

SIMPLE DIGITAL SPEECH SYNTHESIS

William M. Fisher and A. Maynard Engbretson

Central Institute for the Deaf
818 South Euclid Street
St. Louis, Mo. 63110

and

Biomedical Computer Laboratory
Washington University School of Medicine
700 South Euclid Street
St. Louis, Mo. 63110

Copyright 1975

Association for Computational Linguistics

SUMMARY

Relatively simple computer methods for synthesizing speech to be used in phonetic/perceptual research are presented, with particular references to the problems and successes encountered in the development of such a system at Central Institute for the Deaf and the Biomedical Computer Laboratory of Washington University. The purpose of this paper is to present a synthesis and clarification of established methods so as to encourage other computational linguists to tackle digital speech synthesis. The approach is semi-tutorial: crucial algorithms are given in Fortran or block-diagram form, and bibliographic references that were found to be most useful in the system development are listed and discussed.

The system described requires a minimum of hardware; a mini-computer is sufficient, if it is equipped with tape or disk secondary memory. The sound pressure wave is calculated entirely by software and only a digital-to-analog converter and a low-pass filter are required to convert it to a recordable electrical signal.

The vocal apparatus is simulated by a rough model which is still general enough to make most speech sounds. The two main types of excitation of the vocal tract -- periodic glottal waves for voicing and random noise for frication or aspiration -- are supplied by algorithms presented as function subroutines. The effect of the vocal tract on these inputs is modelled by combinations of three other elemental functions, whose coding is based on recursive equation theory for computational efficiency. A resonance provides the user with a means for accentuating the signal at a certain frequency, such as a formant frequency of a vowel; an anti-resonance or notch filter is provided to cut back the energy at a certain frequency, as in simulation of the nasal anti-formant; and a radiation-effect subroutine simulates the effect on the speech signal of passage from the lips through a short stretch of air. Empirically obtained wave shapes and spectra of the outputs of these five basic functions are given in order to give the reader a better feel for what they do.

These five elements can be combined in a number of ways. A detailed discussion is given for one of the simplest reasonable models, in which the glottal wave and frication generators excite a series of three variable resonators, using a set of fixed resonances to simulate higher frequency formants in addition to the radiation-effect simulator. Several other more complex arrangements are presented, including parallel resonator models and models with separate filters for shaping voiced, fricative, and nasal components, and their advantages and disadvantages are discussed.

An example is given of a complete modular Fortran program generating the word "seat". The equations for specifying the parameters controlling the elemental functions were derived, with much effort, from analysis of one token utterance, and spectrographs of the real and synthetic words are shown to illustrate the degree of naturalness obtainable with the simple three-resonator series model. A simpler example for generating a constant vowel sound is also given, along with a summary of data useful in making many vowels.

This paper is a slightly expanded version of one given orally at the 12th annual A.C.L. meeting in Amherst Massachusetts.

Table of Contents

	Page No
I. Introduction	4
II. Overview	5
III. Basic Elements	10
A. Sources	10
1. Glottal Wave Generator	10
2. White Noise Generator	16
B. Spectral Shaping Elements	17
1. Resonances and Anti-resonances	17
2. Radiation Effect	22
IV. Organization of Elements	22
A. The Simplest Model	22
B. Control	28
C. Other Models	30
1. Parallel Formant	31
2. Separate Noise Shaping Channel	31
3. Other More Complex Models	33
D. A Complete Example	33
V. Final Remarks	44
Bibliography	46

I. Introduction

Several years ago, it was decided that the Research Department at Central Institute for the Deaf in St. Louis should have a digital speech synthesizer to aid studies in psychoacoustics and phonetic perception. The equipment on hand at the time consisted primarily of a 12-bit-word mini-computer with keyboard and scope, with special-purpose hardware for doing digital-to-analog conversion, low-pass filtering, and floating-point arithmetic. More peripheral devices and core memory have now been added. We have been working since then on writing digital computer programs to synthesize speech, studying the literature and gaining practical experience.

Almost all of the theory and techniques necessary to program the synthesis of English sounds can be found in published literature, but in bits and pieces, here and there. Utilizing the contributions of many authors, plus our own experience, we present and explain a basic program for synthesizing speech, in the hope that computational linguists who have not worked with low-level speech phenomena may be encouraged to program synthesizers.

The use of synthetic-speech stimuli has been extremely important to the investigation of the perceptually distinctive features of speech and of low-level phonological rules, but much work remains undone. Synthesizing speech is clearly important to phonetic research, and the field could well use more researchers with linguistic training. The system described in this paper requires relatively small investments in equipment and programming.

II. Overview

Synthesizers use varying amounts of special-purpose hardware. The type of synthesis we describe here uses the bare minimum, calculating the speech wave on a digital computer and requiring only a digital-to-analog converter and a low-pass analog filter as special hardware. This minimum set of equipment is shown in Figure 1, page 6. If we assume that tape or disk secondary storage is available, then a 4 k 12-bit-word mini-computer is sufficiently large, and a 12-bit D/A converter will give enough dynamic range.

Once the speech wave is generated and stored on tape or disk, there is a problem of writing a program to output enough of it synchronously at a fast enough rate. We will not go further into this problem here, since the solution will depend on the particular machine installation you have. Whatever output sample rate you achieve, there are two things to note: the analog low-pass filter should pass only frequencies below the output sample rate, and the output sample rate is a parameter whose value must be fed into the digital calculations.

Although our synthesizer can be described as a terminal analog model of the vocal apparatus, the attitude we take is that our method of synthesis is used rather to produce the significant acoustic features of speech. To set the stage for an understanding of the synthesizer presented here, and for the benefit of those not familiar with acoustic phonetics, let us review for a minute what we are synthesizing.

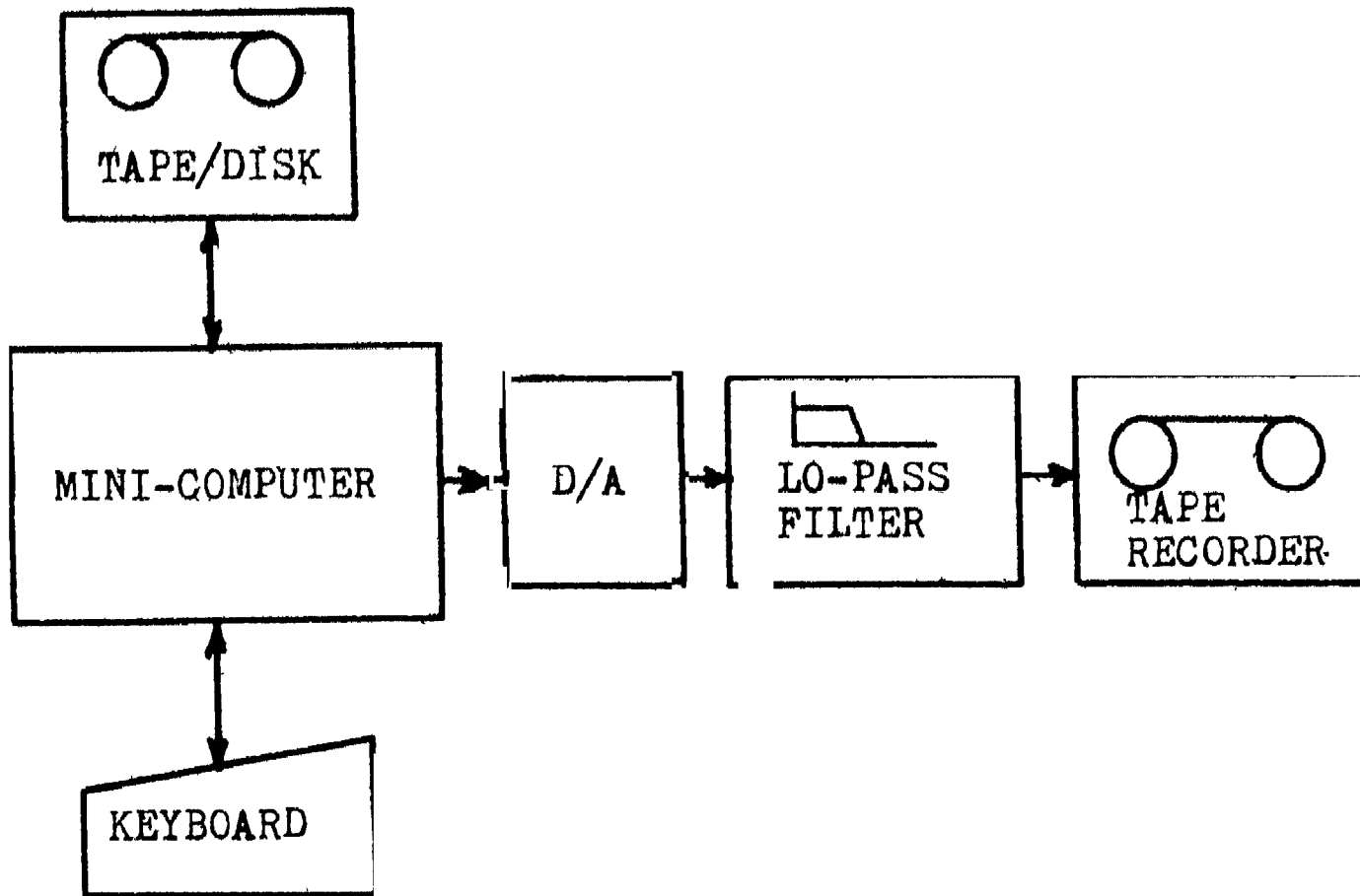


Figure 1. Minimum Hardware Needed

Figure 2, page 8, shows a typical spectrogram of the word "seat." Frequency is the vertical axis, time the horizontal, and darker marks indicate more energy. The mottled area at the upper left is the quasi-random noise of the /s/ sound. To make other fricatives, such as /f/, we need to alter the frequency spectrum and intensity of the noise. The dark horizontal bands in the center are concentrations of energy -- resonances called "formants" -- characteristic of vowel sounds. To make different vowels, we need to change the center frequencies, bandwidths, and relative intensities of the three lower formants visible here. There are higher-frequency formants, which do not show up well in this spectrogram, but they seem to be important only to the naturalness of the speech, not to which vowel is perceived. Note the beginning and ending slopes of the formants; these formant transitions are crucial to the perception of occlusive consonants such as the stops /b/, /d/, and /g/. And finally, the vertical mark at the far right is a burst of noise marking the release of the final consonant.

The detailed algorithms we present here will be expressed in Fortran for clarity, although the synthesizer with which we have had the most experience is coded in assembly language. We are presently converting to Fortran, and the subroutines and final example program listed in this paper have been tested in their Fortran form.

Before we get into details of synthesis, consider the overall logic, Figure 3, page 9. The main thing to note in

