The synthesis of music and speech

By P. M. Woodward*

This paper describes how a fast general-purpose computer with an amplifier and loudspeaker connected to one of its jump instructions can be programmed to play more than one musical note at a time. The same method of tone-generation can be used to produce artificial speech without special apparatus.

1. Introduction

Programs which purposely generate noises in the monitor loudspeaker of a computer are generally regarded as an amusement, but they may be the first playful steps towards serious aural communication between machine and man. Little appears to have been published about the actual techniques used. This may be because the work is unfashionably tied to the use of machine code, because its serious import may be doubted or because there is some difficulty in publishing the experimental results. Be that as it may, the subject is not devoid of mathematical interest.

The work reported here was concerned with what we might call the pianola problem, that is to say, the use of the computer to reproduce a sequence of predetermined sounds from a previously handwritten "score" without the use of specialized peripheral equipment or analogue devices. A good tune was, to the best of the author's knowledge, first played on the Ferranti Mark I computer at Manchester, with Mr. C. Strachey at the controls. Nowadays, with faster computers, it is possible to play several good tunes at the same time. S. N. Higgins produced a common chord from a computer (TREAC) by time-multiplexing as long ago as 1954. The compilation of the necessary instructions made a more startling sound than the climactic chord itself.

The computer, as an organ of musical tone-generation, differs from all other instruments in that the fundamental variable is time, not frequency. This is more than a mere change of viewpoint; it is shown in Section 3 of this paper that the physical limitations relating to pitch are actually inverted. In Section 4 is described a program which produces sound waveforms by means of a loop which is traversed at a fixed frequency above the audible range and unrelated to the audio frequency desired. By time-multiplexing, this may be applied to the production of several sounds at the same time, as described in Section 5. The method is suggested as being appropriate to the problem of synthesizing speech-sounds without the use of any special tone or noise generating equipment other than the computer itself. A brief discussion of some easy experiments is given in Section 6. A few such simple trials are enough to convince any experimenter of the extreme difficulty and subtlety of this subject. The paper concludes with the suggestion that an artificial noise-language might be a more practical possibility for man-machine communication.

**R.R.E., St. Andrew's Road, Malvern, Worcs.*

2. Production of a single tone

Any program which goes into a spin produces a musical tone if the instructions in the loop include something which affects the circuits feeding the loudspeaker. It is common practice for computer designers to have a disturbance of some kind sent to the speaker whenever a certain type of instruction is obeyed, the instruction being selected with a view to ensuring some form of audible output under all circumstances. An over-monotonous sound reveals a spin, and some degree of program-recognition is alleged to be possible. On the computer (RREAC) used by the author, a pulse of a few micro-seconds duration is sent to the speaker circuit whenever one of a certain class of jump instructions is obeyed. Most loops contain at least one such instruction and the resulting periodicity generates a musical tone. The tone is not, of course, physically pure. It may be analysed into a fundamental frequency and a very large series of harmonics which arise from the non-sinusoidal form of the disturbance, but the ear accepts this combination as a single musical note and associates it with the lowest frequency present, the repetition frequency of the program loop.

It is a very natural step to vary the length of the loop by padding it out with silent instructions, so adjusting the pitch of the sound. There is a variety of ways of doing this. **Fig. 1** shows a column of instructions, broken lines representing quiet instructions, and the solid line a pulse-generating jump. By suitable address-modification, this jump can be made to embrace a variable number of quiet orders, and so produce a variable pitch. Using this method, the loop may have to contain a thousand or more dummy instructions merely to pass the time while the lengthy period of a bass note is being produced. This may be regarded as a waste of storage space, though it must be admitted that the criterion of waste in this type of work must

