Programming for the Glace-Holmes Synthesizer

Ignatius G. Mattingly
Haskins Laboratories and University of Connecticut

This afternoon I'm going to talk about methods for computer control of a synthesizer like the Glace-Holmes synthesizer. I'm going to make the assumption that most users of this synthesizer will be driving it from a laboratory computer, like the Honeywell DDP-224 or one of the DEC PDP series, and that the immediate objective is to write a control program which will enable an experimental phonetician, a linguist, or a psychologist to make the synthesizer do what he wants it to do: to enable him to enter a parametric description of an utterance into memory, edit this description, synthesize the corresponding utterance, display the description, and store it for later use. With such a program the experimenter can familiarize himself with the synthesizer's advantages and limitations, and master what Gunnar Fant has called "synthesis strategy": how best to produce the various speech sounds with this particular synthesizer. He can then consider more elaborate and specialized undertakings, such as synthesis by rule, or the preparation of psychological tests.

The most useful approach, I believe, is to organize the control program as a series of independent routines, each of which accomplishes some specific task. The experimenter calls one of the routines by typing a mnemonic command, e.g. T-A-L-K, "TALK"; he may then be asked to type in additional information, the routine is executed, and he is then notified that the system is

Paper read at the 77th Meeting of the Acoustical Society of America, Philadelphia, 8-11 April 1969.
ready for another command. It is best to begin by programming a few obviously essential routines and then add other, more specialized ones as the experimenter develops new requirements. While these major routines are independent modules of the program, the programmer will find that certain subroutines will suggest themselves, and that he will use these subroutines repeatedly in building up new major routines.

Whether you prepare the program in compiler language or assembly language will depend on the state of the software already available, how much core space you have and so on. Most of the programming required is input-output programming, to control not only the synthesizer itself but the typewriter, the printer, the display unit and whatever data storage units you have -- magnetic tape units or disc drives. If you already have standard subroutines to deal with all these devices, and ample core space, it is of course much easier to prepare the program with a compiler like FORTRAN. This is what Peter Denes at Bell has done. On the other hand, if you are starting more or less from scratch, as we had to at Haskins, it may prove easier to work in assembly language.

In preparing the control program, one must keep constantly in view the experimenter's probable limitations as a computer operator and his lack of interest in the inner details of the program. His mind is on his research, and the more simple and straightforward the program appears to him, the more willing and able he will be to use it. He will be most comfortable with a limited set of commands, each clearly corresponding to something that he wants the computer to do. As far as possible, the errors he will make should be anticipated. For each command, he should have written instructions telling him in detail what he must do and what he can expect of the computer. It should be possible for him to spend an hour or so looking over your instructions, and another few minutes learning how to turn equipment on and off, and then begin synthesizing speech. To achieve this kind of simplicity requires forethought, extra programming and careful documentation, but if the experimenter actually uses the program,
the investment is well worth it.

Let us begin with the speech synthesis routine itself, and return later to other forms of output and to input and editing. We assume that the configuration of the synthesizer has been decided upon, so that we know how many channels of parametric information we propose to control; and that some form of digital-to-analog interface is available so that the problem of control reduces to the specification of time-varying digital values for the various parameters. In what format do we store the parametric description of the utterance? Unless we have a great deal of core storage we will probably wish to pack parameter values several to a word, allotting a certain number of bits to each parameter. During synthesis, we will unpack the values for successive moments in time and transfer them to the digital registers of the D/A converters. Thus we must decide how finely we wish to quantize the range available for each parameter. In the end this is a practical question which depends on the word-length and the amount of core storage available in your particular computer, and on how coarse a quantization is consistent with your particular research objective. All I can do at this point is to suggest some plausible numbers. I am assuming the parameter ranges specified by Glace: if you choose to alter these ranges seriously, different quantizations may be appropriate.

For F0, F1, and F2, 5 bits are adequate and anything more than 6 bits is probably just wasted. For F3 and FN I suggest 4 bits, and for the various amplitude parameters, 4 bits. I have no experience yet with the consonant parameters, but I am thinking in terms of 5 or 6 bits. Two more bits are required for the switch control. If we take advantage of the provision for representing 15 parameters with 11 channels of information, the state of the synthesizer can be specified with 50 bits, more or less: say two 24-bit words, or 4 12-bit words, or 3 18 bit words.

The next consideration is timing. We need to have some kind of clock available to keep track of the interval between transmission of successive sets of parameter values to the registers of the D/A converters. This clock can be an external hardware clock,
or just a subroutine of the program; the hardware clock has the advantage that it can be used with an "interrupt" system.

Two approaches to timing are possible: regular or irregular transmission of parameter values. If we choose to transmit sets of values regularly, we have to decide upon the interval. 20 msec. is pretty coarse; 5 msec. seems more than fine enough for most purposes. It is quite easy to make this interval a variable which the experimenter himself selects by means of an appropriate routine. If an interval of 10 msec. is chosen and three words of memory are required for each set of parameter values, a parametric description of 5 seconds of speech will require 1500 words of core. By the way, it does not turn out to be very useful to be able to synthesize utterances much longer than 5 seconds. The core space which would be required will generally be needed for other purposes, and if you do want to synthesize a long piece of connected speech, it is more practical to do it sentence by sentence by sentence, anyway.

If we choose irregular transmission of sets of values, then the interval between transmissions depends on the set. The duration of the interval then becomes a sort of pseudo-parameter. This approach has the advantage that a new parameter set is required only when at least one parameter value is to be changed, permitting very substantial storage economies for some types of synthesis. But if both the excitation parameters and the formant amplitude and frequency parameters are changing independently -- as tends to be the case in imitations of natural speech -- the saving in storage is not so great, and the stored information is more difficult to program and to think about -- or so it seems to me.