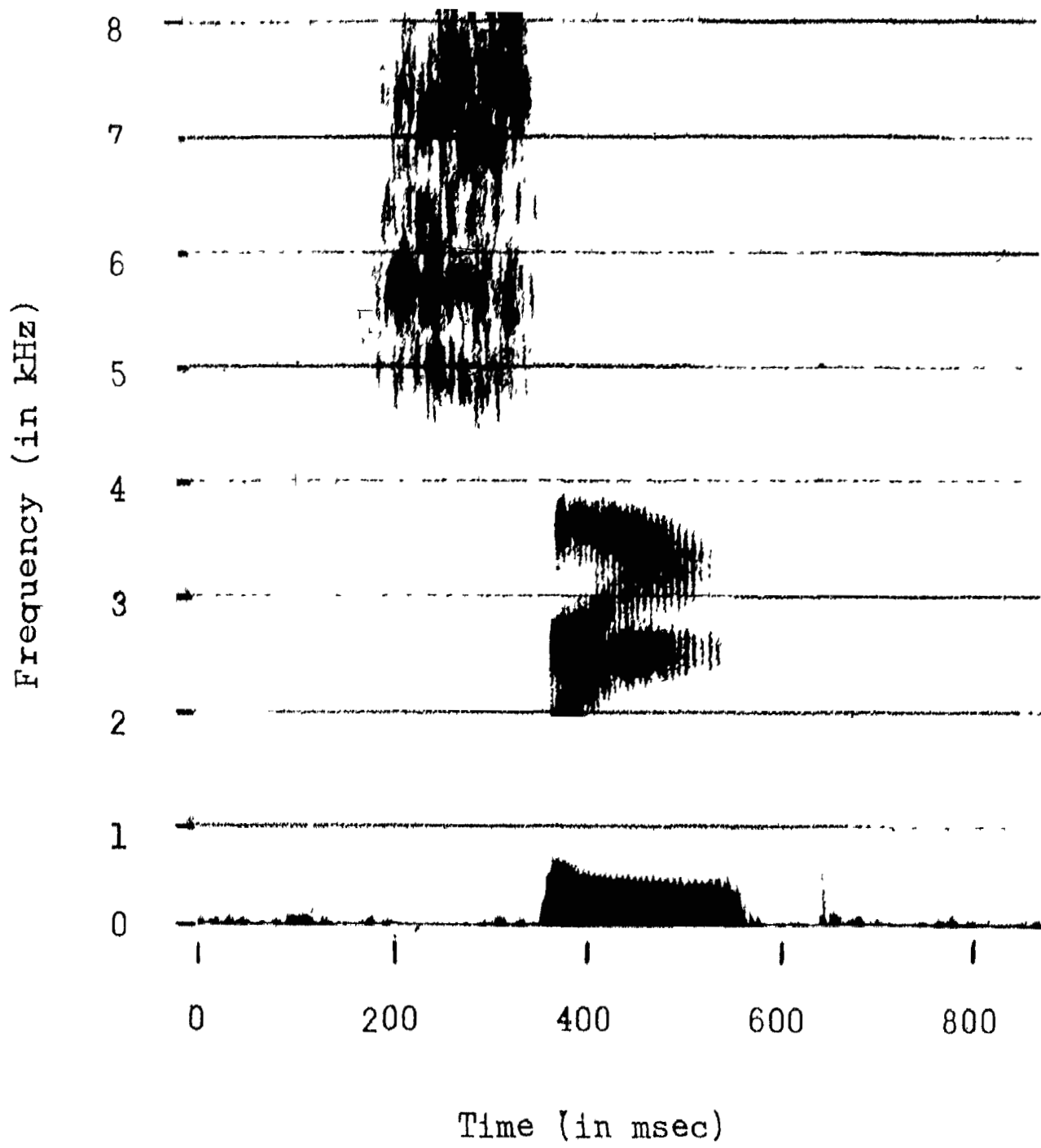


Figure 2. Typical Spectrogram of "Seat"

```

C  SKELETON OF SYNTHESIS PROGRAM
C
C      DIMENSION NBUF(256)
C
C  INITIALIZE
C
C      NBSIZE=256
C      NBLKS=10
C
C  GENERATE
C
C      DO 800 NB=1,NBLKS
C        DO 750 NPT=1,NBSIZE
C
C  GENERATE NEXT SPEECH WAVE POINT
C  AS THE VALUE OF YN
C
C
C
C  STORE SPEECH WAVE POINT
C
C      NBUF(NPT)=IFIX(YN)
750  CONTINUE
C
C  WRITE OUTPUT BUFFER
C
C      CALL WRITB(NBUF)
800  CONTINUE
C      CALL EXIT
C      END

```

Figure 3. Over-all Synthesizer Program Logic.

WRITB is a subroutine to write the contents of the array NBUF onto tape or disk.

this logic is that we synthesize the speech a point at a time, in one pass, storing each buffer-full on tape or disk as it is generated. This program will produce an integral number of buffer-loads, but other methods for terminating the main loop are easy to implement.

III. Basic Elements

There are five basic elements in this method of speech synthesis:

1. A glottal wave generator, with controls for repetition rate (pitch) and amplitude;
 2. A white-noise generator, with a control for amplitude;
 3. A resonant filter, with center frequency and bandwidth controls;
 4. An anti-resonant filter, with similar controls;
- and 5. A radiation-effect simulator.

These five elements can be connected in a variety of ways to produce models of greater or lesser complexity. The glottal wave generator and noise generator produce sounds whose spectra are then shaped by combinations of the other elements.

A. Sources

1. Glottal-Wave Generator

Natural glottal waves, while subject to much variation, are usually considered to consist of three parts: a glottis-opening phase in which the volume velocity is increasing, a glottis-closing phase in which the volume velocity is decreasing, and a glottis-closed phase in which the volume velocity is zero. The spectrum of such waves is supposed to fall off at about 10 to 12 dB/octave,¹

¹ Cf. Flanagan (1958)

and a theoretical spectral analysis of a linear approximation shows regular lobes superimposed on this spectrum.¹ We have just begun to study natural glottal waves in our Research Department, and while no definite results can be reported yet, Figure 4, page 12, shows the wave shape and spectral analysis of one typical male glottal wave period. The opening and closing phases will be more apparent if the wave is considered to begin and end at the minimum value instead of a zero value as our analysis program has done. There does not appear to be a closed phase of the wave, but this is one of the variations reported in the literature.

Two general methods have been used to produce approximations to glottal waves: in the first, an impulse is generated and fed into appropriate spectral shaping filters,² while in the second, a standard wave shape is used as a pattern for each glottal pulse.³ We chose the second method, because Rosenberg has shown that some simple wave shapes produce synthetic speech that sounds natural,⁴ and the method seemed conceptually cleaner.

Rosenberg studied the naturalness of speech synthesized from a number of different wave shapes used as glottal pulses. We have used two of his shapes in our work: the linear (called by some "triangular") and the polynomial approximations. Of the shapes he studied, the linear requires the least amount of

-
1. cf. Flanagan (1958) and Dunn et al. (1962)
 2. cf. Rabiner (1968) and Fant and Martony (1962)
 3. cf. Rosenberg (1971) and Sekimoto (1973)
 4. Rosenberg (1971) reports that some subjects preferred synthetic speech with certain glottal wave shapes to natural speech.

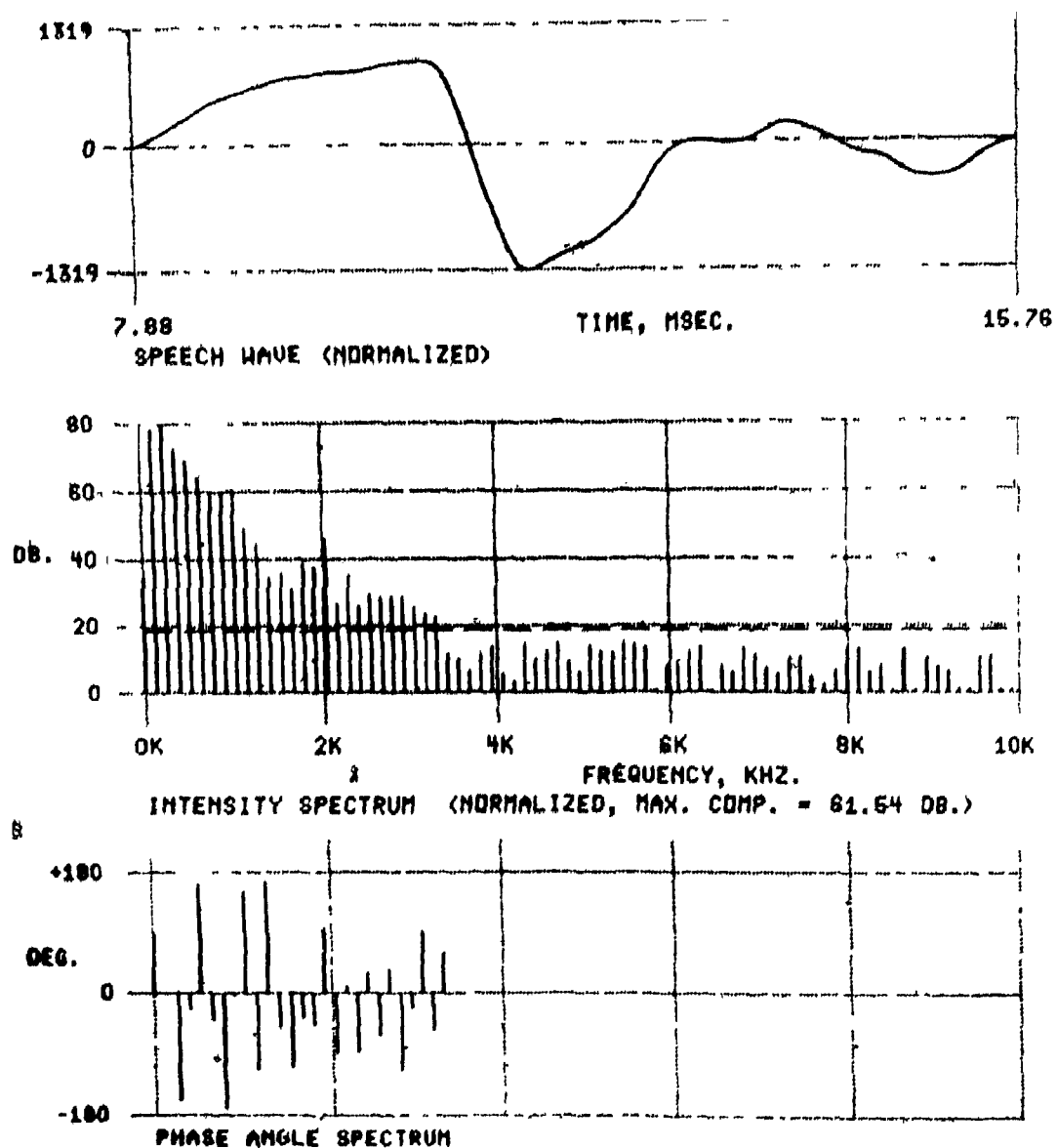


Figure 4. Wave Shape and Spectral Analysis of a Typical Glottal Wave Period. The intensity spectrum has been normalized so that the largest component is at the 80 dB level; before normalization it was 61.54 dB relative to an arbitrary standard of 0.25. The horizontal line across the intensity spectrum graph is a conservatively estimated noise cut-off line; any component with magnitude less than this level may be the product of computational noise. The phase angles of the components are ϕ_n , where the wave is represented by $\sum_n A_n \sin(n\omega_1 t + \phi_n)$ and ϕ_1 is arbitrarily zero. The phase angle display is suppressed for any component whose intensity falls below the noise cut-off line.

execution time and the polynomial scored the highest in his tests of naturalness. Figure 5, page 14, shows the wave shape and spectrum of a linear approximation and Figure 6, page 14, shows that of a polynomial approximation. The over-all falloff of the spectrum of the polynomial more nearly matches our example wave, but no lobes are apparent, as they are in the spectra of the linear approximation and the natural example. Judging from some informal listening tests we have made, these distinctions do not seem to make a great difference: both approximations sound good.

As Figure 7, page 15, we present a Fortran function GLOT(P,AV) for generating glottal waves using the polynomial approximation. Values of pitch (P) and zero-to-peak amplitude (AV) are passed directly as control parameters.

The subroutine uses the common area to store several variables, which of course could be declared as formal parameters instead. ISWV is a voicing switch used in the logic internal to GLOT. TDEL is the period between output sample points in milliseconds; in the over-all initialization of the program, its value should be calculated as $1000.0/OSR$, where OSR is the output sample rate in samples per second. TG is a variable used as a simulated time clock by GLOT, keeping track of how far through the glottal wave it has gone. TP, T1, and T2 are durations (in milliseconds) from the beginning of the glottal pulse, calculated and used by GLOT: TP is the duration of the pulse, T1 is the duration of the opening phase, and T2 is the duration of the opening and closing phases combined. OPTR (an acronym for "opening time

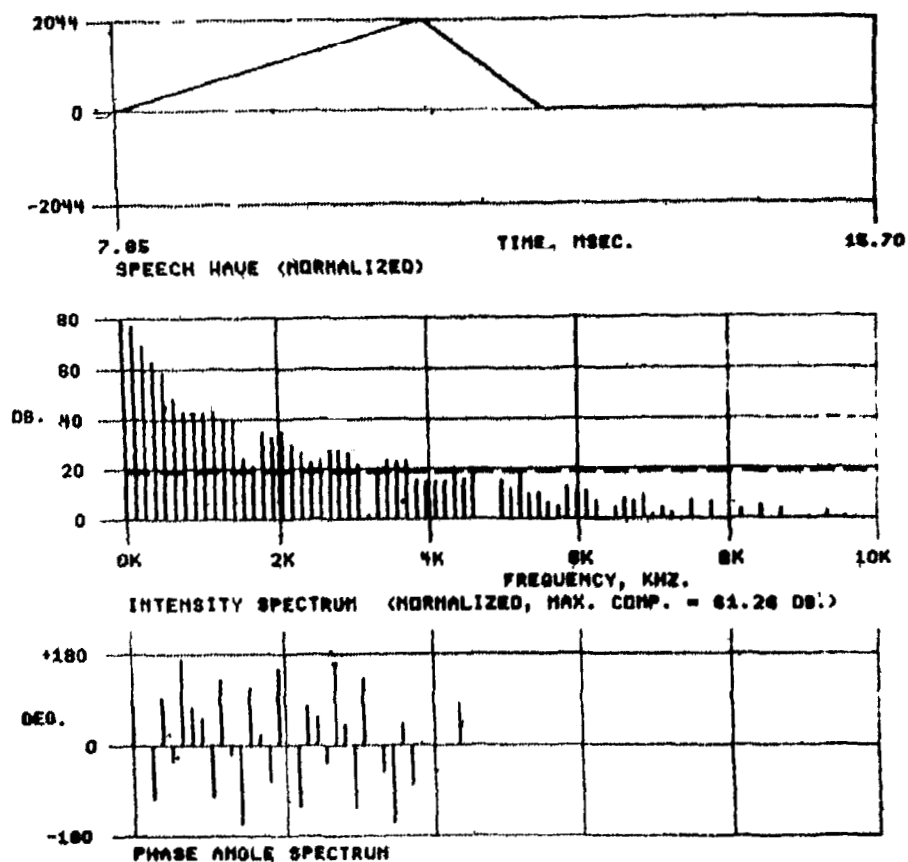


Figure 5. Wave Shape and Spectral Analysis of a Linear Approximation to a Glottal Wave. For details of the display, cf. Figure 4.

