Klatt Synthesizer in Simulink®

Sean McLennan
S522 – Digital Signal Processing
Dr. Diane Kewley-Port
April, 2000
Indiana University

1 Introduction

This paper outlines the construction and use of a speech synthesizer based on Klatt (1980) that was implemented in The MathWorks Inc's Simulink® environment which operates in conjunction with MATLAB®. Two aspects of the model are described: the model itself and the manipulation of the model by a function that translates pseudo-phonological coding into time-indexed parameter values to be used by the model in synthesizing particular utterances.

2 The Model

Simulink allows users to create working block-diagrams to create simulations that operate in time. One of its primary intended uses has been for digital signal processing, and it is thus ideal for replicating a Klatt synthesizer. A diagram of the top level of the model appears below in fig. 1 along with the block diagram that appears in Klatt (1980). There are a few major differences between the two; the area outlined in fig. 2 is required only to implement the synthesizer in a completely parallel mode. Since the Simulink model is a hybrid parallel / cascade synthesizer, those parts are not required and have no correlate in fig. 1. Also, in fig. 2, the resonators (R1 - R6) appear both in the cascade section and in the parallel section, although they perform identical functions. In Simulink, since the diagram is equivalent to how the model functions, the two sets of resonators have been collapsed into one set that acts in both cascade and parallel streams simultaneously thereby reducing

