GROOVE—A Program to Compose, Store, and Edit Functions of Time

M. V. Mathews and F. R. Moore
Bell Telephone Laboratories, Murray Hill, New Jersey

A program which makes possible creating, storing, reproducing, and editing functions of time is described. The functions are typical of those generated by human beings. Multiple functions (up to 14) are produced for long periods of time (up to several hours) at sufficiently high sampling rates to describe fast human reactions (up to 200 samples per second). The functions can be used for a variety of purposes such as the control of machine tools or sound synthesizers or anything a person normally controls.

The program operates on a small computer (DDP-224). Functions are stored on a disk file. Functions may be created by real-time human inputs to the computer which can interact with already stored functions and computed functions. Real-time feedback from the process being controlled is an important link in the system. The environment for effective man-machine interaction has been carefully nurtured.

KEY WORDS AND PHRASES: computer music, music, real-time control, digital control, time functions, interactive software, hybrid systems, conductor program
CR CATEGORIES: 3.44, 3.8, 7.3

Objectives and Concepts

Many tasks now done by people are best described simply by one or more functions of time. To list only a few examples, the control of machine tools, the control of plants such as rolling mills or chemical processes, the control of the body itself, speaking, and playing music can all be characterized by a suitable set of time functions. If these tasks are to be automated, the basic job of the computer is to create the functions. Hence it seems worthwhile to develop a general program to produce time functions which are characteristic of human actions. GROOVE is the first attempt at such a program. It can generate multiple functions (up to 14) for long periods of time (up to several hours) with bandwidths sufficient to reproduce fast human responses (up to 100Hz).

To create time functions, several facilities must be provided. Some are obvious. Methods for generating and storing the functions are necessary. Equally important are methods for changing or editing the stored functions. The resulting concept is a "file system" for time functions and such a system is central to GROOVE.

A second concept is feedback control. People inevitably use their sensory inputs to control their motor activities in a feedback operation. In creating time functions on the computer, GROOVE provides opportunity for immediate feedback from observations of the effects of time functions to computer inputs which compose the functions. In the composing mode of the GROOVE system, a human being is in the feedback loop as shown in Figure 1. Thus he is able to modify the functions instantaneously as a result of his observations of their effects.

It is also possible with GROOVE to have feedback directly from the analog device to the computer. Such feedback is an essential part of some programs, for example those used to control mechanical arms. We have chosen to emphasize the human rather than the direct feedback. Systems without human links require more intelligent programs, or from a different viewpoint, with the addition of a human link, we can currently do more complex tasks. In addition, the authors, who wrote the program, enjoy not only being in the loop but retaining command.

The final concept is more nebulous. Since GROOVE is a man-computer system, the human engineering of the system is most important. For example, we discovered

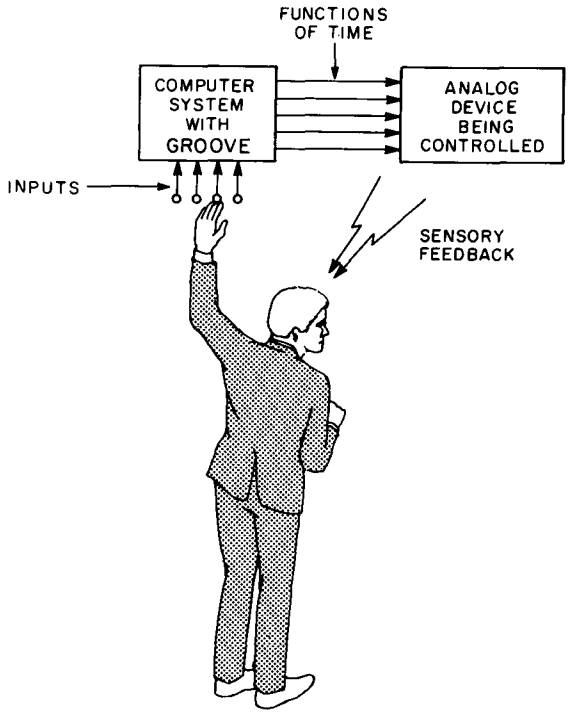

Fig. 1. Feedback loop for composing functions of time

that the control of program time[1] needs to be quite different for composing than for editing, and the program was modified accordingly. Human engineering has affected the entire structure of GROOVE in ways which will be pointed out subsequently.