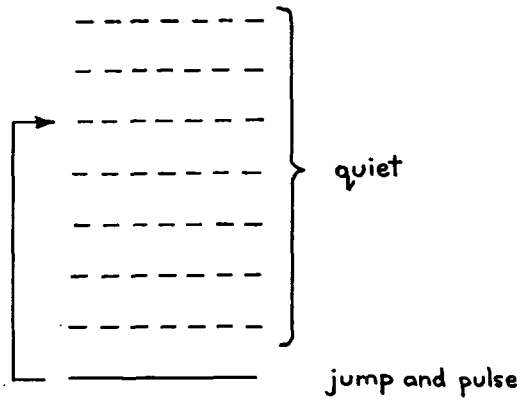

Fig. 1.—Loop producing a single note

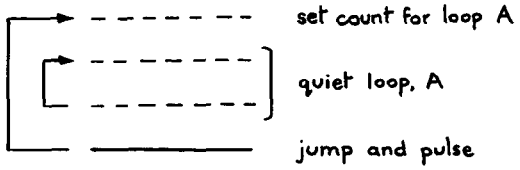

Fig. 2.—Alternative method of producing a note of variable pitch

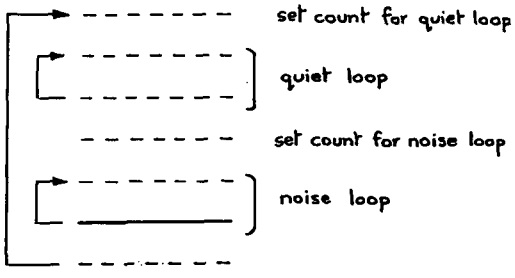

Fig. 3.—Tone generator with variable mark and space

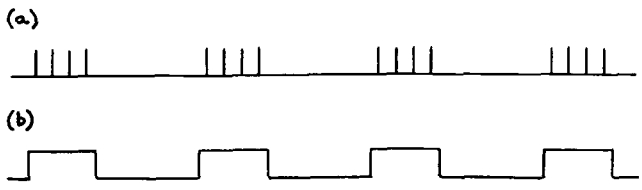

Fig. 4.—Waveform (*a*) before and (*b*) after smoothing

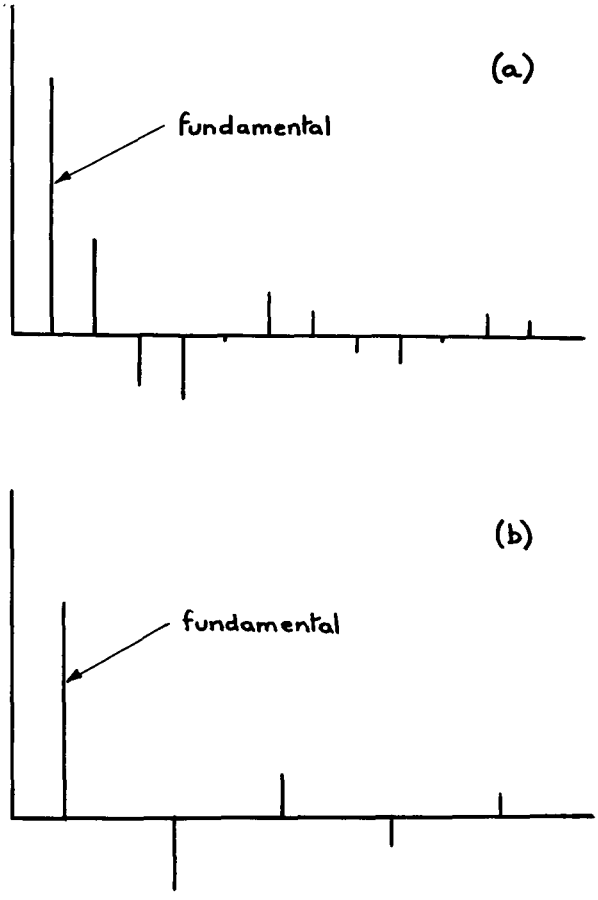

Fig. 5.—Line spectra of square waves
(*a*) as shown in Fig. 4(*b*)
(*b*) when up and down sections are equal

suffer some readjustment. **Fig. 2** shows an alternative plan which makes use of the fact that, on RREAC, a loop can also be closed without the inclusion of a noisy instruction. One possible trouble with both these arrangements is that the single very short pulse may be lost amongst the hundreds of quiet instructions which are obeyed between each recurrence. In engineering language, the mark-to-space ratio is too small. **Fig. 3** is a better arrangement if the speed of the computer permits. This method produces a waveform of the type illustrated in **Fig. 4**, which, when smoothed by the amplifier circuits, yields a square-shaped wave. For maximum intensity, the up and down sections should be equal, though large departures from equality can be tolerated.

A square wave is strong in harmonics, which fall off in amplitude broadly as the reciprocal of the frequency as indicated in **Fig. 5.** When the up and down sections are exactly equal, the even-order harmonics disappear, and a pleasing tone is obtained, resembling the clarinet stop on an organ.

