.0002 1 .0004 500 100 ;

6. Instrument 2 shown below uses F4 function of problem 2a. It plays the note

NØT 0 2 .001 500 80 3 100 ;

Plot the samples

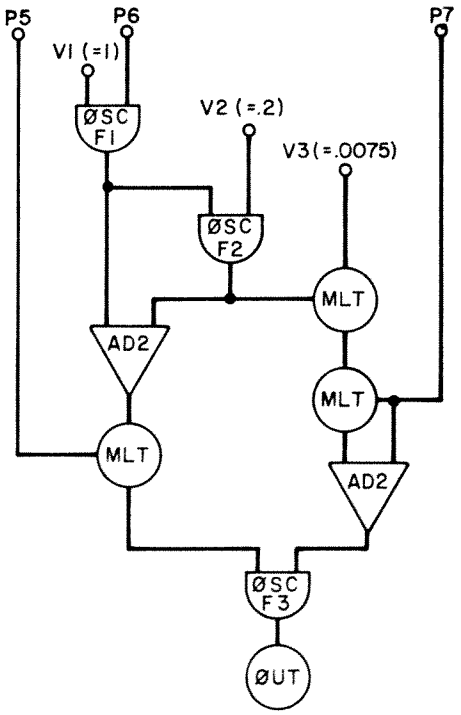
B2(1)...B2(20)
B3(1)...B3(20)
B4(1)...B4(20)



Instrument 2

Simple Instruments

7. Score the instrument diagrammed here.



(a) What do P5, P6, and P7 control?
(b) What is the % vibrato?
(c) % amplitude variation?
(d) What is the rate of vibrato?

(e) Write score records for F1, an attack and decay function; F2, a vibrato function; and F3, a modified square-wave waveform.

(f) Write the score for the following passage

8. Diagram, score, and write functions and a note for an instrument that has attack and decay in amplitude and a frequency attack. The frequency of each note should start 10% low and rise linearly to the final frequency of the note within the first 10% of the note's duration.

9. Diagram and score an instrument with attack and decay in amplitude, with vibrato, and with attack and decay on the vibrato.

10. Diagram an instrument that uses four ØSC's to change the waveform of a note as a function of both amplitude and frequency. The composition of the output waveform should be

$$A \cdot [\{1000 - f\}\{(1 - A)ØSC_1 + A \cdot ØSC_2\} + \{f\}\{(1 - A)ØSC_3 + A \cdot ØSC_4\}]$$

where A is an amplitude control going from 0 to 1 and f is frequency in hertz.

CØNVT Functions

11. Write a CØNVT function for the instrument shown in Fig. 32 which will process a note record of the form

NØT T 2 D A F ;

where A is amplitude in decibels and F is frequency in hertz. V50 is the proportion of vibrato. For each NØT record, CØNVT should write out three records to produce a three-note chord, the highest voice having a frequency of A Hz, the middle voice A/2 Hz, and the lowest voice A/4 Hz.

12. Write a CØNVT function for the instrument shown in Fig. 33 which reads a NØT record of the form

NØT T 3 D A_1 A_2 . . . A_n Freq

where $A_1 \ldots A_n$ is a sequence of amplitudes in decibels and Freq is frequency in hertz. The CØNVT function outputs n + 1 successive notes of equal duration, whose total duration is D. The first note starts at amplitude 0 (linear scale) and ends at A_1(dB), the second goes from A_1 to A_2, . . ., the last goes from A_n(dB) to 0 (linear scale).

Additional Unit Generators

13. Design an instrument with an amplitude-modulated band-pass noise having a bandwidth equal to $\frac{1}{4}$ the center frequency of the noise band, and a noise band whose center frequency changes linearly from an initial to a final frequency during each note.

14. Design an instrument having a random amplitude variation of $\pm 50\%$ of the average amplitude and a low-pass spectrum going from 0 to 15 Hz.

15. Design an instrument producing a band-pass noise by both frequency and amplitude modulation. Have the center frequency of the noise band controlled by P7 and the (bandwidth/center frequency) ratio by P8.

16. Write a CØNVT function for the instrument shown in Fig. 39 which will generate notes with an attack time of .1 sec and a decay time of .2 sec, provided the note duration is greater than .3 sec. The steady-state time should be (duration − .3) sec. For notes of duration between .2 and .3 sec, the attack time should be .1 sec and the decay time (duration − .1) sec. Any durations less than .2 sec should be increased to .2 sec.

17. Compute I2 and I3 for filters with a center frequency of 500 Hz and bandwidths of 2 Hz, 10 Hz, 50 Hz, and 260 Hz. What is the dc gain of these filters? What is the peak frequency gain? What is the maximum input signal that will not cause the output to exceed 2048?

Composing Subroutines

18. Write a set of PLF routines that will process note data in Pass I memory. Assume that the note data are stored in the Pass I D array in the manner used for the Fig. 41 example, and that notes will be written for the instrument shown in Fig. 39. Write the following subroutines:

(a) PLF1 rewrites n notes in the D array, multiplying all logarithmic pitch intervals by S, adding a constant K to the logarithmic pitch intervals, and changing the tempo by a factor T.
(b) PLF2 substitutes a new note for note n in the array.
(c) PLF3 makes a copy of n notes starting at D(m) and stores the copied notes at D(p), overwriting anything that was previously at D(p).
(d) PLF4 divides each of n notes starting at D(m) into k notes of equal length whose total duration equals that of the note they replace. The new notes are written starting at D(p).
(e) PLF5 writes NØT records for n notes starting at D(m). The starting times of all notes are shifted by T sec.

Use these subroutines to compute a composition.

Graphic Scores

19. Write a subroutine PLF1 that will generate pitch and amplitude functions as the computed functions

$$\text{Pitch}(t) = f_1(t) * f_2(t) + f_3(t) * f_4(t)$$
$$\text{Amplitude}(t) = f_5(t) * f_6(t) + f_7(t) * f_8(t)$$

where $f_1(t)$ through $f_8(t)$ are functions stored in the D array. Compute the starting and stopping times of notes as the positive-going zero crossings and the negative-going zero crossings, respectively, of a function

$$\text{Notes}(t) = f_9(t) * f_{10}(t) + (1 - f_9(t)) * f_{12}(t)$$

where $f_9(t)$, $f_{10}(t)$, $f_{12}(t)$ are stored in the D array. Let $f_{10}(t)$ and $f_{12}(t)$ correspond to the rhythmic sequence of two well-known melodies. What notes will be generated when $f_9(t) = 1$?; when $f_9(t) = 0$?; when $f_9(t)$ has some intermediate value? Follow the general procedures used in the Fig. 44 example.

Pitch Quantizing

20. Write PLS1, a pitch-quantizing routine which will quantize a voice for instrument 1 into the closest note in the C major scale. Assume that voices for instruments 2 and 3 produce notes in synchrony with instrument 1. Adjust these voices to harmonize instrument 1 according to the following rules.

(a) Harmonize C and E with the chord CEG.
(b) Harmonize F and A with the chord FAC.
(c) Harmonize B and D with GBD.
(d) Harmonize G with CEG if it starts on a multiple of four beats and with GED if it starts on any other beat.

Use a minimum adjustment of the other voices to achieve these chords.

Interconnected Instruments

21. Define an orchestra and an appropriate CØNVT function so that the output of an instrument is the sum of two ØSC's, the proportion of each being determined by two separate instruments I1 and I2. The proportion will change continuously and frequently during the course of the notes to add interest to the sound quality. Use LSG unit generators and follow the general procedures of the Fig. 46 example.