The Conductor Program

Although GROOVE is a general purpose program, it has been initially used to control an electronic music synthesizer consisting of oscillators whose frequency is set by a voltage and amplifiers whose gain is likewise voltage controlled. Originally we had thought simply of attaching an organ keyboard to a DDP-224 computer which was being used to study speech synthesis and was equipped with 14 digital-to-analog converter outputs. In this way we hoped to make possible the nuances of real-time performance in computer music. However, with a simple program there seemed to be greater danger of imposing on the computer the limitations of the organ rather than improving the organ by means of its association with the computer. Further thought convinced us that the desired relation between the performer and the computer is not that between the player and his instrument, but rather that between the conductor and the orchestra. The conductor does not personally play every note in the score; instead he influences (hopefully controls) the way in which the instrumentalists play the notes. The computer performer should not attempt to define the entire sound in real-time. Instead, the computer should have a score and the performer should influence the way in which the score is played. His modes of influence can be much more varied than that of a conventional conductor who primarily controls tempo, loudness, and style. He can, for example, insert an additional voice of his own, or part of a voice such as the pitch line while the computer supplies the rhythm. He should also be able to modify or edit the score. The computer should not only remember the score, but also all the conductor's functions, so when he achieves a desired performance, it can subsequently be replayed by the computer from memory. These concepts led directly into the GROOVE program for composing and editing time functions.

Computer System for GROOVE

Figure 2 shows a block diagram of the equipment on which GROOVE is run and Figure 3 shows a picture of the facility. The DDP-224 is a medium sized computer with a 16 000, 24-bit, 1.7 μs word memory. Heavy use is made of two memory-access channels which are independent of the central processor and of each other. The main secondary memory is a CDC-9432 disk file with removable disk packs, which transmits 1200 computer words in 30 ms. A typewriter provides control input. The magnetic tape is used only as backup memory for the disk file.

[1] The abscissa of the functions of time which are the principal outputs of GROOVE is called program time.

Several special devices, originally devoleped for speech synthesis are utilized by GROOVE. Twelve 8-bit and two 12-bit digital-to-analog converters form the principal outputs of GROOVE. Two additional converters supply the X and Y deflection voltages to a cathode-ray tube which displays points or characters via a character generator. An analog-to-digital converter plus multiplexor allows sampling up to 20 voltages in about 10 μs per voltage. At present GROOVE inputs 7 voltages. Four come from rotary potentiometers or knobs, which may be turned by the operator in real-time. Three come from a 3-dimensional linear wand shown projecting up from a square box [Figure 3(a)]. These real-time inputs are called knob inputs.

An external oscillator in the equipment rack on the left controls the sampling rate of the output functions by means of one of the interrupt lines on the DDP-224. The sampling rate is also controlled by a knob—the frequency dial on the oscillator.

The two loudspeakers hanging on the wall provide the perceptual feedback to the operator. The oscilloscope (center) resembles a medium sized television screen. The acoustic partitioning situated behind the typewriter and computer console keeps the operating environment isolated from the sounds of the computer (air-conditioning, etc). Through the viewing window [Figure 3(b)] may be seen the magnetic tape unit and the disk. These devices may be remote-controlled from the operator's console, so that except to mount or dismount either digital magnetic tapes or disk packs the operator need never enter the actual computer room.

The specially-built keyboard input device is shown sitting to the right of the typewriter. Each key has a potentiometer associated with it, so it may be set to any desired group of discrete input values. An intercom is used to communicate with another room containing the voltage-controlled equipment.