3. The musical scale

The notes in one octave of the musical scale are a set of frequencies ranging from a keynote frequency of F (say) up to $2F$, both of these frequencies going by the same name in music. (For example, 440 cycles per second is the note A, and so is 880.) The notes between

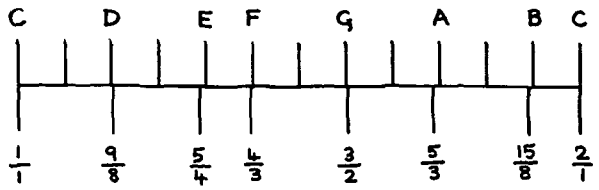

Fig. 6.—Frequency ratios for the true diatonic scale of C compared with the 12 equally tempered semitones

F and $2F$ are a kind of interpolation between the ratios 1 and 2, chosen so as to hit as closely as possible certain simple and aurally satisfying ratios such as 3/2 (the musical interval of a perfect fifth), 4/3 (the perfect fourth) and 5/4 (the major third). By a very fortunate accident of numbers, a subdivision into 12 parts with equal ratios suits well (see **Fig. 6**), and is what makes keyboard instruments usable with any note as the origin. The ratio 2 to the power 4/12 is roughly 5/4, whilst 2 to the 5/12 is near 4/3 and hence 2 to the 7/12 is near 3/2. The twelve equal intervals are, of course, semitones.

The object of this brief summary of musical theory is merely to show that the musical physicist thinks in terms of frequency ratios. It may be added that wind instruments can resonate at frequencies of F, $2F$, $3F$ etc., according to how hard they are blown, and as the musician's sense of pitch goes as log F, these notes seem to be progressively closer and closer together. Consequently, a closely spaced "table" of notes becomes available when the instrument is blown hard enough, and for a given fixed tube, the higher it is played, the easier it is to satisfy the requirements of the musical scale. Below the fundamental frequency F, the instrument will not work at all.

With a computer, the whole picture is turned upside down. For a given machine, there is a fundamental *time* T below which it will not function. And, like the trumpet in frequency, the computer can easily be made to generate periodicities of T, $2T$, $3T$ and so on. Its harmonic series is the reciprocal of that for any other instrument, so the *lower* the pitch of the music, the easier it is to play. It is quite surprising how fast a computer has to be to produce a reasonably accurate musical scale at a reasonable pitch. If, for example, we want a note an octave above middle C (i.e. soprano voice), we have to produce a frequency of about 500 cycles per second, and if we settle on an accuracy in pitch of 1% (a semitone is 6%), then we find

$$\Delta T = 0{\cdot}01 \times 0{\cdot}002 \text{ seconds}$$
$$= 20 \text{ microseconds.}$$

A computer whose fastest instruction time is any longer than this cannot satisfy the requirement if any of the simple schemes described in the previous section is used. There are several ways of overcoming this difficulty if need be. The same problem occurs in music; brass wind instruments would have to be too long unless some means were provided for augmenting the harmonic series. This is done by stubs of extra tubing which alter the fundamental frequency, so that the harmonics of the modified fundamental fill the gaps in the series F, $2F$, $3F$ etc. In exactly the same way, we could arrange a choice of program loops using instructions of slightly differing speeds. Selecting one or another loop would correspond to pressing a valve-key on the trumpet.

Another possibility is to look for some accident of numbers which might give a truer scale than the spacings of the harmonic series would lead us to expect. For example, if one were to be content with a simple diatonic scale (the "white" notes), one could take the rational frequency ratios shown in Fig. 6, which constitute the true musical scale to which the equally tempered scale is merely an approximation. For a frequency-based instrument, one could then take advantage of the small LCM of these numbers. Thus, taking $24F$ as the keynote, one would have the exact scale

$$24F,\ 27F,\ 30F,\ 32F,\ 36F,\ 40F,\ 45F,\ 48F.$$

Unfortunately with time as the base, we are concerned with the reciprocals of the frequencies, and the exact periods with smallest integral coefficients are

$$180T,\ 160T,\ 144T,\ 135T,\ 120T,\ 108T,\ 96T,\ 90T.$$

Since we should probably have been content with 1% accuracy of period anyway, there is little to be gained from this idea.

None of the above simple methods of obtaining a musical scale is capable of generalization to the production of several parts at a time. For work such as speech synthesis, it is essential to be able to produce sounds with a complicated spectrum, so the ability to generate musical chords was considered a step in the right direction. We show, in Section 4, that a yet further method of producing a given pitch then results, relying on the use of a loop which is not directly synchronized to the desired musical frequency.