Having decided about parameter quantization and about timing, we can now write the routine which synthesizes the stored utterance by transmission of successive parameter value sets to the D/A converters. An obvious scheme is to synthesize the utterance repeatedly, with a second or so's delay after each iteration, until the experimenter raises a sense-switch to terminate the process. But this repetition can become annoying, and I have found it helpful to provide an alternative arrangement.
under which synthesis is not repeated until the experimenter raises a switch. The last parameter set will be used to silence the synthesizer, and it is a good idea to set the frequency parameters for this set to the initial values in the utterance. This avoids the undesirable effects of a sudden shift of frequency at the beginning of the utterance.

So much for the basic output routine. Let us next consider the problem of getting the parametric description into core. There are two approaches here, one digital, the other analog.

If a digital input is required, it is convenient for the experimenter to use a decimal coding corresponding to the quantization assigned each parameter. Thus a five-bit coding of FØ would be represented in the input by the decimal integers 0-31. Such a coding takes a bit of getting used to, at first, but it is faster to work with than actual frequencies in Hz or amplitudes in db, and has the advantage that the experimenter knows what he is actually getting. This input will probably be prepared on cards or paper tape: in either case, it is quicker for the experimenter to prepare the input parameter-function by parameter-function, than to type, one after another, the parameter value sets for successive moments of the utterance. An appropriate routine reads the information in, translating the decimal numbers to octal and packing them into the appropriate parts of words in core. In practice, I have found that experimenters use a routine of this sort only to get started. They then use the editing routines to create new parametric descriptions from old ones.

Analog input is highly desirable, too. If you're trying to copy a natural utterance from a spectrogram, it is very convenient to be able to trace the formant path directly into the computer instead of manually converting it to digits first. This kind of operation is most conveniently handled with a display scope and a light pen or similar device. The function is traced on the face of the scope and simultaneously displayed, then stored.

Once we have read in a parametric description of an utterance by one means or another, and synthesized the corresponding
utterance, it is more than likely that we will wish to edit the
description. Editing proves to be a matter of changing one or
more successive values of a parameter function, and of stretching
or deleting portions of the parametric description. Certain
other possibilities which occur to one, for example, the substi-
tution or insertion of a parameter value set, have not in my
experience turned out to be useful. In stretching, the latter
part of the parametric description is moved to a higher memory
location and the gap left is filled with a specified number of
repetitions of the last parameter value set in the earlier portion
of the utterance. In deletion a specified series of parameter
value sets is dropped and the following part of the parametric
description is moved to a lower memory location. Alteration of
parameter functions can be done value by value, but it is also
very useful to have an interpolation routine, so that transitions
can be calculated automatically and long steady states can be
quickly altered.

Like the original read-in, these editing tasks can be per-
formed either by digital or by analog methods. As a basis for
digital editing, it is very helpful to have a routine which prints
out the parametric description in a convenient format. At Haskins
we have relied heavily on a printout in which the coded values
for the parameter functions are arranged in columns, each line
of the printout corresponding to a parameter value set. It is
easy to tell from such a printout what each parameter is doing and
what other parameters are doing at the same time, and to plan the
changes you wish to make. If you have a line printer, you may
as well print the entire parametric description; if only a type-
writer is available it is more convenient to print a few selected
lines.

For analog editing, a parameter function can be shown on the
display unit and then modified with a light pen. It is helpful
to display the function twice. One of the functions then serves
as a reference while the other is modified.

Whether analog or digital editing is used depends on the
equipment available and on the experimenter's research objectives.
An analog display is more conceptually direct and probably preferable for exploratory work. But if you wish to generate a systematically related family of stimuli or to work out rules, you are better off with digital methods. Better still, have both!

Two other routines are also useful in editing. One is a routine which synthesizes first the utterance corresponding to the original parametric description and then the utterance corresponding to the edited description. This permits one to evaluate supposed improvements. The second routine restores the edited parametric description to its original form. Both of these routines require, of course, that a duplicate of the original parametric description be stored elsewhere in memory, or in auxiliary storage, at the time of read-in.

Once the parametric description has been edited, we will probably want to file it for future use. Disc-files, magnetic tape units and paper tape can all be used to save parametric descriptions. In the case of the first two it is a good idea to include, on the file or the tape itself, an inventory of the contents and their locations. The experimenter can then be prevented from inadvertently writing over material he wants to retain. Otherwise, the routines required are straightforward input-output data transfer routines, and there seems no reason to discuss them in detail. The disc file, I would say, is the preferred mode of storage, because of its quick retrieval.

The routines discussed thus far are basic and generally useful for all kinds of synthesis work. More specialized routines and programs can be written to meet the experimenter's particular objectives: speech perception experiments, synthesis by rule, close imitation of natural speech, vocoder studies and so on. Before he starts any of these projects, however, the experimenter will find it worthwhile to do some simple exercises with the basic program. Producing some vowels and some consonant-vowel and vowel-consonant nonsense syllables which can be identified by someone else with reasonable reliability is not a trivial task, despite all we supposedly know about the cues for speech perception, and is a very good way to develop a "synthesis strategy." I can
also recommend the exercise of translating into the particular coding you have chosen the parametric description for some utterance already successfully synthesized on another synthesizer: John Holmes' "bird in the hand," for example. Not only is the exercise instructive, but a parametric description of this sort can be used as a quick, informal check on the performance of the synthesizer itself.

I have tried to review some basic aspects of computer control of formant synthesizers: input, output, storage of parametric descriptions and the use of such a description to drive the synthesizer itself. Let me conclude by emphasizing again the importance of a straightforwardly organized, well-documented control program, and of a period of exploration using this program, as a necessary foundation for further work in speech synthesis.