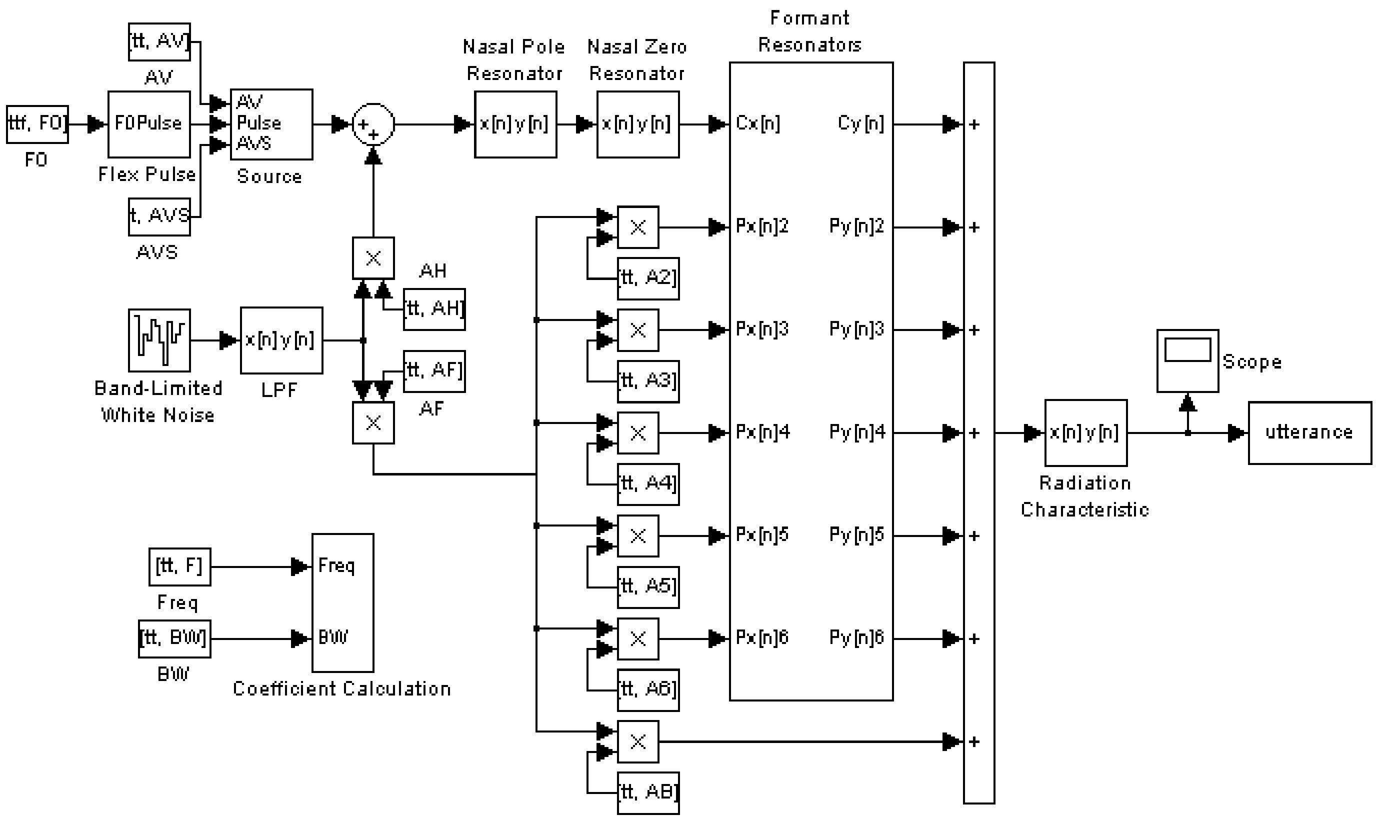


Figure 1: Top Level of Klatt Synthesizer in Simulink

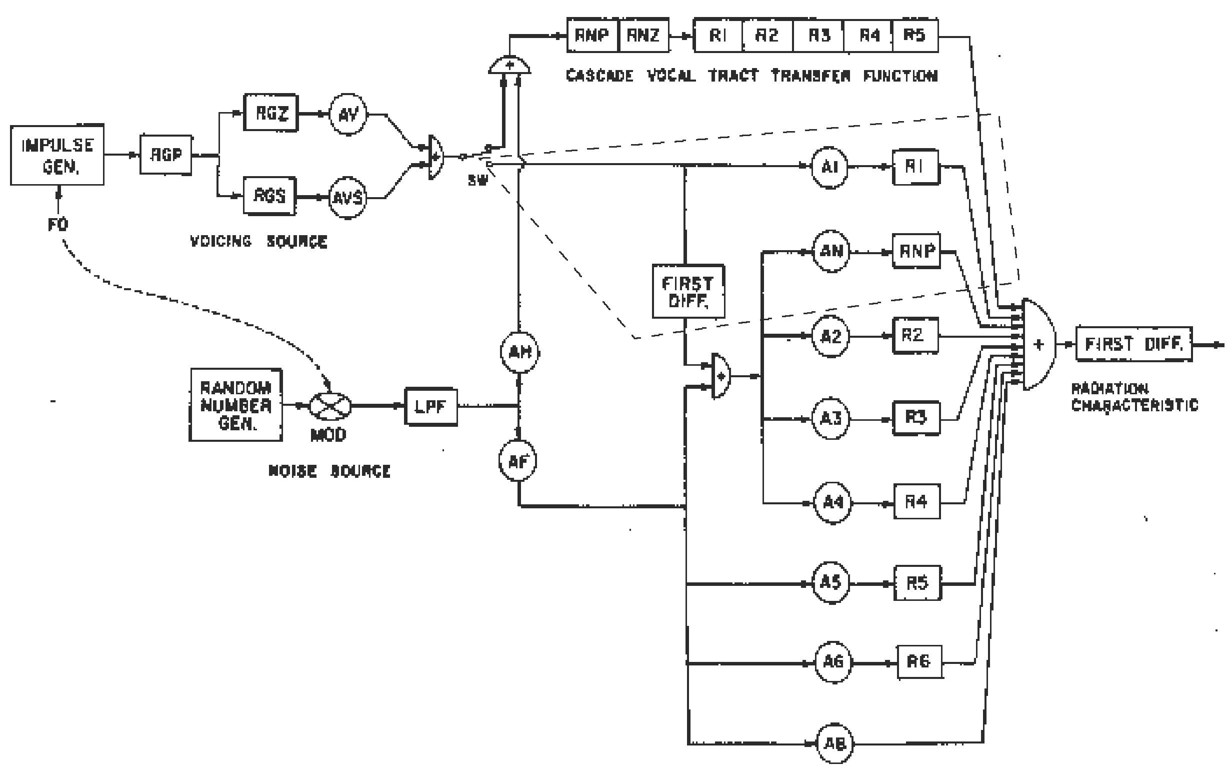
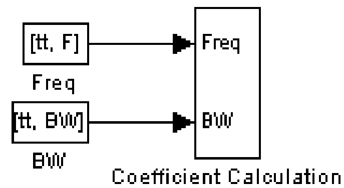


Figure 2: Klatt Synthesizer from Klatt (1980)

the computational load of the simulation. In the bottom left corner of the fig. 1. there is a set of 3 blocks labeled "Coefficient Calculation" – this is where the A, B, and C coefficients for each resonator and anti-resonator are calculated based on frequency and bandwidth as per Klatt.

2.1 Coefficient Calculation



The parameters for the simulation are input from the MATLAB workspace via `From Workspace` blocks such as those that appear on the left side of the diagram at right. In this case, the two variables `F` and `BW` (both matrices) are input along with a time index `tt`. The index and matrix must have an equal number of rows each of which specifies the variable values that are to be used at the corresponding time step. For example:

```
tt = [0; 1000; 2000];
F = [480; 500; 530];
```

The starting value of `F` will be 480, at time step 1000 in the simulation, the value of `F` will be 500, etc. Values at time steps in between those that appear in the index are interpolated by Simulink, and the final value is held until the simulation ends. `F` and `BW` are input into the subsystem that appears in fig. 3. (Subsystems can be accessed by double-clicking on them). Here `F` and `BW` are both matrices with 11 columns corresponding to the frequency and bandwidth values

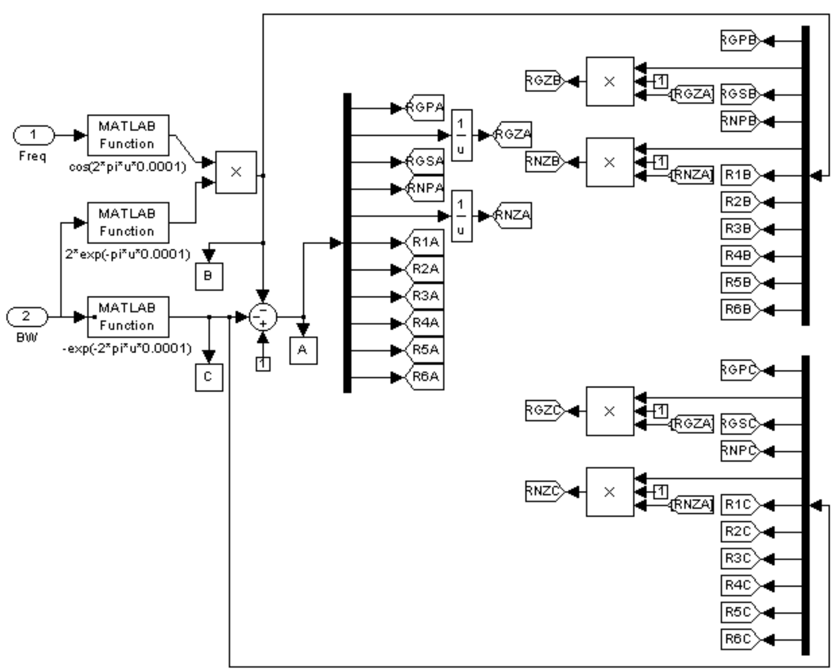
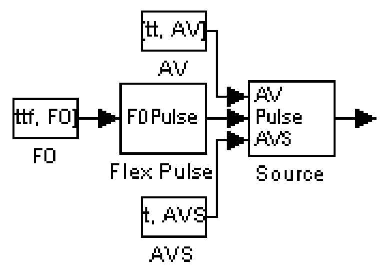