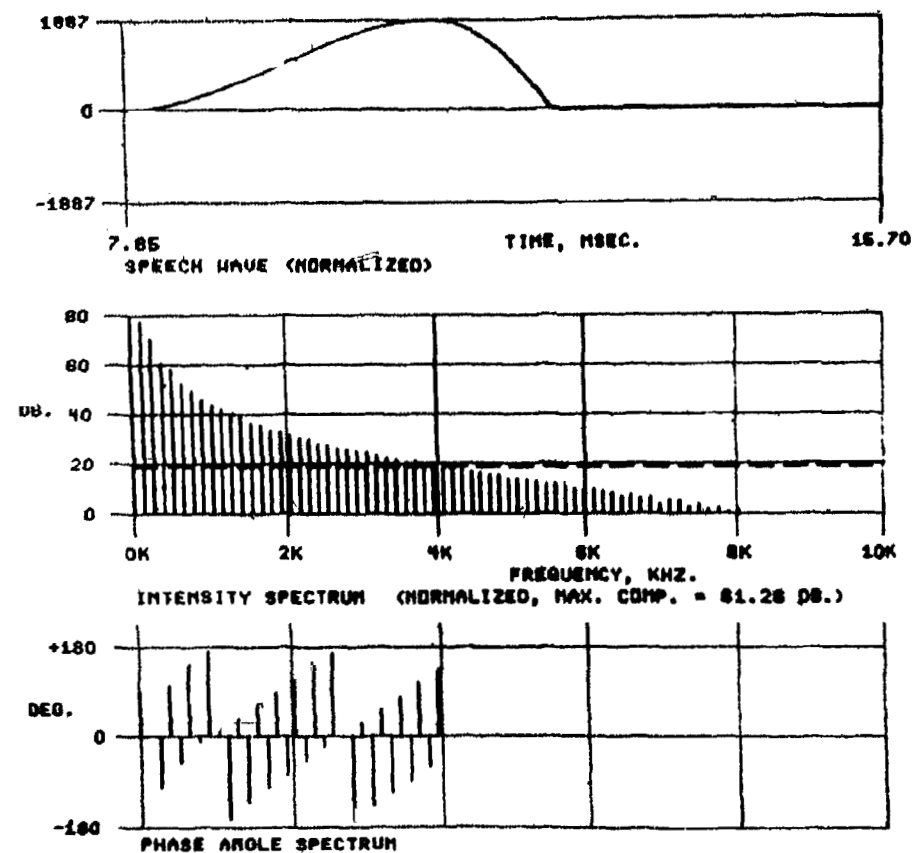


Figure 6. Wave Shape and Spectral Analysis of a Polynomial Approximation to a Glottal Wave. For details of the display, cf. Figure 4.

```

0001      FUNCTION GLOT(P,AV)
0002      COMMON ISWV,TDEL,TG,TP,T1,T2,OPTR,CLTR,AVSAVE
0003      C
0004      C PRODUCES A POLYNOMIAL APPROXIMATION TO A GLOTTAL WAVE
0005      C WITH CONSTANT WAVE SHAPE
0006      C
0007      C USES THE FOLLOWING VARIABLES FROM COMMON: ISWV,
0010      C TDEL,TG,TP,T1,T2,OPTR,CLTR,AVSAVE
0011      C
0012      C FIRST SEE IF WE NEED TO RE=INITIALIZE.
0013      C FOR BEGINNING OF GLOTTAL WAVE
0014      C
0015      C ARE WE NOW GENERATING VOICE?
0016      IF (ISWV) 000,20,10
0017      C YES == IF WE ARE NOT AT THE END OF A PULSE,
0020      C PARAMETERS ARE O.K.
0021      C OTHERWISE WE NEED TO CHECK AV TO SEE
0022      C IF WE NEED ANOTHER PULSE
0023      10 IF (TG+0.5*TDEL=TP) 50,20,20
0024      C EITHER WE HAVE NOT BEEN GENERATING VOICE OR
0025      C WE HAVE JUST FINISHED A PULSE ==
0026      C IF AV > 0, INITIALIZE TO GENERATE A(NOTHER) PULSE
0027      C AND RESET ISWV TO 1
0030      C OTHERWISE (RE)SET ISWV TO 0
0031      20 IF (AV) 30,30,40
0032      30 ISWV=0
0033      GO TO 50
0034      C INITIALIZE FOR ANOTHER PULSE
0035      40 ISWV=1
0036      AVSAVE=AV
0037      TG=0.0
0040      TP=1000.0/P
0041      T1=OPTR+TP
0042      T2=T1+CLTR+TP
0043      50 CONTINUE
0044      C
0045      C END OF PARAMETER SETTING
0046      C
0047      C BEGINNING OF LOGIC TO GENERATE GLOTTAL WAVE
0050      C
0051      IF (ISWV) 57,55,57
0052      55 Y=0.0
0053      GO TO 180
0054      57 CONTINUE
0055      C
0056      C IF TG < T1, Y=AV*(3*(TG/T1)**2+2*(TG/T1)+3)
0057      130 IF (TG=T1) 140,150,150
0060      140 Y=AVSAVE*(3.0+((TG/T1)**2)+2.0+((TG/T1)**3))
0061      GO TO 180
0062      C ELSE IF TG < T2, Y=AV*(1+(((TG-T1)/(T2-T1)**2))
0063      150 IF (TG=T2) 160,170,170
0064      160 Y=AVSAVE*(1.0+(((TG-T1)/(T2-T1)**2))
0065      GO TO 180
0066      C ELSE Y=0
0067      170 Y=0.0
0070      GO TO 180
0071      C
0072      C END OF GLOTTAL PULSE GENERATION
0073      C RETURN VALUE OF GLOTTAL WAVE
0074      C AND INCREMENT TG
0075      C
0076      180 GLOT=Y
0077      TG=TG+TDEL
0100      RETURN
0101      C
0102      C ERROR IN VALUE OF ISWV
0103      C
0104      900 WRITE(1,910)ISWV
0105      910 FORMAT(" ** ERR IN GLOT: ISWV=",I5)
0106      CALL HOLD
0107      CALL EXIT
0110      END
0111      C GLOT

```

Figure 7. A Fortran Function to Generate Glottal Waves

ratio") is the fraction of the wave occupied by the opening phase, and CLTR is the fraction occupied by closing. In the over-all initialization, OPTR should be set to .40 and CLTR to .16, values which maximize naturalness according to Rosenberg's paper.

If the instantaneous values of AV were used by GLOT, the standard wave shape would be altered if AV were changing during generation of a glottal wave. To keep the wave shape constant, GLOT uses the variable AVSAVE to store the value of AV at the beginning of each pitch period, and during the generation of the pulse, AVSAVE is used as the (constant) amplitude.

Between calls to GLOT, the values of ISWV, TG, TP, T1, T2, and AVSAVE should not be altered.

Rosenberg's equations for the polynomial approximation are used in GLOT; to get the linear approximation, the following two lines of Fortran should be substituted in GLOT for lines number 60 and 64:

```
140     Y=AVSAVE*(TG/T1)
160     Y=AVSAVE*(1.0-(TG-T1)/(T2-T1))
```

2. White Noise Generator

Almost any reasonably good random-number generator can be used as a source of white noise. If the spectrum of the random numbers produced is flat, it will be easier to shape into the desired spectra for the different fricative sounds.

The algorithm we use was developed for use in synthetic speech work; it is very fast, and produces noise with a quite flat spectrum. Its presentation by Rader, Rabiner, Schafer,

and Perry¹ is easy to follow and implement. The logic of the algorithm is formally stated in Fortran in Figure 8, page 18, but if possible this should be one function coded in assembly language: if the right machine instructions are available, it will be a snap, but "bit fiddling" in Fortran is very slow.

Our function IRN4(X) -- X is a dummy variable required by our Fortran compiler -- contains this algorithm in assembly language, producing on successive calls a series of random numbers with a uniform distribution over the interval from -2047 to +2047. To implement a white noise generator, only this line of coding is needed:

$$Y=AN*(FLOAT(IRN4(X))/2047.0)$$

where AN is a variable whose value is the amplitude of noise desired.

A typical stretch of noise produced in this manner, along with its spectrum, is given as Figure 9, page 19. Note that there does not appear to be any significant deviation from flatness in the spectrum intensity.

B. Spectral Shaping Elements

1. Resonances and Anti-resonances

We use recursive equations, a technique developed by electrical engineers, to simulate resonant and anti-resonant (notch) filters as elements to shape spectra. Each individual filter can be represented by a second-order linear differential

1. Rader, Rabiner and Schafer (1970) and Perry, Schafer, and Rabiner (1972)

```

FUNCTION IRN4(X)
COMMON NM1(19), NM2(19)
DIMENSION NX1(19), NX2(19)
C FORM BIT-WISE EXCLUSIVE OR OF NM1, NM2
DO 10 I=1,19
10  NX1(I)=MOR(NM1(I), NM2(I))
C ROTATE NX1 8 PLACES TO THE RIGHT
DO 20 I=1,11
20  NX2(I+8)=NX1(I)
DO 30 I=12,19
30  NX2(I-11)=NX1(I)
C SHIFT PAST VALUES
DO 40 I=1,19
NM2(I)=NM1(I)
40  NM1(I)=NX2(I)
C RETURN VALUE OF LEFT-MOST 12 BITS OF NX2
IRN4=MINT(NX2)
C END
RETURN
END

```

Figure 8. Random Number Generator Documented in Fortran. NM1(19) and NM2(19) are arrays whose elements have either the value 0 or 1. MOR(N1, N2) is a function returning the exclusive or of N1 and N2, variables having either the value 0 or 1. MINT(N) is a function whose argument N is a bit-string array such as NM1 and which returns a 12-bit integer value consisting of the left-most 12 elements of N packed into a single word.