4. The sampling method

Even before one attempts the simultaneous production of several notes, certain drawbacks to the methods described in Section 2 become evident. First, there is the limitation to a discrete set of pitches which, even for quite a fast computer, can be embarrassing. Secondly, the value of the loop count setting is not in constant ratio to the time duration of the sound, that is to say a note lasting for one second needs more audio cycles for a high note than for a low one. This means we have to perform a division every time we set a note-duration in the program. And thirdly, there is no convenient control over the loudness of the sound. All of these drawbacks are eliminated by the use of a loop which is not synchronized to the audio frequency.

Consider the following program

S: Add n to a finite register
Skip next instruction if result exceeds k

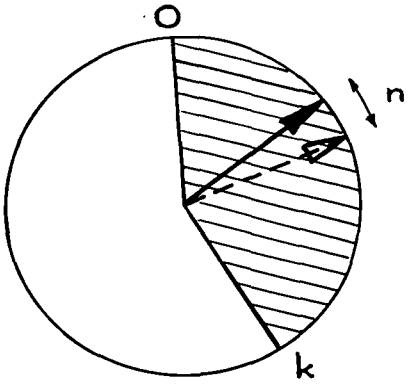

Fig. 7.—The sampling method. Speaker is pulsed when pointer moves within shaded sector

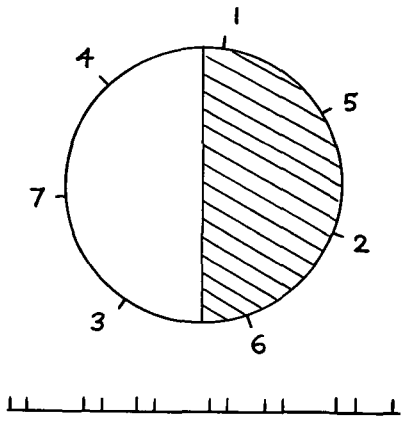

Fig. 8.—Sampling sequence for "period" of 3·5T, and pattern of pulses generated

Jump to next instruction but one and pulse speaker
Do nothing for time taken by a jump
Go to *S* (*without pulsing the speaker*)

The action is most readily explained by reference to **Fig. 7.** The number in the register is represented by the angular position of the pointer, which revolves at the audio frequency. In other words, one period of the audio wave is mapped on to the whole range of the register, and repetition occurs as a result of overflow of the number in the register back to zero (say). Each traverse of the program loop advances the pointer through an angle proportional to n and pulses the speaker if the pointer lies in a certain sector of the circle defined by the overflow point 0 and the constant k. The time taken to traverse the program loop is constant (whether the speaker is pulsed or not) and must be less than the audio period to ensure that the waveform is adequately sampled. With this arrangement, pitch, volume and duration can each be simply and independently controlled. The pitch is governed by the choice of n, which is directly proportional to the audio frequency, the volume is governed by k, which regulates the mark-to-space ratio, and the duration is now proportional to the count round the program sampling loop.

On the RREAC computer, which was used for the experiment, the four obeyed instructions take rather less than 50 microseconds. The frequency of the program loop is therefore more than 20 Kc/second, which is high enough to prevent the sampling frequency from being audible.

The program has other advantages. The control over pitch is extremely fine, and is not limited to periods which are multiples of some basic instruction time. Surprisingly, perhaps, the fineness of pitch adjustment depends only on the length of the register. Each extra bit of the register doubles the fineness. There is still an upper limit of pitch, since it is impossible to go faster than half a revolution round the circle in one loop time without appearing to go backwards. The period of the top note is thus $2T$ where T is the time for one traverse of the program loop. Periods $3T$, $4T$, $5T$ etc. are obtained by increments n which are exact submultiples of the range of the register, i.e. which divide the circle of Fig. 7 into a whole number of equal sectors. Whilst these are the only *true* periods obtainable, "false" periods intermediate between integral multiples of T are also usable. **Fig. 8** illustrates a false period of $3\frac{1}{2}T$, showing that pulse groups go to the speaker every $3\frac{1}{2}T$ on the average. There is no question that a musical note of the desired pitch is obtained when a false period is used; it is easily shown that there is a definite line in the spectrum at the intended frequency. The drawback lies elsewhere. The lack of synchronism between the sampling frequency and the audio frequency introduces extra lines into the spectrum of the sound. These occur at frequencies given by the differences between the sampling frequency $1/T$ and the frequency harmonics of the intended audio frequency. The difference tones, which are musically quite unrelated to the desired note, become increasingly objectionable the higher we go up the scale, and become easily audible at periods of less than about $10T$. If they can be tolerated here, the sampling system can provide a means of obtaining fine pitch control up to much higher pitches than would otherwise be possible.