Figure 3: Coefficient Calculation

of each of the 3 source resonators, the 2 nasal resonators and the 6 formant resonators. For efficiency, the calculations are performed as matrix operations and demuxed (i.e. separated into individual column vectors) to goto tags that correspond to the appropriate inputs elsewhere in the model. The boxes labeled A B and C in fig. 3 are variable "sinks" to the workspace. I.e. all the filter coefficients used in the model are stored in workspace variables for later reference.

2.2 Voicing and Frication Source



The voicing source of the synthesizer consists of two primary parts: the pulse generator and the glottal source resonators. In the depiction to the right, they are both subsystems – the pulse receives input from the workspace variable F0 and the source resonators from the pulse generator and the workspace variables AV and AVS which are gain coefficients corresponding to voicing amplitude and sinusoidal voicing amplitude (used for voiced obstruents).

Although Simulink has pulse generator sources as standard library blocks, their frequency is constant which prevents F0 contours from being added to synthesized utterances. Thus, the pulse generator in fig. 4 was created. A full explication of its functioning is not particularly relevant to this discussion; suffice to say, it produces a variable pulse train dependent upon the incoming F0 values. The glottal source resonators that act upon the pulse train are more or less identical to Klatt's – the subsystem is shown in fig. 5. More details about the resonators themselves appear below. The glottal waveform that is

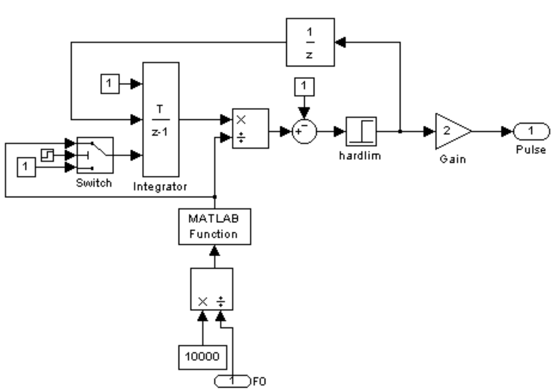


Figure 4: Pulse Generator

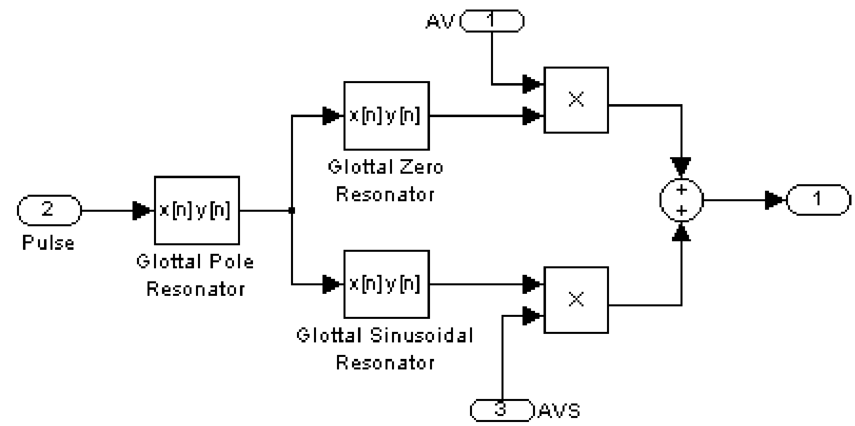


Figure 5: Glottal Source Filters

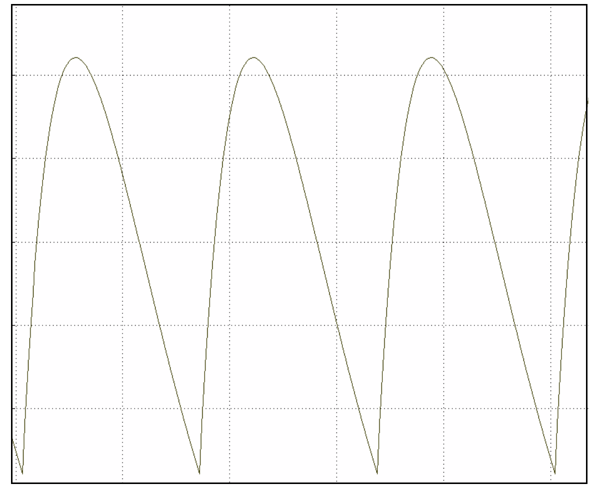


Figure 6: Glottal Waveform

produced by a 120Hz pulse appears in fig. 6.