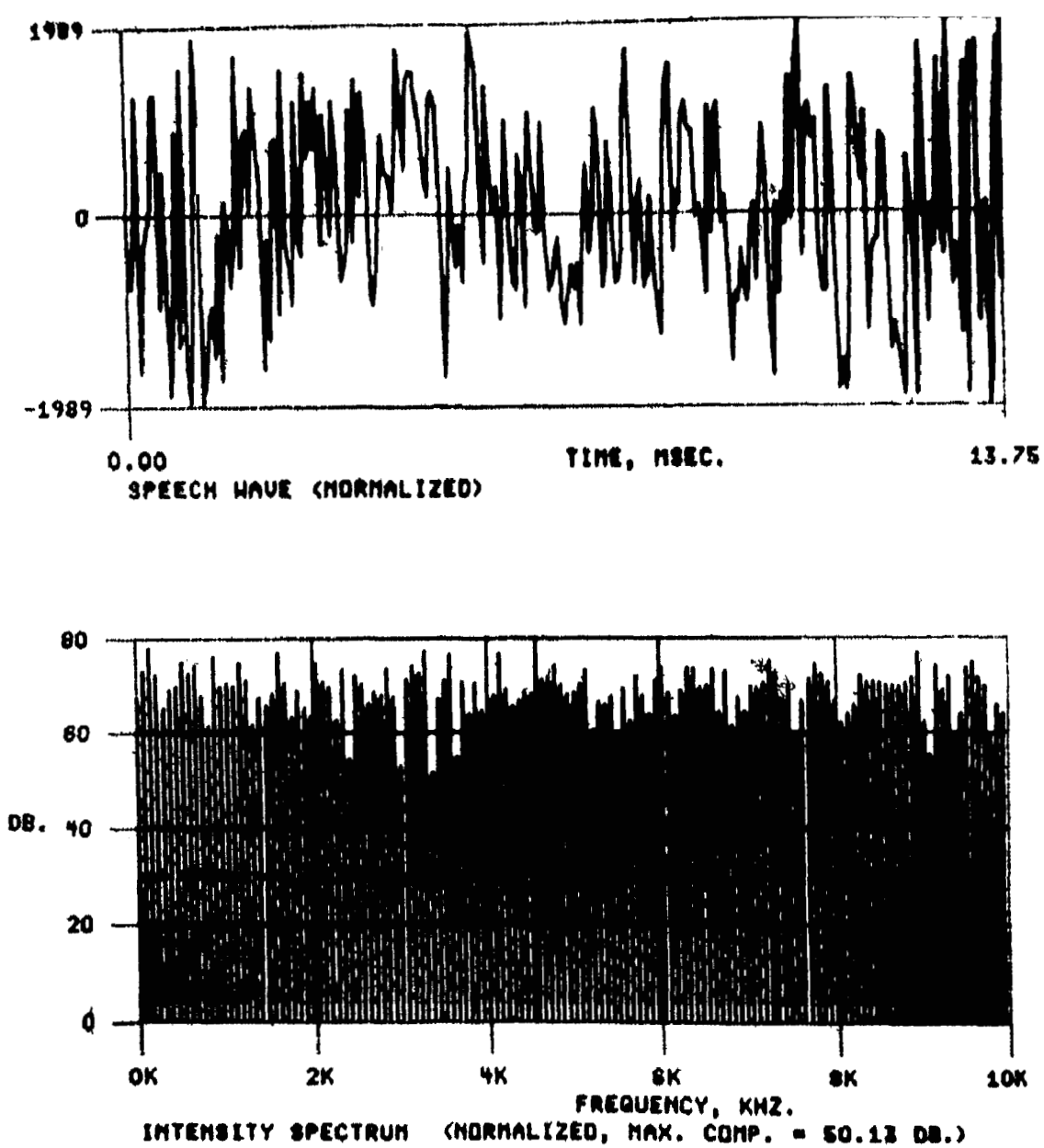


Figure 9. Typical Wave Form and Spectral Analysis of the Output From the White Noise Generator. For details of the display cf. Figure 4.

equation, giving only one resonance or anti-resonance. For those who feel at home in the s-plane, the recent book Speech Synthesis edited by Flanagan and Rabiner contains reprints of papers developing the theory of recursive equation filter simulation: for the rest of us, the paper by Lovell et al. (1973) is a clear presentation, with some more general Fortran algorithms than will be given here.

Figure 10, page 21, gives one Fortran subroutine and two Fortran functions which we use to simulate resonant and anti-resonant filters.

The functions RES and ARES return the output values of simple resonant (conjugate pole pair) and anti-resonant (conjugate zero pair) filters, respectively. A0, A1, and A2 are coefficients used in the recursive equations. YM1 and YM2 are remembered previous values of the signal, and Y is the input to the filter. A0, A1, and A2 determine the characteristics of the filter: center frequency and bandwidth. Each simulated filter should have its own variables in which to save the values of YM1 and YM2, and between calls to the function simulating that filter, the values of these variables should not be changed.

The subroutine COEFF is used to calculate appropriate values for A0, A1, and A2, based on CF, the center frequency, and BW, the bandwidth, of the resonance or anti-resonance.¹ SR is the output sample rate, and MPZ tells the subroutine

1. In the version of this paper presented at the 12th annual A.C.L. meeting, there was an error in line 16 of subroutine COEFF

```

0001          SUBROUTINE COEFF(CF,BW,A0,A1,A2,SR,MPZ)
0002          C
0003          C      COMPUTES THE RECURSIVE EQUATION COEFFICIENTS
0004          C      A0,A1,A2 FOR EITHER A RESONANT OR ANTIRESONATE FILTER
0005          C      SPECIFIED BY CENTER FREQUENCY CF AND BANDWIDTH BW (HZ.)
0006          C      SR IS THE SAMPLE RATE (SAMPLES/SEC)
0007          C      IF MPZ=1, FILTER WILL BE A RESONANCE
0010          C      IF MPZ=0, FILTER WILL BE AN ANTIRESONANCE
0011          C      PI=3.14159265
0012          C      A=PI*BW/SR
0013          C      B=2.0*PI*CF/SR
0014          C      A2=EXP(-2.0*A)
0015          C      A1=2.0*EXP(-A)*COS(B)
0016          C      A0=1.0-A1+A2
0017          C      IF (MPZ) 20,10,20
0020          10      A0=1.0/A0
0021          20      RETURN
0022          END

```

```

0001          FUNCTION RES(Y,YM1,YM2,A0,A1,A2)
0002          C
0003          C      SIMULATES RESONATOR (CONJUGATE POLE PAIR)
0004          C      GIVEN BY RECURSIVE EQUATION COEFFICIENTS A0,A1,A2
0005          C      YM1 AND YM2 ARE PAST VALUES OF Y; THEIR VALUES
0006          C      MUST BE SAVED
0007          C      YRES=A0*Y+A1*YM1+A2*YM2
0010          C      YM2=YM1
0011          C      YM1=YRES
0012          C      RES=YRES
0013          C      RETURN
0014          END

```

```

0001          FUNCTION ARES(Y,YM1,YM2,A0,A1,A2)
0002          C
0003          C      SIMULATES ANTIRESONATOR (CONJUGATE ZERO PAIR)
0004          C      GIVEN BY RECURSIVE EQUATION COEFFICIENTS A0,A1,A2
0005          C      YM1 AND YM2 ARE PAST VALUES OF Y; THEIR VALUES
0006          C      MUST BE SAVED
0007          C      TEMP=Y*A0
0010          C      ARES=TEMP-A1*YM1-A2*YM2
0011          C      YM2=YM1
0012          C      YM1=TEMP
0013          C      RETURN
0014          END

```

Figure 10. Fortran Implementation of Elemental Filters

whether to compute coefficients for a resonance or an anti-resonance

The spectral effect of a resonance with center frequency of 3000 Hz and bandwidth of 200 Hz is illustrated in Figure 11, page 23, while Figure 12, page 23, is a similar illustration of the effect of an anti-resonance of the same center frequency and bandwidth. In these figures, the first graph shows output for an impulse input and the second graph shows the normalized intensity spectrum of that output.

2. Radiation Effect

The effect on the spectrum of the radiation of sound from the lips through a short stretch of air can be reasonably approximated by a differentiator.¹ Figure 13, page 24, gives a simple Fortran function RAD simulating this effect. Y is the speech wave input to the simulator, YM1 is the remembered immediately previous value of Y, and G is a normalizing gain control which should be calculated in the over-all initialization as a direct function of the output sample rate, some K times OSR. The spectral effect of RAD, approximately a 6 dB/octave rise, is illustrated in Figure 14, page 24.

IV. Organization of Elements

A. The Simplest Model

The simplest reasonable model for connecting these elements, which we have taken to calling "Model T", is given in block diagram form in Figure 15, page 25. We use this organization in our currently running synthesizer. The three variable

1. Rabiner (1968) p. 823

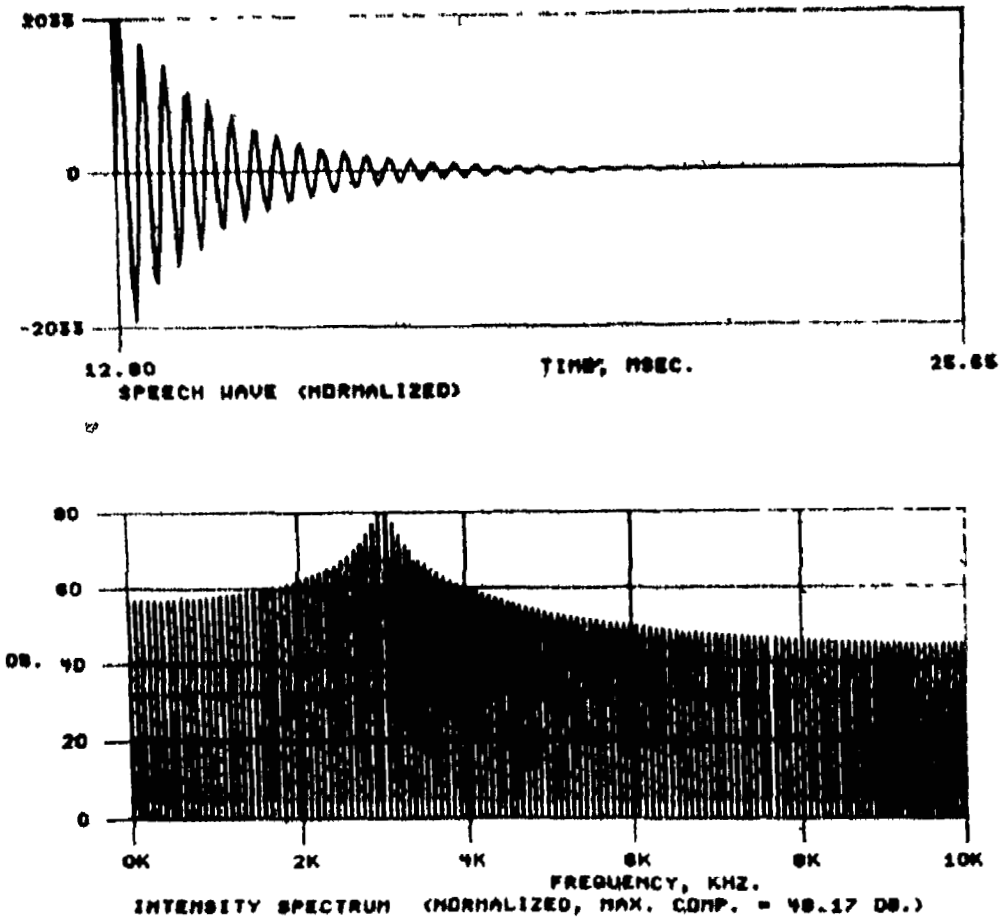


Figure 11. Spectral Shaping Effect of an Elemental Resonant Filter with $CF=3000$ Hz and $BW=200$ Hz.

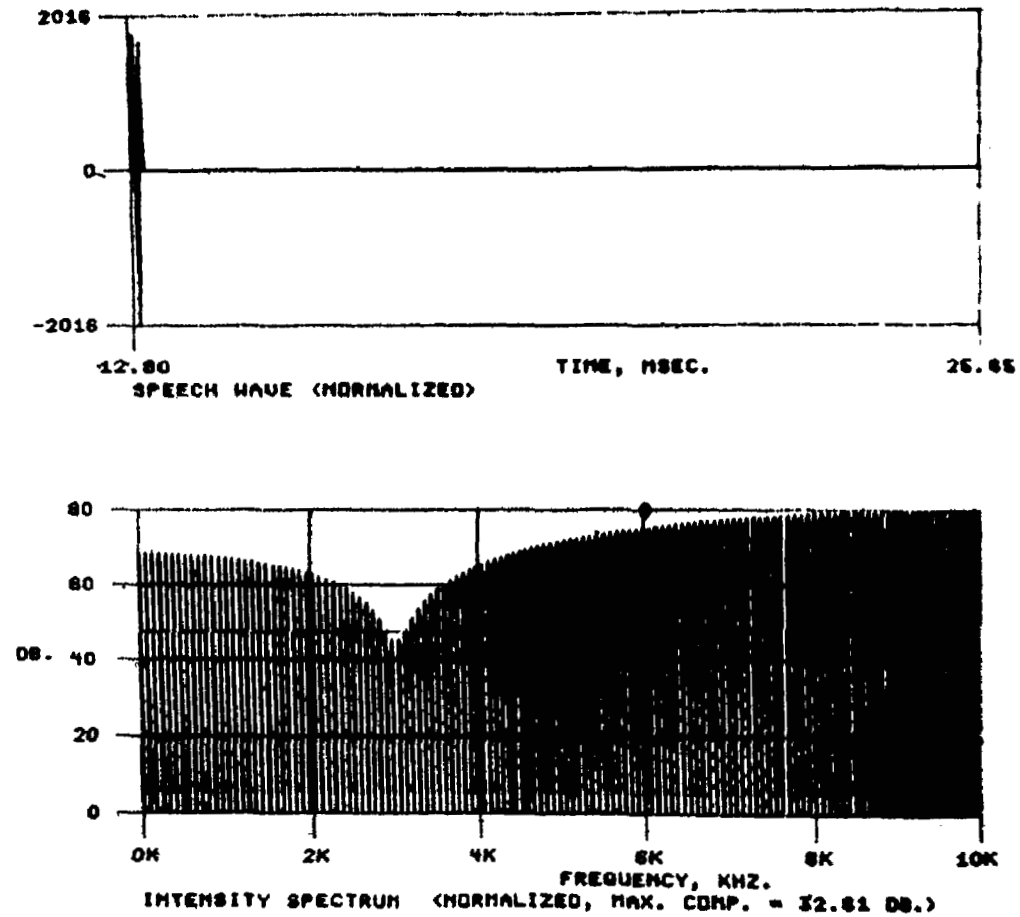


Figure 12. Spectral Shaping Effect of an Elemental Anti-Resonant Filter with $CF=3000$ Hz and $BW=200$ Hz.

```

0001          FUNCTION RAD(Y,YM1,G)
0002          C
0003          C      SIMULATES RADIATION EFFECT
0004          C      BY TAKING FIRST TIME DERIVATIVE OF Y
0005          C
0006          RAD=G*(Y-YM1)
0007          YM1=Y
0010          RETURN
0011          END

```

Figure 13. Fortran Function to Simulate Radiation Effect.