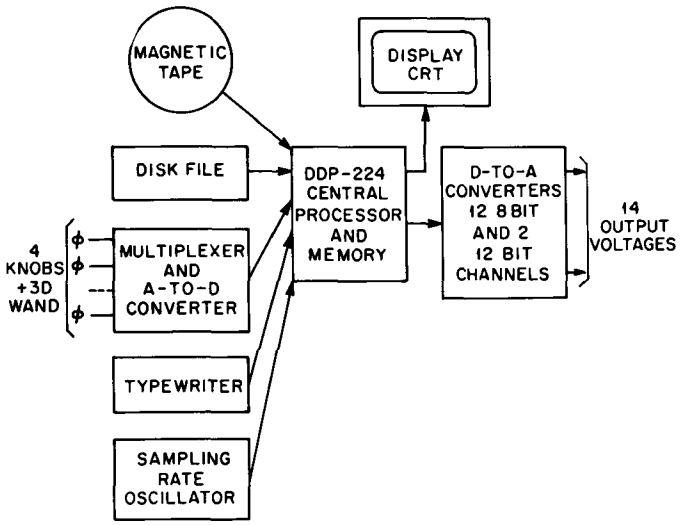

FIG. 2. Block diagram of GROOVE computer system

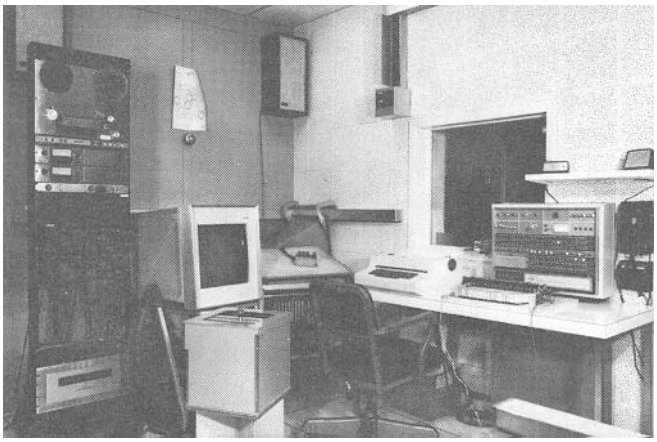

FIG. 3(a). GROOVE facility

FIG. 3(b). Operator's console

Figure 3(c) is a close-up view of the four knob inputs, K1, K2, K3, and K4. The small box on which they are mounted also has 4 toggle and 2 push-button type sense switches, which may be used to communicate with the program in real-time. Pushing box sense-switch one (BSSW1) causes real-time processing to stop and the typewriter to request a command. BSSW3 is used to put the program in the "edit mode" described later. Other switches are undefined and may be used to control the user-supplied portion of the GROOVE program.

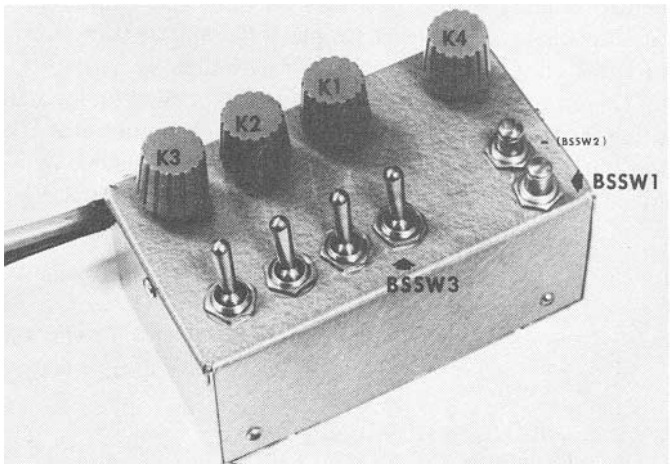

FIG. 3(c). Knob inputs

General Operation of the GROOVE Program

The objective of the GROOVE program is to read samples of functions stored on a memory file at a rate determined by a sampling rate oscillator, to combine these with samples of knob functions which are generated in real-time, to compute and put out samples of output functions in real-time, and if desired, to record revised functions on the memory file.