The frication noise source is a standard Simulink block: `Band Limited White Noise` which is passed through a lowpass filter (labeled `LPF` in fig. 1) as per Klatt. Noise can be passed through either the cascade or parallel streams of the synthesizer, depending upon the gain values AH (aspiration amplitude) and AF (frication amplitude) that appear in fig. 1.

2.3 Resonators

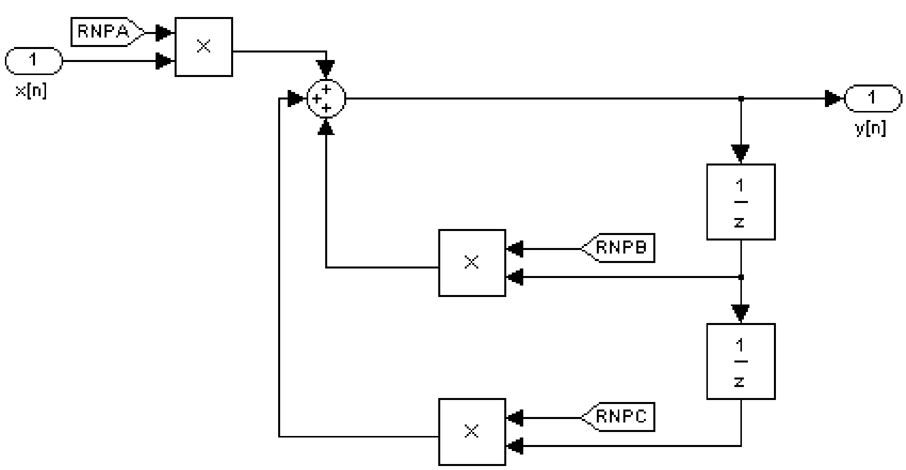
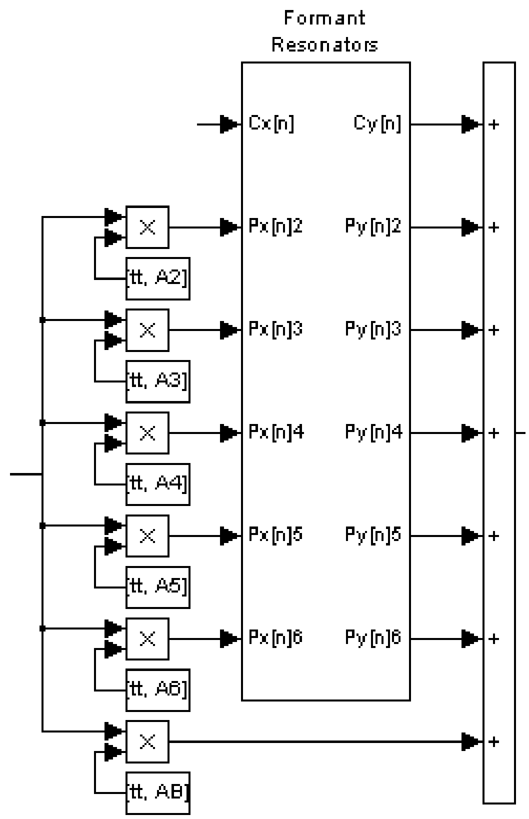


Figure 7: Resonator

All the resonators in the model are the same: second order IIR filters and are instantiated as subsystems. Fig. 7 shows how the filters are implemented; the coefficients are brought from the `Coefficient Calculation` subsystem via `From Tags`.

2.4 Parallel and Cascade Streams



As was mentioned previously, the parallel and cascade streams of the model are processed simultaneously by the same resonators for computational efficiency. In the excerpt at right, the input/output labeled `Cx[n]/Cy[n]` indicate the cascade stream, and the `Px[n]`*m*/`Py[n]`*m* inputs / outputs correspond to the appropriate parallel streams. The parallel inputs are adjusted by the appropriate gain terms (A2-A6, and AB) as in Klatt, and all outputs, cascade and parallel, are summed. The formant resonators are within a subsystem that is shown in fig. 8.

The simultaneous processing of both streams is not as difficult to implement nor as mysterious as it may sound. The cascade input and the parallel input for the first resonator are `mux`ed together (the column vectors are concatenated together into a 2 column matrix), the resonator filters each column independently, and the two columns are `demux`ed. The parallel signal is output from the subsystem and the cascade signal is `mux`ed with the next parallel signal to be passed through the next resonator.

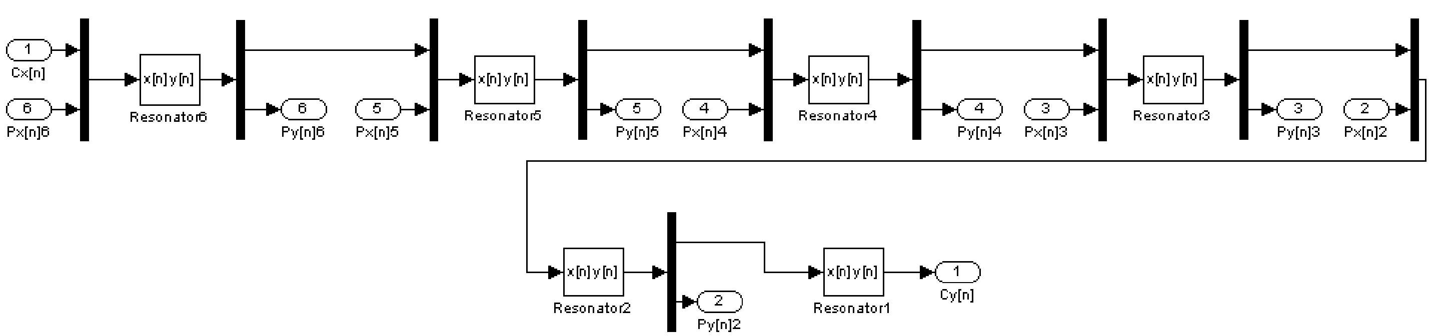
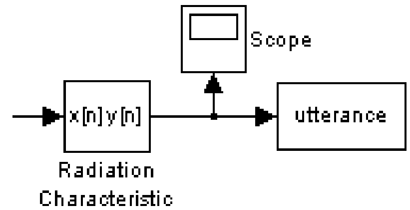