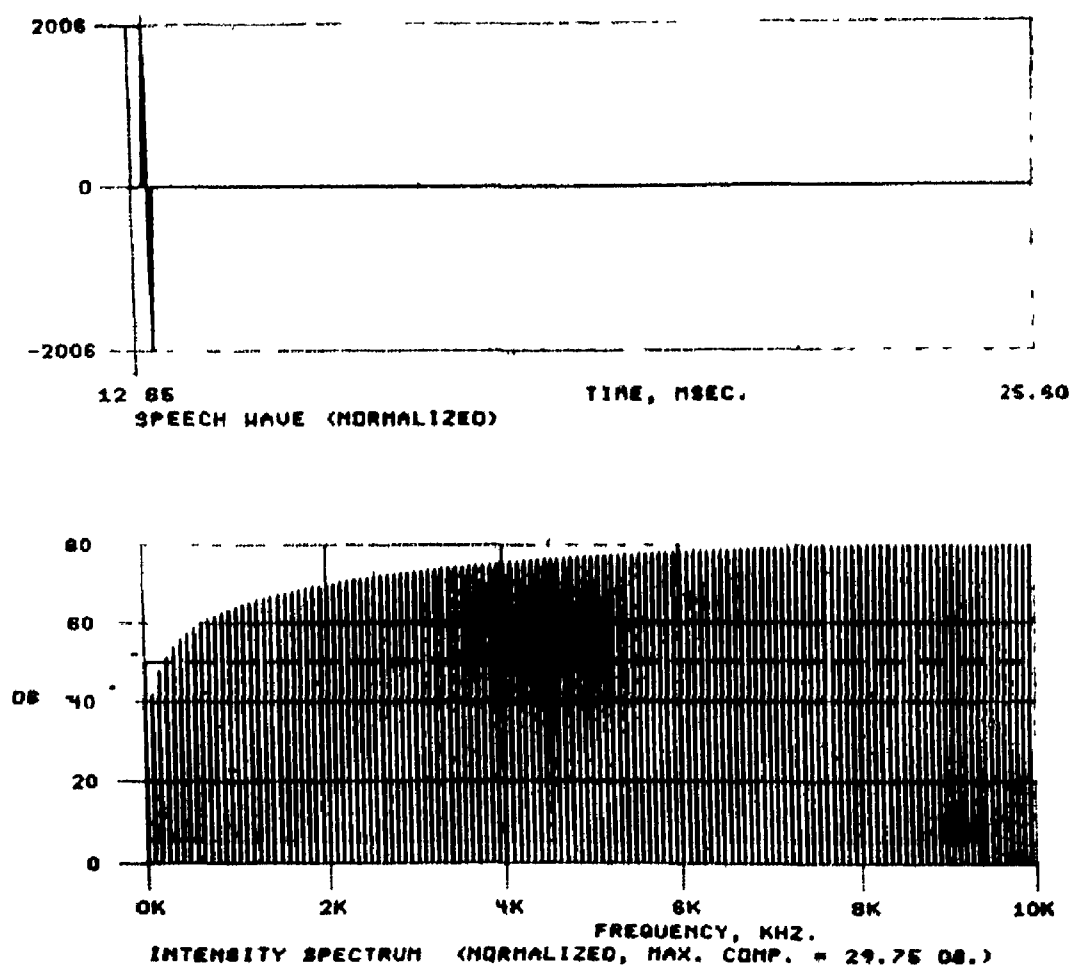


Figure 14. Spectral Shaping Effect of Radiation as Simulated by Function RAD.

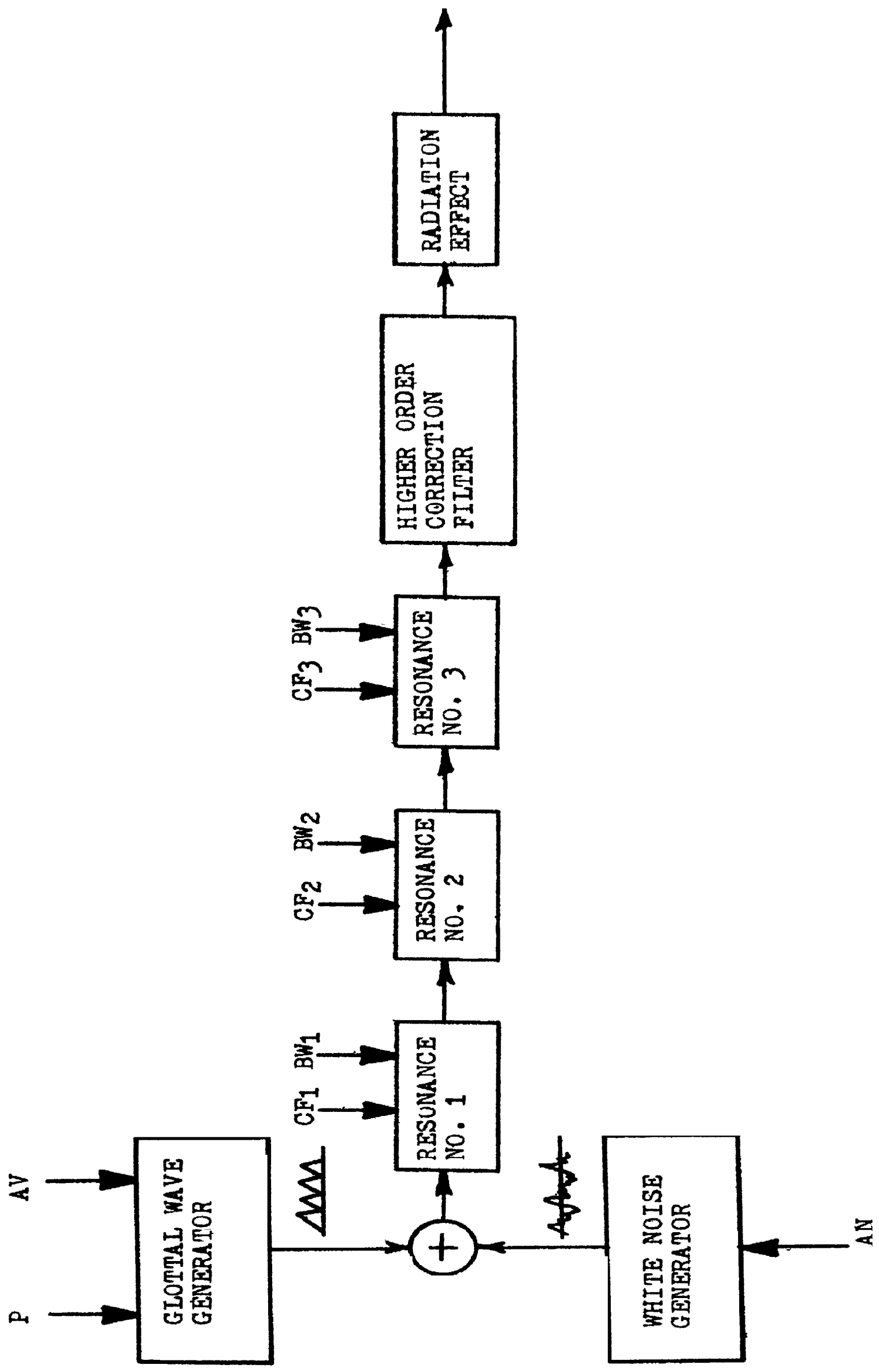


Figure 15. "Model T" Organization Block Diagram

resonant filters are used to make the formants of voiced speech and, in a rather strained fashion, to shape the noise spectrum during frication and aspiration.

All of the elements in this figure should now be familiar except the one called "higher order correction filter." This is a series of resonant filters of fixed center frequency and bandwidth which compensate for the effect of higher-frequency resonances present in a real vocal tract but absent in a digital simulation of this kind. Their use is discussed in Rabiner (1968) from which the values presented in Figure 16, page 27, were taken. These are the values to use for center frequency and bandwidth of the higher order correcting filters. Only higher-order filters with center frequency less than $\frac{1}{2}$ the output sample rate should be used. The recursive equation coefficients A_0 , A_1 , and A_2 need be calculated only once, in the over-all initialization.

Theoretically, the order of computation of the series elements such as those in the main stem of Model T, makes no difference. However, because the digital numbers are finite in length round-off or truncation errors are introduced at each step of the computation. The overall error increases as the number of computational steps increase. Some types of computation such as differentiation tend to increase the error, while other types such as integration tend to decrease the error. For this reason, overall system error is related in a complex way to the order of computation. An understanding of error buildup and testing of the various algorithms will help in choosing the computational sequence that results in smallest errors. In the case of cascaded resonators, it is better to perform the computation in reverse order from that implied by Fig. 15 - radiation effect first, then higher order filters in descending center frequency order, then formant filters.

Resonator No.	Center Freq. (Hz)	Bandwidth (Hz)
4	3500	175
5	4500	281
6	5500	458
7	6500	722
8	7500	1250
9	8500	2125
10	9500	4750

Figure 16. Higher Order Correction Filter Center Frequencies and Bandwidths. From Rabiner (1968)

```

C ZERO VARIABLE HOLDING NEXT SPEECH WAVE POINT
      YN=0.0
C ADD GLOTTAL WAVE
      YN=YN+GLOT(P,AV)
C ADD FRICATIVE NOISE
      YN=YN+AN*(FLOAT(IRN4(X))/2047.0)
C APPLY FORMANT FILTERS
      DO 200 I=1,3
200    YN=RES(YN,YM1(I),YM2(I),AO(I),A1(I),A2(I))
C APPLY HIGHER ORDER CORRECTING FILTERS
      DO 250 I=1,7
250    YN=RES(YN,HYM1(I),HYM2(I),HAO(I),HA1(I),HA2(I))
C,APPLY RADIANCE EFFECT
      YN=RAD(YN,RYM1,GRAD)

```

Figure 17. Model T Logic in Fortran.

The conversion of the Model T block diagram into Fortran is illustrated by Figure 17, page 27, the series elements being here computed in their natural order. This coding is an example of what should be inserted into the over-all logic (Figure 3, page 9) following the comment lines "GENERATE NEXT SPEECH WAVE POINT...".

B. Control

In the Model T organization, nine control parameters are available -- P (pitch), AV (amplitude of voicing), AN (amplitude of noise) and the center frequency and bandwidth of three variable formant filters. In the main loop, just before generating the next speech wave point, a subroutine (it can be in-line code, of course) calculating values of these parameters is needed. If at the beginning of the program the variable T (time) is initialized to zero and incremented by TDEL at the end of the main loop, it can serve as a simulated-time clock on which to base calculation of the control parameters. The simplest method of control is to formulate the desired control parameter curves algebraically and just

include Fortran statements in this section calculating their values as in the algebraic equations. For example, suppose we wanted the pitch to rise linearly from 80 to 120 Hz in the first 100 msec, stay constant at 120 Hz for 200 msec, then fall linearly to 100 Hz in the next 100 msec and stay at that value from then on. The following Fortran statements can be used to calculate P:

```

                IF (T-100.0) 210,210,220
210             P=80.0+40.0*T/100.0
                GO TO 270
220             IF (T-300.0) 230,230,240
230             P=120.0
                GO TO 270
240             IF (T-400.0) 250,250,260
250             P=120.0-20.0*(T-300.0)/100.0
                GO TO 270
260             P=100.0
270             CONTINUE

```

When new values of CF and BW are computed for the three variable formant filters, subroutine COEFF should be called to translate these into the coefficients A0, A1, and A2 actually used by function RES.

If new values of the control parameters are calculated every sample point, the execution time of the program will be very long. There are several obvious ways to speed up this calculation.

One way is to calculate new values for P and AV only at the beginning of each pitch period, since this is the only time GLOT uses them.

With some error introduced, the control parameters can be re-computed only every so many msec to speed things up. A variable used as a time clock in the same way that TG is used by GLOT can control this period. Computing the source control parameters (P, AV, and AN) this way introduces error only in that the actual parameter curves will follow the desired curve in a step-wise fashion, but changing the characteristics of the formant filters this way will introduce another type of error, which comes out sounding like clicks or static if the change in filter characteristics is too large.