The primary storage medium for time functions is the disk file. Successive samples of n functions, denoted $T1$ through Tn, where n can be set by the program to values between 2 and 40, are kept in 1200-word disk records and are called disk functions. The general mode of operation is as follows.

(1) Read one record of disk functions into core memory.

(2) Unpack the current n samples of disk functions $T1, \cdots, Tn$ upon receipt of a pulse from the sampling rate oscillator.

(3) Compute the 14 output functions and revised values of the disk functions. Data for the computation consists of the current samples of the disk functions plus current samples of the knob inputs plus current samples of any periodic functions which may be defined and will be discussed below. An algebraic expression interpreter which will handle Fortran-like expressions is part of GROOVE and is used to define the relations which are compiled.

(4) Output the current values of the 14 output functions.

(5) Pack the n revised samples of the disk functions and write these onto disk in place of the original disk functions.

The program configurations embodying these functions is shown in Figure 4. Operation begins in a typewriter command section in which the various parameters of the program are set; the algebraic statements defining output voltages and revised disk functions are written, and simple periodic functions are defined. At a start command from the typewriter the program enables the sampling oscillator interrupt line and enters the "update CRT display" loop. A sense switch returns control to the typewriter at any time to revise parameters or terminate the program.

The program continues in the CRT display loop until interrupted by the sampling rate oscillator. It then goes through the computations to output and revise one sample's worth of functions as discussed above and returns

to the CRT loop. The amount of time spent in the sample computations depends on the number of disk functions and the complexity of the algebraic statements. The amount of time available depends on the sampling rate. Any extra time between interrupts is spent in the CRT loop, and hence, as the sampling rate is increased, the display deteriorates first before the output functions are affected. By observing the display, the operator can avoid failures due to "speeding." The most complex computations so far tried with 14 disk functions have allowed sampling rates up to 150 Hz, which is quite comfortable.

Reading and writing disk functions are both buffered and overlapped so no interruption in outputting or display is involved at the end of a disk record.

Arithmetic Expression

One of the advantages gained by treating control signals as functions of time is that they may then be operated on mathematically. It is easy to imagine relatively complex control signals which are really only the sums, differences, products, etc., of several simple functions of time. Many of these simple functions can be periodic.

Therefore, the GROOVE system has a facility for handling arithmetic expressions which determine how the various inputs to the system are to be combined, and a facility for defining many types of periodic functions. Each

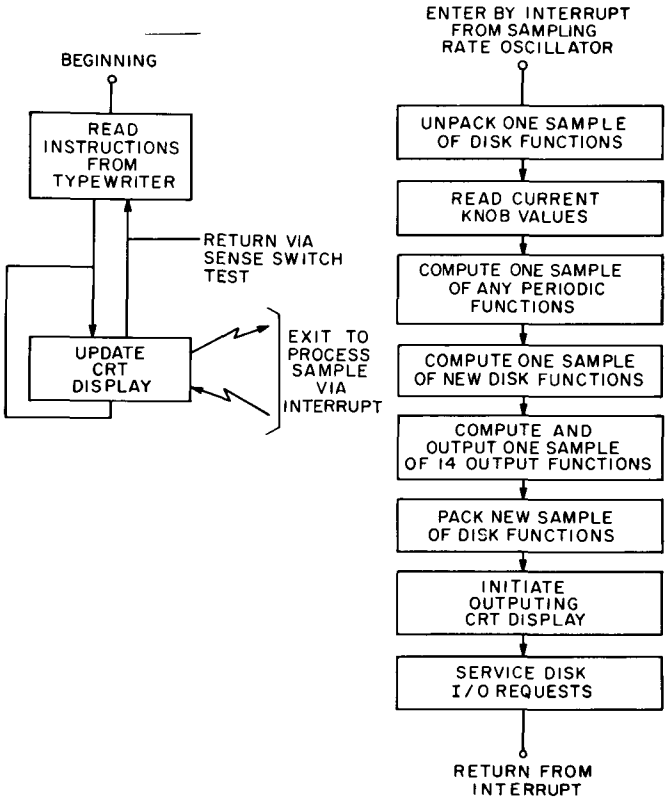

FIG. 4. Block diagram of GROOVE program

arithmetic expression which is typed in at the on-line typewriter is an assignment statement. Some examples of typical statements are