Figure 8: Formant Resonantors

2.5 Final Output



Finally, the entire signal is passed through a first difference filter to simulate the radiation characteristic (as in Klatt) and the output waveform is sent to the Workspace as the variable `utterance`.

The output can also be viewed graphically by opening the `scope`. From the workspace, the final waveform can be played using the command:

```
wavplay(utterance, 10000);
```

3 Manipulating Parameters

The following list of variables must be present in the workspace to for the simulation to run:

Variable Name	Size	Comment
`tt`	*m* x 1	time index - must be monotonically increasing integers
`ttf`	*n* x 1	time index for `F0` (again monotonically increasing integers)
`F0`	*n* x 1	fundamental frequency values
`F` `B`	*m* x 11	resonator frequency and bandwidth values in the order: **`RGP RGZ RGS RNP RNZ R1 R2 R3 R4 R5 R6`** where G=glottal; N=nasal; P=pole; Z=zero; and S=sinusoidal

`AV`
`AVS`
`AH`
`AF`
`A2-A6`
`AB`
	m x 1	gain term for voicing amplitude
		gain term for sinusoidal voicing amplitude
		gain term for noise to cascade resonators
		gain term for noise to parallel resonators
		gain terms for individual parallel formant resonators
		gain term for noise that bypasses formant resonators

3.1 `makeutt.m`

Of course maximum control (and the most successful results) is achieved by manipulating these parameters by hand, coding them in an m-file. One is only limited by their patience: the nature of the model allows control of parameters at each time step, i.e. every 0.1 ms since the model (like Klatt) uses a 10,000 Hz sample period. However, it is much more efficient to automate the task of assembling the variables above. With that in mind, a function, **`makeutt.m`** was created that hacks together default values in a rudimentary way. It is by no means an accurate depiction of English speech – the focus was more on simplicity and code-efficiency – however the output is certainly understandable and perhaps could act as a starting point for more a more fine-grained synthesis. It does take into account in basic ways: variation in vowel length, vowel nasalization, formant transitions from pre/post-vocalic stops, aspiration, unreleased stops, and it adds an basic rising to falling F0 contour.

IPA	ASCII	IPA	ASCII	IPA	ASCII	IPA	ASCII
/p/	p	/b/	b	/i/	i	/ɪ/	I
/t/	t	/d/	d	/e/	e	/ɛ/	E
/k/	k	/g/	g	/æ/	A	/a/	a
/tʃ/	c	/dʒ/	j	/ə/	*	/ʌ/	^
/f/	f	/v/	v	/aj/	@	/aw/	&
/θ/	T	/ð/	D	/u/	u	/ʊ/	U
/s/	s	/z/	z	/o/	o	/ɔ/	>
/ʃ/	S	/ʒ/	Z	/ɔj/	!		
/h/	h	/^{h}/	"				
/m/	m	/w/	w				
/n/	n	/j/	y				
/ŋ/	N						
/l/	l	/ɹ/	r				

Table 1: ASCII Phonemes for input to `makeutt.m`

makeutt.m takes a string of characters (pseudo-phonemic symbols) as input and produces the variables above. Thus the command line call to **makeutt.m** would be:

```
[F0, ttf, A2, A3, A4, A5, A6, AB, AV, AH, AF, AVS, tt, F, BW] = makeutt('text');
```

The ASCII "phonemes" that encode the utterance appear in Table 1.

Additionally, a "-" may be entered following stops to suppress their release. (Releases are automatically repressed utterance-finally.) The function will display the total simulation time that needs to be set under the simulation parameters in Simulink.

The actual code for **makeutt.m** follows at the end of this document.

4 An Example

The following call was made to **makeutt.m** to synthesize "Sean has a cat" (wav-file included):

```
[F0, ttf, A2, A3, A4, A5, A6, AB, AV, AH, AF, AVS, tt, F, BW] =
            makeutt('SanhAz*k"At');
```

The waveform produced is shown in fig. 9 along with a spectrogram in fig. 10.