Our currently implemented assembly-language synthesizer reads a file of tabled values created by another program as values representing the parameter curves. The period between tabled parameter values is changeable, but on the order of 5 to 10 msec. In computing the actual parameter values used, the synthesizer interpolates linearly along the tabled parameter data curves. The step size of the interpolation can be easily changed, allowing a smooth trade-off between accuracy and execution time.

To sum up, computing new control parameter values for each sample point generated is the easiest and most accurate way, but alternative schemes allowing a convenient trade of accuracy for speed are easily programmed

C. Other Models

We will briefly describe several alternative organizations of the elements, although most of our practical experience has been with the Model T organization.

1. Parallel Formant

One model used in some synthesizers is the parallel formant model, whose block diagram is given as Figure 18, page 32. In serial formant models such as Model T, no independent control of the relative intensities of formants is possible, since the order of operations is immaterial. It has been shown that the relative intensities of formants in a serial synthesis closely match those found in natural speech,¹ which is some justification of the serial model as an analog of the vocal tract. But in a parallel formant arrangement, each parallel channel must have a separate gain control. This is fine if you're investigating the perception of relative formant intensities, but not many have chosen this model for general speech synthesis

Rabiner (1968) contains a worthwhile discussion of the relative merits of serial and parallel synthesis.² If you decide to try a parallel formant model, the higher order correcting filters are apparently unnecessary, and Rabiner (1968) mentions that zeros -- anti-resonances -- are introduced into the spectrum.

2. Separate Noise Shaping Channel

In Model T, the same three filters are used to make the formants of voiced speech and to shape the spectrum of noise during unvoiced speech. This is cumbersome and difficult, and a simple alternative is illustrated in Figure 19, page 34.

1. Fant (1956)

2. so does Flanagan (1957)

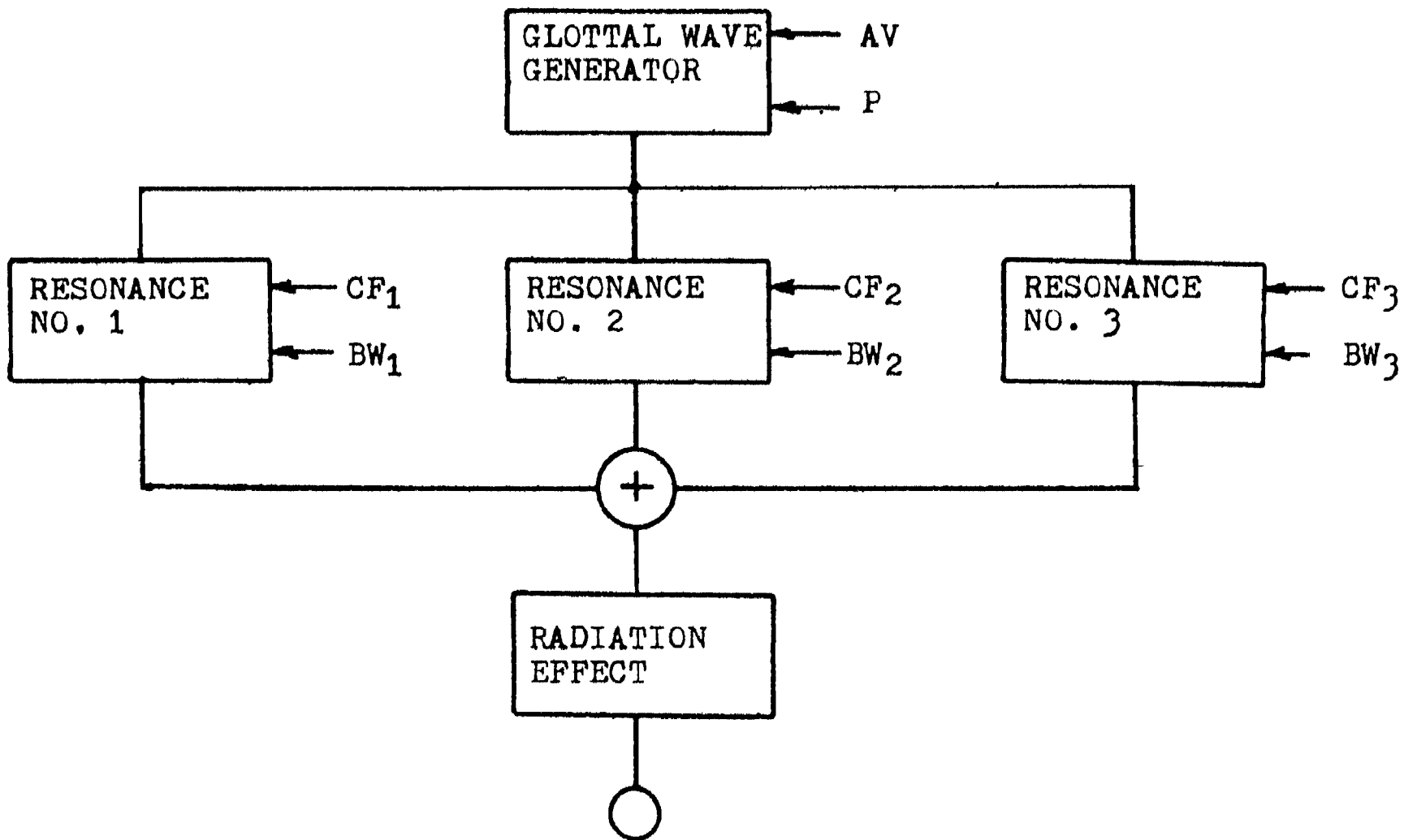


Figure 18. Parallel Formant Organization Model

A separate channel is devoted to noise, with its own resonant and anti-resonant filters for spectral shaping. It has been suggested that one resonance and one anti-resonance are sufficient to model most English fricatives.¹ Of course, this model adds two new control parameters to be computed.

3. Other More Complex Models

Model T does not use anti-resonances; they are not typical of voiced speech, but rather are present in the spectra of fricatives and nasalized segments. For making nasal sounds, a parallel nasal channel whose input is the glottal wave and whose output is added in just before the radiance effect calculation can be added. The spectral shaping filters needed in this channel are not obvious from published reports, but one variable anti-resonance and several fixed resonances are probably a minimum complement.

A multitude of more complex models can be seen in the literature -- Rabiner (1968), to take one example, includes a special arrangement for generating voiced fricatives.

D. A Complete Example

To illustrate the capabilities of the simplest synthesis model, on the following pages we present as Figure 20 a complete Fortran program for synthesizing the word "seat" ($[sit^h]$). We tried to duplicate one particular token utterance of this word. Spectrograms of the original sound used as a model and the synthesized sound calculated by the Fortran program are shown in

1. Heinz and Stevens (1961)

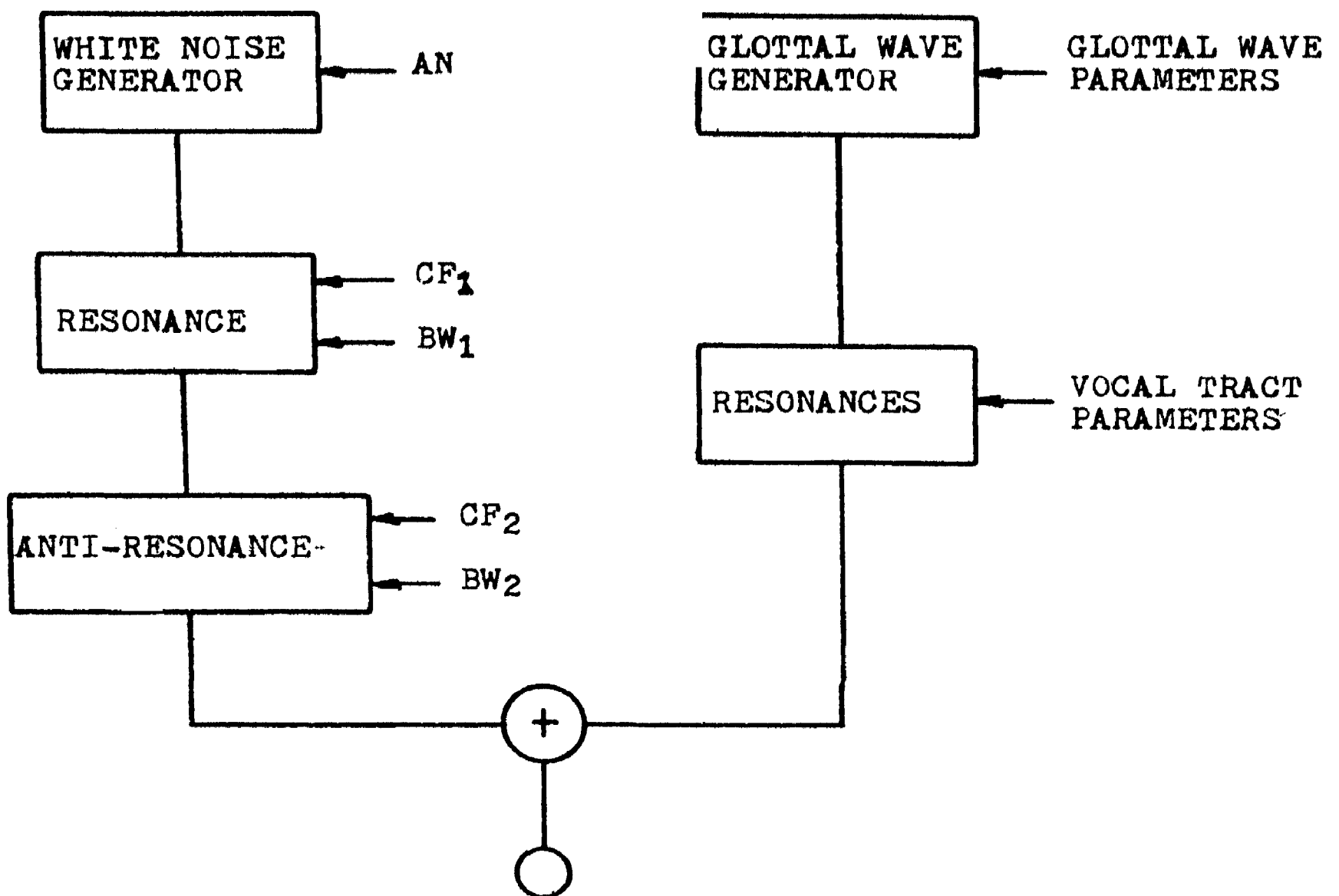


Figure 19. Block Diagram of Model With Separate Noise Shaping Channel

Figure 21, page 44. Deriving the parameter curves from the token utterance took a lot of work, but as Figure 21 shows, the resulting synthetic word is a reasonably close copy of the original.

For those who may want to use this program as a beginning to their work, several features of it will be explained.

The basic model used is the simplest, model "T" (Figure 15, page 25) with one addition: the noise signal is multiplied by a relative gain constant (GFRIC) before entering the vocal tract (variable filter) section.

The output sound wave is stored a block at a time in file RAPWRK1. This file is opened by the subroutine called in line 43 written into in line 455, and closed in line 470.

Parameter values are periodically calculated from piece-wise polynomial algebraic specifications in the section called "GPAR", lines 142 to 415. The periods between parameter re-calculations are controlled by two variables serving as clocks, TVOC for voicing parameters and TFRIC for frication parameters. The values of PVOC and PFRIC are the times in msec between re-calculations of vocalic and fricative parameter values, respectively. These parameter values can be reset more often by merely changing the values assigned to PVOC and PFRIC in lines 33 and 37; at present frication parameters are reset every 0.1 msec and vocalic parameters every 0.2 msec. The Fortran logic calculating the parameters was coded for clarity, not economy, and though lengthy should be easy to follow. The primitive subroutine TPOW returns powers of a variable for ease in calculating polynomial functions of time: after calling TPOW(T,N, TX) $TX(1)=T^1$, $TX(2)=T^2$, ... $TX(N)=T^N$

Certain parameters are constant during some of the sounds, e.g., spectral parameters during "S". As a minor concession to execution speed, these constant parameters are not reset after the first entry into the section during which they are constant. The variables ISW1, ISW2, and ISW3 are "first-time-through" switches, used to remember whether or not the temporarily constant parameters have been calculated yet.