$T6.\ T2 + 480,$
$T1.\ T1,$
$T13:\ -F1 * (K3 - 2000),$
$T28:\ 4095/(1 + U1(F1 + U2(T1 + K2))).$

The left-hand side of this statement is one of $T1, T2, \cdots, T40$, which specifies one of the disk functions which was retrieved from the computer's disk memory. The right-hand side of the statement is any arithmetic expression made up of the four standard arithmetic operators (+, −, *, and /), any number of balanced pairs of parentheses, and the following operands:

(1) $T1$–$T40$, which refer to the current value of a disk function,

(2) $K1$–$K7$, which refer to the current value of some real-time input device, such as a knob or keyboard, and

(3) $F1$–$F40$, which refer to the current value of a periodic function.

In addition, the right-hand expression may include a notation of the form $U1$ ($\langle X \rangle$), where $U1$ refers to the first of 95 possible user-supplied arithmetic functions, and $\langle X \rangle$ is any allowable arithmetic expression. The user-supplied functions may carry out such operations as exponentiation and quantization of a function which are not provided for in the basic GROOVE arithmetic. These user functions insure that the mathematical capabilities of the system can be as powerful as necessary for a given task, without providing for every possible operation in advance.

The operator which relates the right- and left-hand sides of a GROOVE equation is either a period or a colon. The period means "Replace the value of the specified disk function with the value of the expression, but do *not* change the permanent record of this disk function" (on the disk unit). The colon both assigns the expression value to the disk function and *does* change the permanent record of this disk function accordingly. Thus GROOVE has the facility to edit disk functions on a trial basis (the period case). If the results of such an edit are found by observation to be desirable, the period is changed to a colon, and the function is permanently altered. Other ways of editing disk functions are discussed later.

Periodic Functions

Since periodic functions are extremely useful in building more complicated functions, GROOVE's facility for handling them is fairly extensive. When $F1$–$F40$ is typed in an arithmetic expression, it is assumed that the characteristics of this function either have been previously defined or will be defined before real-time processing is started. Essentially, three different kinds of periodic functions can be handled at present: (1) functions consisting of one or more joined line segments (ramp functions); (2) functions con-

sisting of one or more discrete values (step functions); and (3) an arbitrary-length piece of some time function from disk memory (core functions). Examples of the three kinds of functions together with the statements used to define them are shown in Figure 5. Just as for the algebraic expressions, typing a command at the on-line typewriter allows a periodic function to be defined. For ramp and step functions, this is a matter of

(1) specifying what function is being defined,

(2) specifying what type (ramp or step) of function it is, and

(3) typing a list of coordinates or time-value pairs.

The ramp function interpreter automatically interpolates between the points typed; the step function interpreter treats each point as the right-hand end of an interval with the value specified. Any number of points may be used to define a function; their abscissa values need not be uniformly spaced.

For a core function, the definition portion of the function statement consists of

(1) A number specifying the length of the function, and

(2) A notation specifying where the function values will come from ($T1$–$T40$).

Any pieces of a previously generated time function may then be combined to form a periodic function. Core functions occupy much core memory since each sample of the function must be stored individually in contrast to ramp and step functions for which only the coordinates of the points need be stored. By packing two samples per computer word, up to 8000 samples can be accommodated in the DDP-224 corresponding to 80 seconds of a function at 100 samples/second. This is a usable, if not a copious amount.

The abscissa value at which the periodic functions are evaluated is normally incremented by one unit for each output sample. However, this increment can be set equal to any constant, knob value, or disk function. In this way the period of the function in output samples can be changed. The phase can be similarly changed.

By converting disk functions to periodic core functions and using phase control, the expression evaluator can effectively combine two disk functions at different abscissa values. This is the only way of achieving such time shifts in GROOVE.