Summarizing, the sampling scheme provides simple independent controls for pitch, volume and duration of sound. It gives effectively a continuously variable pitch, but the high notes are marred by the presence of unwanted difference tones which may themselves be lower than the wanted note and are almost always dissonant.

5. Polyphony

The next step is polyphony, or many sounds at once. In music, waveforms are merely added together and the ear separates them by frequency analysis. With a serial computer, we must resort to a system of time-multiplexing, and the sampling method comes into its own. The following program plays three notes at once.

*S*1: *Add n*1 *to register* 1
*Skip next instruction if result exceeds k*1
Jump to next instruction but one and pulse speaker
Do nothing for a jump-time
*S*2: *Add n*2 *to register* 2
*Skip next instruction if result exceeds k*2
Jump to next instruction but one and pulse speaker
Do nothing for a jump-time
*S*3: *Add n*3 *to register* 3
*Skip next instruction if result exceeds k*3
Jump to next instruction but one and pulse speaker
Do nothing for a jump-time
Go to *S*1

The principle should be clear. There are now three separate circles of the type shown in Fig. 7, processed cyclically. The number of notes is not limited to three; it depends on the speed of the computer. For best results, the time taken to go round the whole program loop once should be less than 70 microseconds to ensure that the sampling frequency for each part is beyond the upper frequency limit of the ear. Each of the three parts could be played as high as two octaves above middle C before difference tones become noticeable.

On RREAC, this was impossible and ways had to be found to improve the performance of a trio, one fair and one foul. The first takes advantage of the fact that only one of the three parts is generally required to play high notes. By writing a loop for four parts instead of three, the top part can be "doubled" and hence sampled twice as frequently to avoid difference tones. The sampling frequencies were approximately 6 Kc/sec for the two lower parts and 12 Kc/sec for the upper. By recording the music an octave too low and at half speed, a good playback can be achieved for public performance. A number of demonstration pieces has been scored for RREAC by J. B. Arthur, one of which has been used as the signature-tune for a recent series of television programmes.

As the preparation of the tape for a piece of music can be most time-consuming and the final performance is of little serious value, the effort is unrewarding unless a suitably convenient code is devised. The code should allow the use of variable-length notes referred to a diatonic scale modifiable by sharps, flats and naturals. Rests can be input as notes of zero frequency. The score is best stored in its entirety with all variables ready converted into the form needed by the program at run-time. During play, a pitch transition in any one part necessitates resetting a pitch constant in the program, and as the various parts have unrelated audio phases, no time for such resetting can be found without interrupting the other parts. If all the other parts move at the same time, the break is insufficiently long to destroy a *legato* effect, but if one part is tied, its continuity is spoiled unless the resetting computation is carefully timed and short enough not to interrupt it. The author has found it necessary to draw the line before incorporating this last refinement, which is only the first of what could easily become an evermore demanding pastime.

6. Speech synthesis

It is well known that the timbre of a musical instrument depends on the relative strengths of the harmonics of the fundamental note. A voiced speech sound, such as a vowel, is likewise distinguished by harmonic content, but in a very different way. The cavity of the mouth enhances those harmonics of the voice which happen to lie within certain resonant frequency bands. The pitches of these bands—the so-called formant frequencies —are independent of the voice frequency and are almost invariant between adults with a similar speech accent. They depend, of course, on the position of the jaw and tongue, which are the two main controls for vowel production. It is important that, whilst the voice produces a well-defined musical note, the formants have an appreciable bandwidth. For any vowel, three formant bands can easily be discerned from spectrographic analysis, though the lower two bands seem to carry most of the information which distinguishes the vowels from each other. If we assume that the voice alone is the principal source of acoustical power driving an otherwise passive and linear system, the waveform must be periodic at the voice frequency. Whilst a system of this type can readily be modelled in frequency, it is not so easy in time, particularly with the limitation that amplitude variations be produced as a result of smoothing a waveform which starts merely as an irregular train of pulses.