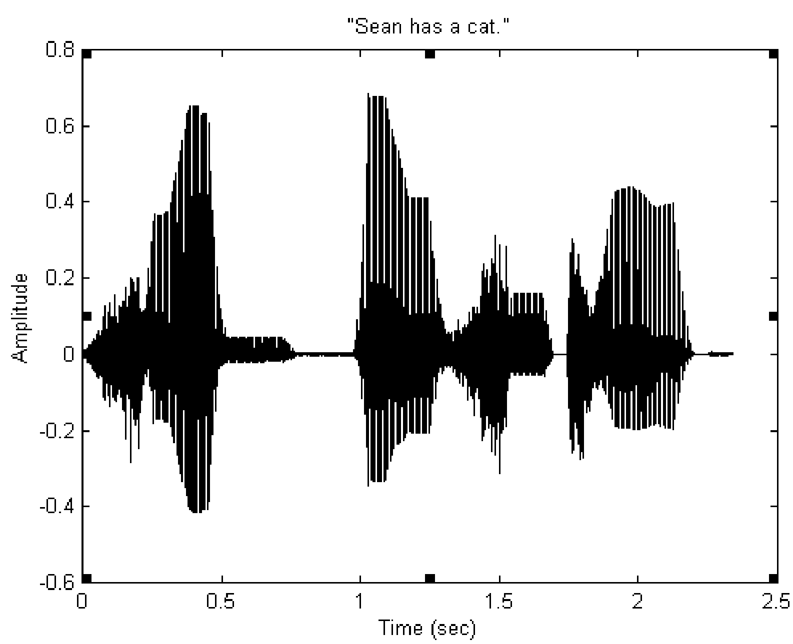


Figure 9: Waveform of "Sean has a cat."

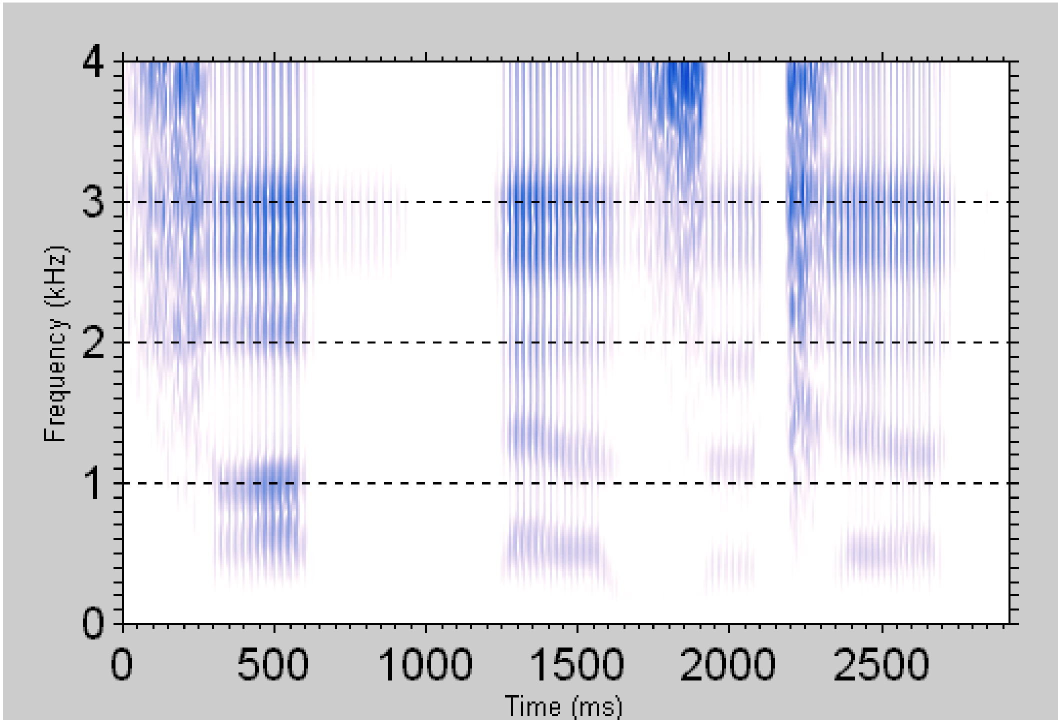


Figure 10: Spectrogram of "Sean has a cat."