Display

Once all of the algebraic expressions and the periodic function definitions have been given to the GROOVE program, we normally will want to see the effects of our formulas, knob turnings, etc., on the functions of time we are either creating or editing. The real-time display enables us to see an oscilloscope photo of any subset of the disk functions ($T1$–$T40$). Figure 6 is a photo of such a display. The functions are displayed in "pages" corresponding to one disk buffer full of information about each displayed function.

In addition to the functions, the position of each of the seven knob inputs is depicted by displaying seven points on a vertical scale. This vertical scale is made to march across the function display, thereby indicating the exact position of "program time" along the abscissas of each of the disk functions. The disk buffer number acts exactly like a "page number" in the conductor's "score" of displayed functions, enabling him to note the places in which mistakes were made, and return to them easily at a later time.

Control of "Program Time" and Editing

One of the most important features of GROOVE is the flexible control of "program time" which may be used both to edit and to alter the generation of the output functions. Coarse control of "program time" may be accomplished by typing TIME N, where N is a disk buffer number. If $N = 0$, the computer will simply go back to the beginning of the disk functions. At any point, we may set a switch which will cause the computer to recycle continuously through the same disk buffer. We may also slow down the progress of program time by reducing the frequency of the

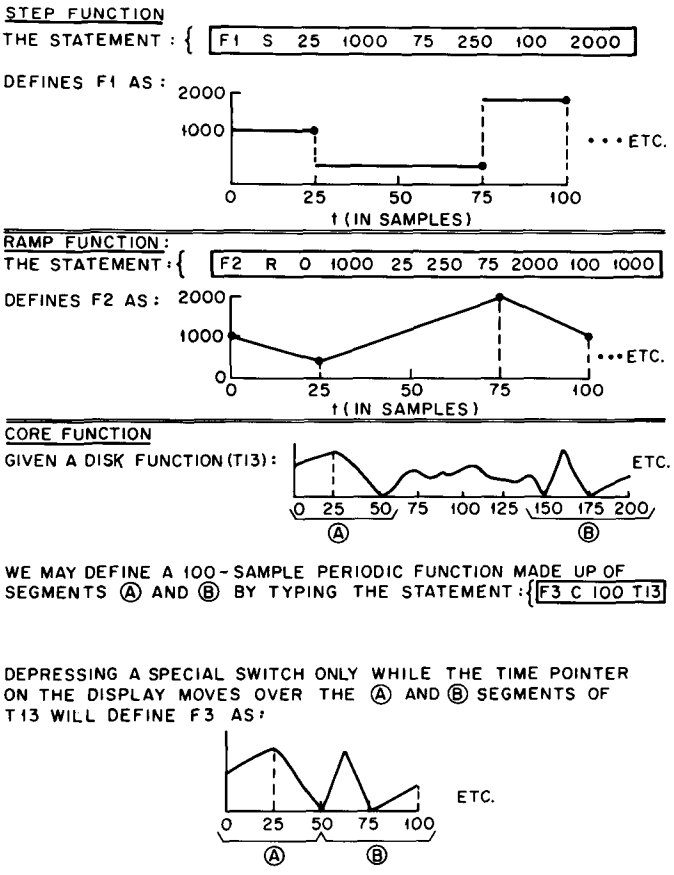

FIG. 5. Periodic function definitions

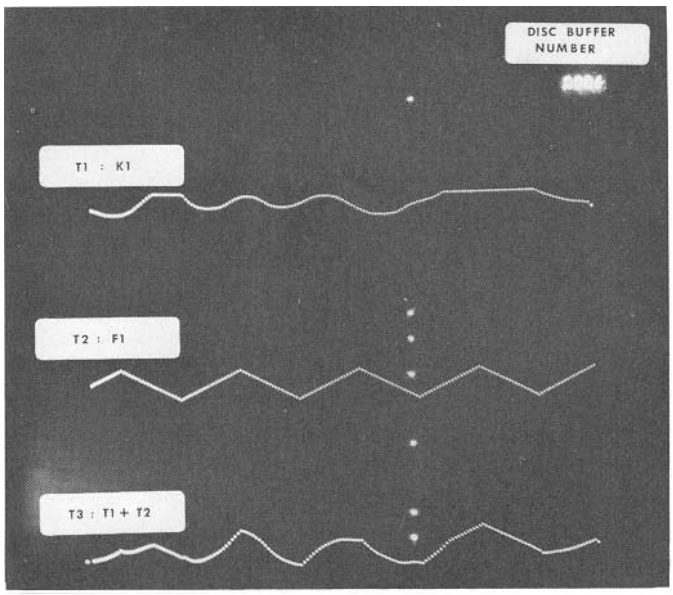

FIG. 6. Real-time display

interrupt oscillator. Or we may stop the progress of time altogether by throwing a switch which essentially tells the computer: "Don't progress time normally at all, but, instead, use the value of a knob to give the current position of time within one disk buffer." The x axis of the 3-dimensional wand is drafted for this task, since moving it from left to right most resembles the perceptual task of moving a time pointer along an abscissa. Along with the visual display of the time functions and the perceptual feedback from our controlled system, we now have a fine control over the exact position of program time. This is a very powerful feature of the editing system.