We have found that duplicating a token of natural speech using this simple basic synthesis model, though possible, can be quite difficult, requiring much trial-and-error work. Fortunately, much research on speech perception does not require exact duplication of given utterances, but instead uses simpler sets of parameter curves. An example of such a program, which synthesizes the vowel /i/ with constant pitch and intensity can be made by substituting the following code for the "GPAR" Section, lines 151 to 411 of the sample program.

```
                IF (ISW1) 150,100,150
100             ISW1=1
                P=100.0
                AVDB=50.0
                AV=10.0**(AVDB/20.0)
                AN=0.0
                CF(1)=270.0
                CF(2)=2290.0
                CF(3)=3010.0
                BW(1)=54.0
                BW(2)=55.0
                BW(3)=170.0
                _____
                DO 110 I=1,3
110             CALL COEFF(CF(I),BW(I),A0(I),A1(1),A2(I),OSR,1)
150             CONTINUE
```

```

0001      C      FORTRAN SYNTHESIS TESTER
0002      C      WSYN18.FO
0003      C      SYNTHESIZES "SEAT"
0004      C
0005      COMMON ISWV,DT,TG,TP,T1,T2,OPTR,CLTR,AVSAVE
0006      DIMENSION IBUF(256),CF(10),BW(10),A0(10),A1(10),A2(10),
0007      YM1(10),YM2(10),TX(10)
0010      C
0011      C
0012      C      O V E R A L L   I N I T I A L I Z A T I O N
0013      C
0014      C
0015      C SET OUTPUT SAMPLE RATE IN SAMPLES/SEC
0016      OSR=20000.0
0017      C SET OUTPUT BLOCK SIZE AND NO. OF BLOCKS
0020      NBSIZE=256
0021      NBLKS=51
0022      C SET WAVE SHAPE CONSTANTS FOR GLOTTAL WAVE GENERATION
0023      OPTR=0.40
0024      CLTR=0.16
0025      C SET OVERALL TIME CLOCK AND DELTA T IN MSEC.
0026      T=0.0
0027      DT=1000.0/OSR
0030      C SET PARAMETER=RESETTING CLOCK FOR FRICATION
0031      TFRIC=0.0
0032      C SET PARAMETER=RESETTING PERIOD FOR FRICATION
0033      PFRIC=0.1
0034      C SET PARAMETER=RESETTING CLOCK FOR VOICING
0035      TVOC=0.0
0036      C SET PARAMETER=RESETTING PERIOD F
0037      PVOC=0.2
0040      C OPEN OUTPUT STORAGE FILE
0041      C OUTPUT BUFFER IS ARRAY IBUF
0042      NOUT=5
0043      CALL OPENR(NOUT,"RAPWRK1 ", "DD",1,LENG,IBUF,X10)
0044      GO TO 20
0045      C ERROR HANDLING IF OUTPUT FILE CANNOT BE OPENED
0046      10      WRITE(1,15)
0047      15      FORMAT(" *** ERROR OPENING WORK FILE")
0050      GO TO 900
0051      C FILE OPENED O.K.,) CHECK IF BIG ENOUGH
0052      20      IF(NBLKS=LENG) 35,35,25
0053      C ERROR HANDLING -- FILE NOT BIG ENOUGH
0054      25      WRITE(1,30)
0055      30      FORMAT(" *** WORK FILE NOT BIG ENOUGH")
0056      GO TO 900
0057      C OUTPUT WORK FILE READY TO GO
0060      C SET VOICING SWITCH OFF
0061      35      ISWV=0
0062      C INITIALIZE HIGHER ORDER FILTER VALUES
0063      C IN REVERSE ORDER BY CENTER FREQUENCY
0064      CF(4)=9500.0
0065      CF(5)=8500.0
0066      CF(6)=7500.0
0067      CF(7)=6500.0
0070      CF(8)=5500.0
0071      CF(9)=4500.0

```

Figure 20. Example Program Synthesizing "Seat".

```

0072          CF(10)=3500.0
0073          BW(4)=4750.0
0074          BW(5)=2125.0
0075          BW(6)=1250.0
0076          BW(7)=722.0
0077          BW(8)=450.0
0100          BW(9)=281.0
0101          BW(10)=175.0
0102          C COMPUTE RECURSIVE EQUATION COEFFICIENTS FOR HIGHER ORDER FILTERS
0103          DO 40 I=4,10
0104          40      CALL COEFF(CF(I),BW(I),A0(I),A1(I),A2(I),OSR,1)
0105          C ZERO INITIAL CONDITIONS OF FILTERS (REMEMBERED PAST VALUES)
0106          DO 50 I=1,10
0107                   YM1(I)=0.0
0110          50      YM2(I)=0.0
0111          C SET OVERALL GAIN FACTORS
0112                   GAIN=1.0/1000.0
0113                   GFRIC=1.0/1000000.0
0114          C CLEAR UTILITY SWITCHES
0115                   ISW1=0
0116                   ISW2=0
0117                   ISW3=0
0120          C
0121          C
0122          C      E N D      O F      O V E R A L L      I N I T I A L I Z A T I O N
0123          C
0124          C
0125          C
0126          C
0127          C      M A I N      P R O C E S S I N G      L O O P
0130          C
0131          C
0132          C LOOP ON NO. OF BLOCKS
0133          DO 800 IBLK=1,NBLKS
0134          C LOOP ON OUTPUT POINTS PER BLOCK
0135          DO 750 IPT=1,NBSIZE
0136          C ZERO YN
0137                   YN=0.0
0140          C GET NEW PARAMETER VALUES
0141          C
0142          C*****
0143          C*
0144          C*
0145          C*          GPAR
0146          C*
0147          C* IN=LINE SUBROUTINE TO GET NEW PARAMETER VALUES
0150          C*
0151          C DECREMENT PARAMETER=RESETTING CLOCK FOR FRICATION
0152                   TFRIC=TFRIC-DT
0153          C DECREMENT PARAMETER=RESETTING CLOCK FOR VOICING
0154                   TVOC=TVOC-DT
0155          C "S" ?
0156          IF (T=219.51) 100,200,200
0157          100      CONTINUE
0160          C
0161          C          "S"
0162          C

```

Figure 20 (continued)

```

0163      2 IF FIRST ENTRY INTO "S" ROUTINE, SET CONSTANT PARAMETERS
0164          IF (ISW1) 105,105,115
0165      105      AV=0.0
0166          CF(1)=4000.0
0167          CF(2)=5700.0
0170          CF(3)=9000.0
0171          BW(1)=1500.0
0172          BW(2)=3000.0
0173          BW(3)=4000.0
0174          DO 110 I=1,3
0175      110      CALL COEFF(CF(I),BW(I),A0(I),A1(I),A2(I),OSR,1)
0176          ISW1=1
0177      C IF TIME TO DO SO, RESET VARIABLE PARAMETERS
0200      115      IF (TFRIC) 120,120,125
0201      120      CONTINUE
0202      C IF T<5, NO NOISE
0203          IF (T=5.0) 122,124,124
0204      122      AN=0.0
0205          GO TO 125
0206      124      CALL TPOW(T,4,TX)
0207          ANDB=+3.50+1.6375*TX(1)+0.02032*TX(2)+0.0001114*TX(3)
0210          1 =0.0000002242*TX(4)
0211          AN=10.0** (ANDB/20.0)
0212      125      CONTINUE
0213          GO TO 650
0214      C
0215          C          END OF "S"
0216      C
0217      C "EA" VOWEL?
0220      200      IF (T=440.90) 205,310,310
0221      C
0222          C          "EA"
0223      C
0224      C IF FIRST TIME THRU, SET CONSTANT PARAMETERS
0225      205      IF (ISW2) 210,210,215
0226      210      AN=0.0
0227          BW(1)=54.0
0230          BW(2)=55.0
0231          BW(3)=170.0
0232          ISW2=1
0233      C IF TIME, RESET VARIABLE PARAMETERS
0234      215      IF (TVOC) 220,220,650
0235      220      CONTINUE
0236      C PITCH
0237          CALL TPOW(T=330.0,4,TX)
0240          P=146.243+0.4251854*TX(1)+0.0001925078*TX(2)
0241          1 =0.0000005018408*TX(3)+0.0000002460681*TX(4)
0242      C AMPLITUDE OF VOICING
0243          IF (T=245.0) 410,410,420
0244      410      CALL TPOW(T=219.51,3,TX)
0245          AVDB=43.493+0.619573*TX(1)+0.02476343*TX(2)
0246          1 +0.0002630669*TX(3)
0247          GO TO 510
0250      420      IF (T=290.0) 430,430,440
0251      430      AVDB=47.34763+0.1457143*(T=245.0)
0252          GO TO 510
0253      440      IF (T=335.0) 450,450,460

```

Figure 20. (continued)

```

0254      450      AVDB=40.8=0.072*(T=290.0)
0255      GO TO 510
0256      460      IF (T=400.0) 470,470,480
0257      470      AVDB=37.29645+0.045*(T=335.0)
0260      GO TO 510
0261      480      IF (T=420.0) 490,490,500
0262      490      CALL TPOW(T=400.0,2, TX)
0263      AVDB=39.86422=0.07127*TX(1)+0.02243*TX(2)
0264      GO TO 510
0265      500      CALL TPOW(T=420.0,2, TX)
0266      AVDB=29.48846=1.379416*TX(1)+0.02857102*TX(2)
0267      510      AV=10.0** (AVDB/20.0)
0270      C CF1
0271      IF (T=260.0) 225,226,226
0272      225      CF(1)=250.0+1.23*(T=219.51)
0273      GO TO 230
0274      226      IF (T=395.0) 227,228,228
0275      227      CF(1)=300.0
0276      GO TO 230
0277      228      CF(1)=300.0+4.8*(T=395.0)
0300      230      CONTINUE
0301      C CF(2)
0302      IF (T=320.0) 232,232,234
0303      232      CALL TPOW(T=219.51,2, TX)
0304      CF(2)=1732.857+14.95833*TX(1)+0.071176*TX(2)
0305      GO TO 250
0306      234      IF (T=395.0) 235,235,240
0307      235      CF(2)=2500.0
0310      GO TO 250
0311      240      CF(2)=2500.0+20.0*(T=395.0)
0312      250      CONTINUE
0313      C BW(2) IS CONSTANT
0314      C CF(3)
0315      IF (T=344.0) 255,255,260
0316      255      CALL TPOW(T=219.51,2, TX)
0317      CF(3)=2411.076+10.77844*TX(1)+0.04028484*TX(2)
0320      GO TO 275
0321      260      IF (T=395.0) 265,265,270
0322      265      CF(3)=3125.0
0323      GO TO 275
0324      270      CF(3)=3125.0+25.0*(T=395.0)
0325      275      CONTINUE
0326      C BW(3) IS CONSTANT
0327      300      CONTINUE
0330      C NO NEED TO RESET FILTERS IF 350<T<394
0331      IF (T=350.0) 305,302,302
0332      302      IF (T=394.0) 650,305,305
0333      305      DO 308 I=1,3
0334      308      CALL COEFF(CF(I),BW(I),A0(I),A1(I),A2(I),OSR,1)
0335      GO TO 650
0336      C
0337      C
0340      C          SILENCE+RELEASE OF "T"
0341      C
0342      310      IF (T=502.70) 315,320,320
0343      315      AN=0.0
0344      AV=0.0

```

Figure 20. (continued)

```

0345          GO TO 400
0346 320      IF (T=017.77) 325,315,315
0347 C IF FIRST TIME, SET CONSTANT FEATURES OF RELEASE
0350 325      IF (ISW3) 330,330,340
0351 330      ISW3=1
0352          CF(1)=1000.0
0353          CF(2)=1000.0
0354          CF(3)=4300.0
0355          BW(1)=150.0
0356          BW(2)=150.0
0357          BW(3)=2500.0
0360          DO 335 I=1,3
0361 335      CALL COEFF(CF(I),BW(I),A0(I),A1(I),A2(I),OSR,1)
0362 C AN
0363 340      IF (TFRIC) 345,345,400
0364 345      IF (T=502.81) 346,347,347
0365 346      ANDB=111.0
0366          GO TO 395
0367 347      IF (T=512.0) 350,355,355
0370 350      ANDB=94.5
0371          GO TO 395
0372 355      IF (T=538.0) 360,365,365
0373 360      ANDB=89.5
0374          GO TO 395
0375 365      ANDB=89.5+0.40*(T=538.0)
0376 395      AN=10.0***(ANDB/20.0)
0377 400      CONTINUE
0400 C
0401 C
0402 C
0403 C FINAL STEP == RESET TIMERS
0404 C
0405 650      IF (TFRIC) 660,660,670
0406 660      TFRIC=PFIC
0407 670      IF (TVOC) 680,680,690
0410 680      TVOC=PVOC
0411 690      CONTINUE
0412 C*
0413 C*      END OF GPAR
0414 C*
0415 C*****
0416 C
0417 C ADD GLOTTAL WAVE
0420          YN=YN+GLOT(P,AV)
0421 C ADD NOISE
0422 C MULTIPLIED BY RELATIVE NOISE GAIN
0423          YN=YN+GFRIC*AN+FLOAT(IRN4(X))/2047.0
0424 C APPLY LOWER THREE (FORMANT) FILTERS
0425 C AND HIGHER ORDER CORRECTION FILTERS
0426          DO 700 I=1,10
0427 700      YN=RES(YN,YM1(I),YM2(I),A0(I),A1(I),A2(I))
0430 C SIMULATE RADIATION EFFECT
0431          YN=RAD(YN,YM1RAD,OSR)
0432 C MULTIPLY BY OVERALL GAIN FACTOR
0433          YN=YN*GAIN
0434 C CHECK FOR CLIPPING
0435          IF (ABS(YN)=2047.0) 730,730,710