Values for the formant frequencies F_1, F_2, and F_3 are given by Holmes, Mattingley and Shearme (1964), but anybody with a musical ear can plot a diagram such as **Fig. 9**—which shows the values of the writer's own vowels obtained by *whispering* the vowels and picking out the notes on the piano. The second formant is relatively easy to hear, and is the note produced by whistling. The formant F_3 is virtually impossible to

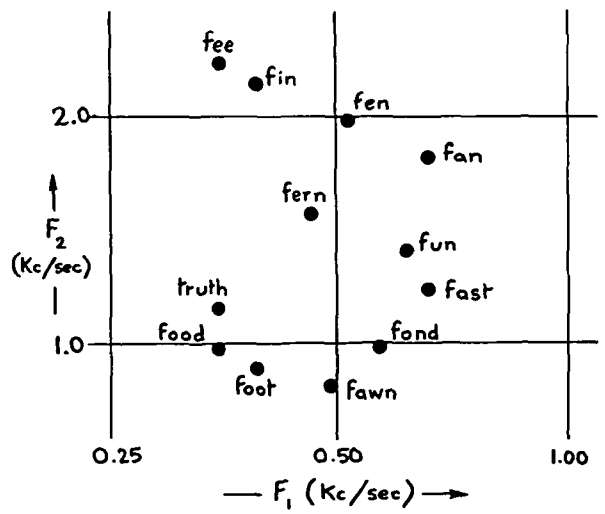

Fig. 9.—Formant frequencies for the author's whispered vowels

detect musically. In RREAC experiments, it was found that the presence of F_3 at $2\frac{1}{2}$ Kc/sec for all vowels seemed to lend a more "human" quality to the sound, reducing the nasal effect to which computers are especially prone.

The synthesis problem amounts to this. We have to generate a few cycles of each of the formant frequencies within one period of the voice pitch, and then repeat this waveform at the voice frequency. This is impossible if the formants are played in sequence because there is insufficient time in a voice period of a few milliseconds. After several experiments, conducted by the writer with J. M. Foster, the following program was found to give remarkably clear results.

S : *Set formant registers to zero*
*S*1: (*program as in section* 5 *above*)
*S*2: *ditto*
*S*3: *ditto*
*Repeat from S*1, *r times*
Interval of silence
Go to *S*

The outer loop gives repetition at the fundamental voice frequency. The interval of silence ensures a sufficient variation of overall pulse density within each voice cycle to ensure a strong-sounding fundamental, and variations of this interval will inflect the voice. The time can, of course, be used for computing and setting running values of the parameters so as to obtain diphthongs and voice inflexions.

It is a mistake to suppose that, once vowels and consonants have been simulated in isolation, their conjunction will result in intelligible speech. If the formants are varied in a stepwise fashion, the resulting sounds are quite unintelligible unless one knows what was intended beforehand. In natural speech, the formants vary slowly and smoothly, even persisting through seemingly colourless sounds like "sh", where they form a smooth interpolation between the preceding and succeeding values occurring in the speech. But with sufficiently fast machines, all these things could be allowed for by program.

Perhaps the intonation of the voice will be the most resistant problem; we have no notation for its inflexion and little understanding of the rules by which inflexion conveys meaning. This, it may be suggested, is why computer voices are often programmed to sing rather than to speak.

The generation of plosive consonants is probably less difficult than it seems at first, since the chief characteristic is the period of silence preceding the "burst". Fricatives are more challenging. Those which are unvoiced (*f*, *s* etc.) demand a wide-band noise source up to several kilocycles per second in place of the steady voice frequency. This can be provided from the computer by using random numbers to control the times at which pulses are sent to the speaker. A random number can be generated in as little as 30 microseconds by two RREAC machine instructions which obtain the next term in an overflowing Fibonacci sequence, thus

Replace f_{n-1} by $f_{n-1} + f_n$
Exchange f_n and f_{n-1}

I am indebted to J. M. Foster for contriving this program. We have not yet succeeded in modulating formants by this noise source except at a very throaty pitch. There is no difficulty in producing unmodulated hissing noises of varying pitches and bandwidths, but this is not sufficient in the context of continuous speech. As for voiced fricatives (*v*, *z* etc.), no work has yet been undertaken.

7. Discussion

The reader is likely to be asking himself whether large heavily loaded computers are ever likely to spare the time required for outputs of the type described in this paper. The answer is probably no, but this does not rule out the whole possibility of aural (and oral) communication. A similar problem occurs for visual displays, which also consume processor time extravagantly.

Music and speech sounds have been attempted because they already exist. Just as it has been suggested that a tactile code for communication with machines might be developed, is it too far-fetched to suggest that a noise-code simpler than natural speech could be devised? The telephone system already relies on public recognition of various different buzzes and pips. Might we not contrive a "language" for two-way communication with the computer?

Reference

HOLMES, J. N., MATTINGLEY, I. G., and SHEARME, J. N. (1964). "Speech Synthesis by Rule", *Language and Speech*, Vol. 7, p. 127.