By throwing a switch, the user may essentially "redraw" any portion of any disk function using any input device he likes, such as the (X, Y) axes of the 3-dimensional wand or a knob value. While he is doing this, not only can he see what he is doing on the oscilloscope display, but he can also observe its effect on the controlled process. So it is quite possible to stop in the middle of a run and "tune up a chord" or adjust a motor speed at some point, and then go right on. The change may be either permanently registered, or as we mentioned before, it may be a "trial edit." Every precaution has been taken to insure that the GROOVE system will not destroy the record of a time function until directions to do so are made very explicit. But at the same time, given the appropriate commands, the system will allow any function of time to be altered in any conceivable manner.

Example

At this point, an annotated protocol from the computer typewriter will best illustrate the overall use of GROOVE. The following example generates the time functions needed to control one voice of a sound synthesizer to produce a simple melody.[2]

Dialogue	*Remarks*
$ *INIT*	1
TYPE NUMBER OF DISK FUNCTIONS	2
14	
TYPE STARTING TRACK ON DISK	3
100	
IS LOAD DESIRED?	4
NO	
TABLES NOW CONTAIN DEFAULT VALUES	5
DEFINE NEW PERIODIC FUNCTIONS?	6
YES	
READY	
F1 R 0 2000 300 0 599 0 600 2000 900 0	7
F2 R 0 0 3149 0 3150 2000 3300 0 3600 0	8
SPEED F1 10	9
SPEED F2 10	
DEFINE NEW T-VALUES?	10
YES	
READY	
T1: F1 + F2 + K4	11
*T2: (K1/128)*128*	12
TURN DISPLAY ON?	13
YES	
TYPE DISK FUNCTION NUMBERS	14
1 2	
TYPE NSDF VALUE	15
1	
INITIALIZATION COMPLETE	16
$	

Remarks

(1) The GROOVE program types a dollar sign, "requesting" a command thereby. We type INIT to cause the following initialization sequence to occur.

(2) This line requests the number of disk functions of time which we wish to reserve space for on the disk storage unit. We create more than we need here in order to save room for additional functions to be added later.

(3) One disk pack may contain several "compositions" beginning at different places (tracks). We arbitrarily start at track number 100.

(4) If "yes" is replied to this question, the program will copy a file from magnetic digital tape onto the disk, starting at track 100 as specified above.

(5) This line informs us that the program found no periodic function definitions or T-value definitions at track 100 on the disk. Therefore, it automatically fills all of the program tables which hold these definitions with "default values", $T1{:}T1$, $T2{:}T2$, etc.

(6) If "yes" is replied to this question, the program readies itself to accept periodic function definitions, as shown. If "no" had been typed, no periodic functions

[2] Both the operator and the computer use the typewriter. In this protocol, the operator's typing was subsequently italicized.

would have been defined and the next question would have been asked.