5 Conclusion

Although there are now more sophisticated techniques for synthesizing speech, Klatt's (1980) framework remains one of the most simple and effect methods. Simulink in particular provides an excellent environment for modeling and manipulating such systems. As Simulink also has an add-on feature called "Real-Time Workshop" that ports models to C-code, there is enormous potential in using it to build efficient, perhaps real-time, applications in this domain.

```
function [F0, ttf, A2, A3, A4, A5, A6, AB, AV, AH, AF, AVS, tt, F, BW] = makeutt(utt)

% FGP  FGZ  FGS  FNP  FNZ  F1   F2   F3   F4   F5   F6 BGP  BGZ BGS BNP BNZ BW1  BW2 BW3  BW4  BW5  BW6 A2 A3 A4 A5 A6 AB AH AF AV AVS Weight
segments=[
    0 1500    0  250  250 290  610 2150 3300 3750 4900 100 6000 100 100 100  50   80  60  250  200 1000  0  0  0  0  0  0  0  0 50   0 1 % /w/
    0 1500    0  250  250 260 2070 3020 3300 3750 4900 100 6000 100 100 100  40  250 500  250  200 1000  0  0  0  0  0  0  0  0 50   0 1 % /y/
    0 1500    0  250  250 310 1060 1380 3300 3750 4900 100 6000 100 100 100  70  100 120  250  200 1000  0  0  0  0  0  0  0  0 50   0 1 % /r/
    0 1500    0  250  250 310 1050 2880 3300 3750 4900 100 6000 100 100 100  50  100 280  250  200 1000  0  0  0  0  0  0  0  0 50   0 1 % /l/
    0 1500    0  270  450 480 1270 2130 3300 3750 4900 100 6000 100 100 100  40  200 200  250  200 1000  0  0  0  0  0  0  0  0 40  50 1 % /m/
    0 1500    0  270  450 480 1340 2470 3300 3750 4900 100 6000 100 100 100  40  300 300  250  200 1000  0  0  0  0  0  0  0  0 40  50 1 % /n/
    0 1500    0  270  450 480 2000 2900 3300 3750 4900 100 6000 100 100 100  40  300 300  250  200 1000  0  0  0  0  0  0  0  0 40  50 1 % /N/
    0 1500    0  250  250 340 1100 2080 3300 3750 4900 100 6000 100 100 100 200  120 150  250  200 1000  0  0  0  0  0 57  0 20  0   0 1 % /f/
    0 1500    0  250  250 220 1100 2080 3300 3750 4900 100 6000 100 100 100  60   90 120  250  200 1000  0  0  0  0  0 57  0 20 47  47 1 % /v/
    0 1500    0  250  250 320 1290 2540 3300 3750 4900 100 6000 100 100 100 200   90 200  250  200 1000  0  0  0  0 28 38  0 20  0   0 1 % /T/
    0 1500    0  250  250 270 1290 2540 3300 3750 4900 100 6000 100 100 100  60   80 170  250  200 1000  0  0  0  0 28 38  0 20 47  47 1 % /D/
    0 1500    0  250  250 320 1390 2530 3300 3750 4900 100 6000 100 100 100 200   80 200  250  200 1000  0  0  0  0 52  0  0 20  0   0 1 % /s/
    0 1500    0  250  250 240 1390 2530 3300 3750 4900 100 6000 100 100 100  70   60 180  250  200 1000  0  0  0  0 52  0  0 20 47  47 1 % /z/
    0 1500    0  250  250 300 1840 2750 3300 3750 4900 100 6000 100 100 100 200  100 300  250  200 1000  0 28 24 24 23  0  0 20  0   0 1 % /S/
    0 1500    0  250  250 220 1840 2750 3300 3750 4900 100 6000 100 100 100  70   60 280  250  200 1000  0 28 24 24 23  0  0 20 47  47 1 % /Z/
    0 1500    0  250  250 350 1800 2820 3300 3750 4900 100 6000 100 100 100 200   90 300  250  200 1000  0 22 30 26 26  0  0 10  0   0 1 % /c/
    0 1500    0  250  250 260 1800 2820 3300 3750 4900 100 6000 100 100 100  60   80 270  250  200 1000  0 22 30 26 26  0  0 10 37  37 1 % /j/
    0 1500    0  250  250 200 1100 2150 3300 3750 4900 100 6000 100 100 100  60  110 130  250  200 1000  0  0  0  0  0 63  0 10 20  20 1 % /b/
    0 1500    0  250  250 200 1600 2600 3300 3750 4900 100 6000 100 100 100  60  100 170  250  200 1000  0 23 30 31 30  0  0 10 20  20 1 % /d/
    0 1500    0  250  250 200 1990 2850 3300 3750 4900 100 6000 100 100 100  60  150 280  250  200 1000 30 27 22 23 23  0  0 10 20  20 1 % /g/
    0 1500    0  250  250 400 1100 2150 3300 3750 4900 100 6000 100 100 100 300  150 220  250  200 1000  0  0  0  0  0 63  0 20  0   0 1 % /p/
    0 1500    0  250  250 400 1600 2600 3300 3750 4900 100 6000 100 100 100 300  120 250  250  200 1000  0 15 23 28 32  0  0 20  0   0 1 % /t/
    0 1500    0  250  250 300 1990 2850 3300 3750 4900 100 6000 100 100 100 250  160 330  250  200 1000 30 26 22 23 23  0  0 20  0   0 1 % /k/
    0 1500    0  250  250 310 2020 2960 3300 3750 4900 100 6000 100 100 100  45  200 400  250  200 1000  0  0  0  0  0  0  0  0 60   0 1 % /i/
    0 1500    0  250  250 290 2070 2960 3300 3750 4900 100 6000 100 100 100  60  200 400  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 400 1800 2570 3300 3750 4900 100 6000 100 100 100  50  100 140  250  200 1000  0  0  0  0  0  0  0  0 60   0 .8% /I/
    0 1500    0  250  250 470 1600 2600 3300 3750 4900 100 6000 100 100 100  50  100 140  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 480 1720 2520 3300 3750 4900 100 6000 100 100 100  70  100 200  250  200 1000  0  0  0  0  0  0  0  0 60   0 1 % /e/
    0 1500    0  250  250 330 2020 2600 3300 3750 4900 100 6000 100 100 100  55  100 200  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 530 1680 2500 3300 3750 4900 100 6000 100 100 100  60   90 200  250  200 1000  0  0  0  0  0  0  0  0 60   0 .8% /E/