```

Figure 20. (continued)

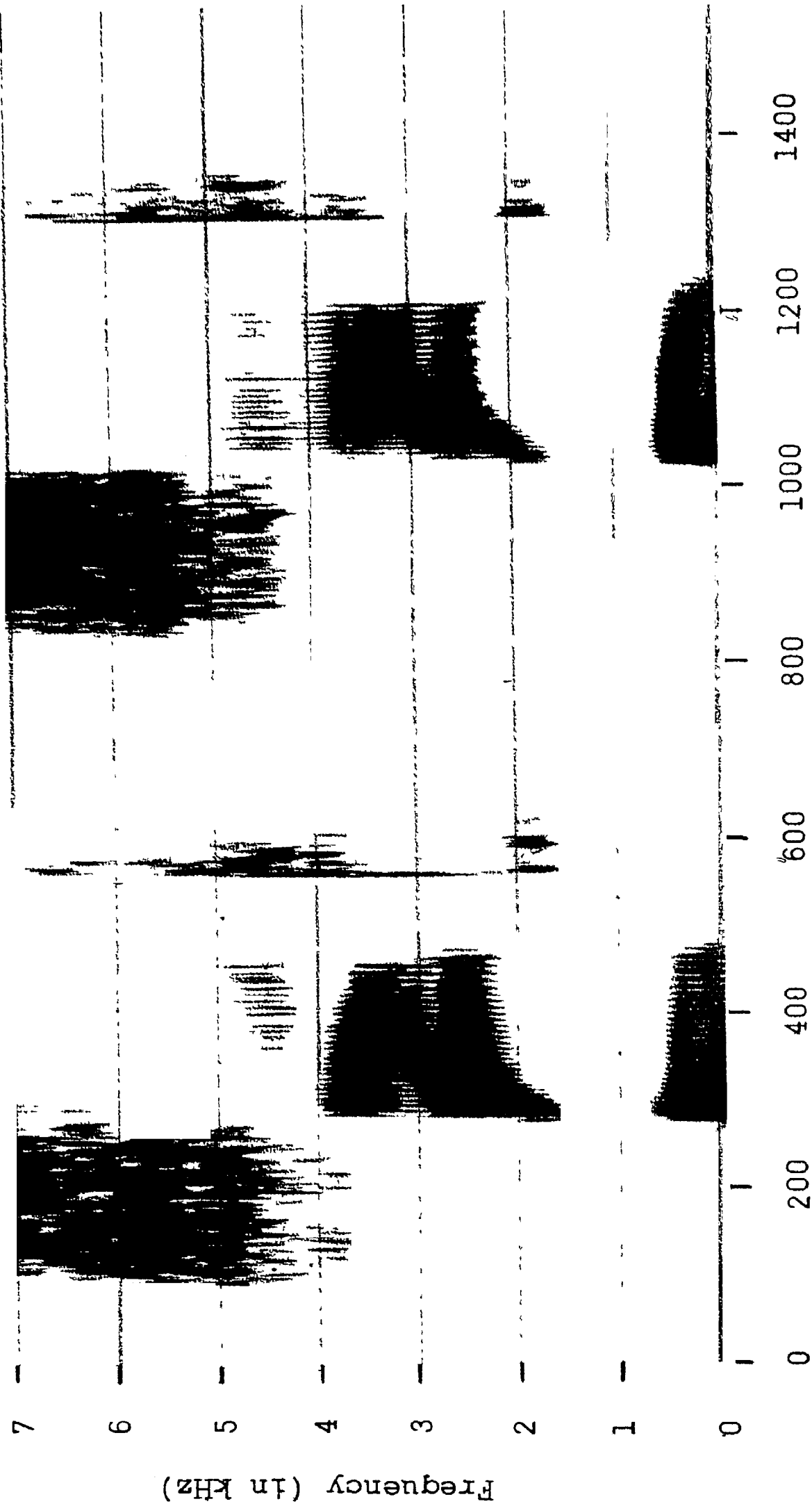
```

0436      C ERROR == SPEECH IS CLIPPED
0437      710      WRITE(1,720)
0440      720      FORMAT(" *** CLIPPED")
0441      GO TO 900
0442      C O.K. == NOT CLIPPED
0443      730      CONTINUE
0444      C STORE SPEECH WAVE POINT
0445      IBUF(IPT)=IFIX(YN)
0446      C INCREMENT GENERAL TIME CLOCK
0447      T=T+DT
0450      750      CONTINUE
0451      C
0452      C      END OF BLOCK-SIZED LOOP
0453      C
0454      C WRITE OUT BLOCK
0455      CALL WRITR(NOUT,IBLK,IBUF,X760)
0456      GO TO 800
0457      C I/O ERROR HANDLING
0460      760      WRITE(1,770)IBLK
0461      770      FORMAT(" *** I/O ERROR BLOCK NO. ",I5)
0462      GO TO 900
0463      800      CONTINUE
0464      C
0465      C      END OF LOOP ON NO. OF BLOCKS
0466      C
0467      C CLOSE WORK FILE
0470      CALL CLOSR(NOUT)
0471      C END
0472      900      WRITE(1,910)
0473      910      FORMAT(" HIT SPACE BAR TO QUIT")
0474      PAUSE
0475      CALL EXIT
0476      END
0477      C
0500      C WSYN1.FD
0501      C
0502      C SYNTHESIZES "SEAT"
0503      C 9/27/74

```

Figure 20. (continued)

8



"Seat" (subject NPE) Time (in msec) "Seat" (Program WSYN2B.FO)

Figure 21. Spectrographic Comparison of Natural " and Synthesized "Seat". This spectrogram was made on the Voice Identification, Inc. Series 700 machine with wide-band setting and high frequency shaping above 1 kHz.

V. Final Remarks

The simple schemes for speech synthesis presented here are sufficient to make speech containing most of the sounds of English.

Figure 22, page 45, is a handy reference containing characteristics of English vowels from several primary sources, assembled in tabular form by Dunn as part of his section of Automatic Speech Recognition (1963). Since this reference is not easily available, the primary sources of Dunn's data are also listed in the bibliography. The fundamental and three formant frequencies for men, women, and children were taken from Peterson and Barney (1952). The three formant amplitudes for each vowel are also from the Peterson and Barney measurements, but averaged over all three classes of speakers. Dunn took the relative strengths of the first formants of the different vowels from the measurements of Sacia and Beck (1926), as quoted by Fletcher (1953) arbitrarily assigning the zero decibel level to the strongest vowel, [ɔ]. The three formant bandwidths are the average of three sets given by Fant (1962), one set his own measurements, one those of House and Stevens, and the third measured by Dunn himself (1961). Dunn cautions that the bandwidths vary widely among individual speakers.

These values of center frequency and bandwidth of formants for the different vowels can be used as variable formant filter control parameters in Model T with unimpeachable results.

Vowel	i	ɪ	ɛ	æ	ɑ	ɔ	u	ʊ	ʌ	ɜ
As in	heed	hid	head	had	hod	hawed	who'd	hood	hud	heard
Fundamental Frequencies (cycles per second)										
M	136	135	130	127	124	129	141	137	130	133
W	235	232	223	210	212	216	231	232	221	218
Ch.	272	269	260	251	256	263	274	276	261	261
Formant Frequencies (cycles per second)										
F ₁ M	270	390	530	660	730	570	300	440	640	490
W	310	430	610	860	850	590	370	470	760	500
Ch.	370	530	690	1010	1030	680	430	560	850	560
F ₂ M	2290	1990	1840	1720	1090	840	870	1020	1190	1350
W	2790	2480	2330	2050	1220	920	950	1160	1400	1640
Ch.	3200	2730	2610	2320	1370	1060	1170	1410	1590	1820
F ₃ M	3010	2550	2480	2410	2440	2410	2240	2240	2390	1690
W	3310	3070	2990	2850	2810	2710	2670	2680	2780	1960
Ch.	3730	3600	3570	3320	3170	3180	3260	3310	3360	2160
Formant Amplitudes (db)										
L ₁	-4	-3	-2	-1	-1	0	-3	-1	-1	-5
L ₂	-24	-23	-17	-12	-5	-7	-19	-12	-10	-15
L ₃	-28	-27	-24	-22	-28	-34	-43	-34	-27	-20
Formant Bandwidths (cycles per second)										
B ₁	54	53	48	63	54	43	50	40	52	44
B ₂	55	69	69	81	57	47	49	44	57	58
B ₃	170	113	101	126	93	68	77	62	89	64

Figure 22. Average Measured Characteristics of Vowels
From Dunn's section (p. D-2) of Automatic Speech Recognition
(1963).

Bibliography

- Automatic Speech Recognition, material for an intensive course for Engineers and Scientists, The University of Michigan Summer Conferences, Ann Arbor, 1963 (2 Vols.)
- Dunn, H.K., "Methods of Measuring Vowel Formant Bandwidths" J. Acoust Soc. Am. 33 (12); 1737-1746 (1961):
- Dunn, H.K., Flanagan, J.L., and Gestrin, P.J., "Complex Zeros of a Triangular Approximation to the Glottal Wave", J. Acoust. Soc. Am. 34, 1977 (A), (1962); reprinted in Automatic Speech Recognition.
- Fant, C.G.M., "On the Predictability of Formant Levels and Spectrum Envelopes from Formant Frequencies", in For Roman Jakobson (The Hague, Mouton, 1956), pp. 109-120.
- Fant, C.G.M., "Speech Analysis and Synthesis", in Air Force Cambridge Research Laboratories Technical Report #62-790, pp. 32-34, January 31, 1962.
- Fant, C.G.M., and Martony, J., "Instrumentation for Parametric Synthesis (OVE II)", STL-QPSR 2, 18-24 (1962).
- Flanagan, J.L., "Note on the Design of 'Terminal-Analog' Speech Synthesizers", J. Acoust. Soc. Am. 29 (2), 306-310 (1957)
- Flanagan, J.L., "Some Properties of the Glottal Sound Source", J. Speech & Hearing Res. 1, 1-18 (1958)
- Flanagan, J.L., and Rabiner, L.R. (eds.), Speech Synthesis; Dowden, Hutchinson & Ross Inc., Stroudsburg, PA (1973)
- Fletcher, H., Speech and Hearing in Communication: D. Van Nostrand, Princeton, N.J. (1953)

- Heinz, J.M., and Stevens, K.N., "On the Properties of Voiceless Fricative Consonants", J. Acoust. Soc. Am. 33 (5), 589-596 (1961)
- Lovell, J.D., Nagel, D.C., and Carterette, E.C., "Digital Filtering and Signal Processing", Behav. Res. Meth. & Instru 5 (1), 21-33 (1973)*.
- Perry, J.L., Schafer, R.W., and Rabiner, L.R., "A Digital Hardware Realization of a Random Number Generator", IEEE Trans. Aud. Elect. AU-20 (4), 236-240 (Oct. 1972)
- Peterson, G.E., and Barney, H.L., "Control Methods Used in a Study of the Vowels", J. Acoust. Soc. Am. 24 (2), 175-184 (1952).
- Rabiner, L.R., "Digital-Formant Synthesizer for Speech-Synthesis Studies", J. Acoust. Soc. Am. 43, 822-828 (1968)
- Rader, C.M., Rabiner, L.R., and Schafer, R.W., "A Fast Method of Generating Digital Random Numbers", Bell Sys. Tech. J. 49, 2303-2310 (Nov. 1970).
- Rosenberg, A.E., "Effect of Glottal Pulse Shape on the Quality of Natural Vowels", J. Acoust. Soc. Am. 49 (2), 583-590 (1971).
- Sacia, C.F., and Beck, C.J., "The Power of Fundamental Speech Sounds", Bell Sys. Tech. J. 5, 393-403 (1926).
- Sekimoto, S., "A Computer-Controlled Speech Synthesizer", Ann. Bull. No. 7, Res. Inst. Logo. & Phon., U. of Tokyo, 39-44 (1973)

An errata sheet for the article by Lovell et al. is available from John D. Lovell, Dept. of Psychology, California State College, Hayward, CA 94542.

END