(7) This statement defines a periodic ramp function named $F1$ which has a period of 900 samples and the shape of two saw teeth. It will be used to generate rhythm patterns consisting of a 300 sample note, a 300 sample rest and a 300 sample note.

(8) This statement defines another periodic ramp function which will produce a 3150 sample rest followed by a 150 sample note.

(9) This statement essentially divides the time scale of $F1$ by 10. That is, it says that rather than using every point of $F1$, we will use only every 10th value of our definition as the value for $F1$. The reasons for this will be made clear below.

(10) Typing a blank line terminates the control of the periodic function definition processor. The INIT command processor now asks whether we wish to mathematically combine time functions (T-values). We do, so the reply is "yes."

(11) Output line number one is connected to control the amplitude of our sound source; therefore, output time function one ($T1$ in this case) is defined as the sum of our two rhythmic periodic functions, plus the "current value" of knob four ($K4$) from our control box. The effect of summing these two functions will be the effect of combining the two rhythmic patterns defined above plus an extemporaneous input from knob four.

(12) Output line number two is connected to control the frequency of our sound source. We therefore use $T2$ to control our melodic pitch sequence, while the periodic functions produce a rhythm automatically through $T1$. The relation given here quantizes (because of the effects of integer arithmetic on a computer) the current value of knob one into 32 equal steps. Thus the current value of knob one will determine which note of a 32-note scale will be sounded (since $0 \leq K1 < 4096$, $K1/128 < 32$).

(13) If "no" is replied here, the initialization sequence would be complete.

(14) We may select any of the 14 time functions for display here by typing their numbers. We choose to observe both our pitch and amplitude functions.

(15) This value allows us to specify the "resolution" of the CRT display; if "1" is typed, *every* sample of the specified function will be displayed, "2" means display every *other* point of each function, etc. Since the display preparation is time-consuming, typing a larger number than one here speeds up operations.

(16) At this point, the initialization sequence is complete, at the program "requests" another command by typing a dollar sign. This command is "START," which causes real-time processing to begin.

Let us now suppose that the above initialization sequence has been used, "START" was typed, and a "melody" in our 32-note scale was improvised by the user in real-time. When the improvisation is over, we push the box sense switch one button, the program stops real-time processing, and we type:

```
$ FUNC                                    17
    READY
  SPEED F1 T3
  SPEED F2 T3
$ TVAL                                    18
    READY
  T2: T2
  T3: K3/64
$ TIME 0                                  19
$ START
```

(17) This command allows us to define new periodic functions or redefine old ones. Here we specify that the "speed" of $F1$ and $F2$ is now given by $T3$, rather than being constant as it was before.

(18) We also input new T-value definitions by typing the TVAL command. $T2$ is set to simple "playback mode" and $T3$, which will control the speeds of $F1$ and $F2$, is a value between 0 and $4096/64 = 64$.

(19) This command resets the current disk track number to zero, which is synonymous with starting again from the beginning. We then start the program again, and, using the improvised melody as it was input before, superimpose new rhythmic patterns on it by turning knob three. Knob four still controls the overall amplitude of the sound.

Conclusions

The GROOVE program has been in a process of development and use since October 1968. Many of the features which we have described, such as knob controlled "program time" and core functions, were added as a result of the demands of users. Future changes will undoubtedly be made.

So far the program has only been used for sound synthesis. For this purpose, it is almost irresistible. Examples of existing music from Bach to Bartok have been realized. They can be performed with great precision and nuances. Wild distortions of existing compositions using algorithms have been done and seem compositionally interesting. The keyboard attracts improvisation which can be stored on the disk and subsequently edited. Typewriter-defined functions have been used to generate rhythm patterns.

Features that seem particularly effective are the arithmetic expression definition, the typewriter-defined functions, direct feedback from the sound generators to the person twisting knobs, and the editing flexibility inherent in knob-controlled program time. In general we are well satisfied with the program.

We believe the general concept of composing, filing, and editing functions of time is a significant addition to computer software for real-time computers. The embodiment of this concept in GROOVE together with its other features seems to be broadly useful. We are anxious to try other applications.

RECEIVED MARCH, 1970