    0 1500    0  250  250 620 1530 2530 3300 3750 4900 100 6000 100 100 100  60   90 200  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 620 1660 2430 3300 3750 4900 100 6000 100 100 100  70  150 320  250  200 1000  0  0  0  0  0  0  0  0 60   0 1.2% /A/
    0 1500    0  250  250 650 1490 2470 3300 3750 4900 100 6000 100 100 100  70  100 320  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 700 1220 2600 3300 3750 4900 100 6000 100 100 100 130   70 160  250  200 1000  0  0  0  0  0  0  0  0 60   0 1.1% /a/
    0 1500    0  250  250 700 1220 2600 3300 3750 4900 100 6000 100 100 100 130   70 160  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 600  990 2570 3300 3750 4900 100 6000 100 100 100  90  100  80  250  200 1000  0  0  0  0  0  0  0  0 60   0 1 % />/
    0 1500    0  250  250 630 1040 2600 3300 3750 4900 100 6000 100 100 100  90  100  80  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 620 1220 2550 3300 3750 4900 100 6000 100 100 100  80   50 140  250  200 1000  0  0  0  0  0  0  0  0 60   0 .7% /^/
    0 1500    0  250  250 620 1220 2550 3300 3750 4900 100 6000 100 100 100  80   50 140  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 540 1100 2300 3300 3750 4900 100 6000 100 100 100  80   70  70  250  200 1000  0  0  0  0  0  0  0  0 60   0 1% /o/
    0 1500    0  250  250 450  900 2300 3300 3750 4900 100 6000 100 100 100  80   70  70  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 450 1100 2350 3300 3750 4900 100 6000 100 100 100  80  100  80  250  200 1000  0  0  0  0  0  0  0  0 60   0 .8% /U/
    0 1500    0  250  250 500 1180 2390 3300 3750 4900 100 6000 100 100 100  80  100  80  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 350 1250 2200 3300 3750 4900 100 6000 100 100 100  65  110 140  250  200 1000  0  0  0  0  0  0  0  0 60   0 1% /u/
    0 1500    0  250  250 320  900 2200 3300 3750 4900 100 6000 100 100 100  65  110 140  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 500 1400 2300 3300 3750 4900 100 6000 100 100 100 100   60 110  250  200 1000  0  0  0  0  0  0  0  0 50   0 .6% /*/
    0 1500    0  250  250 500 1400 2300 3300 3750 4900 100 6000 100 100 100 100   60 110  250  200 1000  0  0  0  0  0  0  0  0 50   0 1
    0 1500    0  250  250 660 1200 2550 3300 3750 4900 100 6000 100 100 100 100   70 200  250  200 1000  0  0  0  0  0  0  0  0 60   0 1.2 % /@/
    0 1500    0  250  250 400 1880 2500 3300 3750 4900 100 6000 100 100 100  70  100 200  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 640 1230 2550 3300 3750 4900 100 6000 100 100 100  80   70 140  250  200 1000  0  0  0  0  0  0  0  0 60   0 1.2 % /&/
    0 1500    0  250  250 420  940 2350 3300 3750 4900 100 6000 100 100 100  80   70  80  250  200 1000  0  0  0  0  0  0  0  0 60   0 1
    0 1500    0  250  250 550  960 2400 3300 3750 4900 100 6000 100 100 100  80   50 130  250  200 1000  0  0  0  0  0  0  0  0 60   0 1.2 % /!/
    0 1500    0  250  250 360 1820 2450 3300 3750 4900 100 6000 100 100 100  60   50 160  250  200 1000  0  0  0  0  0  0  0  0 60   0 1];
```

```
%intitiate the matrices and variables
tt = [];
ttf = [];
vals = [];
nextt = 0;
nas = 0;
offset = 0;
stop = 0;
vowels = 0;

for i=1:1:length(utt)
  id = getid(utt(i));
  if id >= 24 & id < 53

     tw = segments(id, 33);  % adjust length of vowel according to parameters in col 33 above.
     vowels = vowels + 1;
     if i == 1
        tt(length(tt)+1) = 0;
     else
        tt(length(tt)+1) = round(tt(length(tt))+(nextt*tw));
     end;

     if i ~= length(utt)  % determine if nasalization requited at end of vowel before nasal
        if getid(utt(i+1)) >= 5 & getid(utt(i+1)) <=7
           nas = 1;
           offset = 2;
        end;
     end;

     tt(length(tt)+1) = round(tt(length(tt))+(500*tw));
     tt(length(tt)+1) = round(tt(length(tt))+(700*tw));
     tt(length(tt)+1) = round(tt(length(tt))+(600*tw));

     [y x]=size(vals);

     vals(y+1:y+4, :) = [segments(id, :); segments(id, :); segments(id+1, :); segments(id+1, :)];

     if nas == 1  % adjust values for nasalization
        vals(y+1+offset:y+2+offset, 6) = vals(y+1+offset:y+2+offset, 6)+100;
        vals(y+1+offset:y+2+offset, 4:5) = [(vals(y+1+offset:y+2+offset, 6)+270)./2 [450;450]];
```

```
      nas = 0;  offset = 0;
   end;

   nextt=round(600*tw);


elseif id <=23 %
   if i == 1
      tt(length(tt)+1) = 0;
   else
      tt(length(tt)+1) = tt(length(tt))+nextt;
   end

   if id <=15  time1 = 2000; time2 = 500; fric = 0;  % values for fricatives, and sonorant consonants
   elseif id >= 21  time1 = 100; time2 = 600; fric = 30; stop = 1;  % values for voiceless stops
         tt(length(tt)+1) = tt(length(tt))+500;
         [y x] = size(vals);
         vals(y+1:y+2,:) = [segments(id, :); segments(id, :)];
         vals(y+1:y+2, 30) = [0; 0];
         tt(length(tt)+1) = tt(length(tt))+200;
   elseif id == 16 | id == 17  time1 = 1000; time2 = 500; fric = 10; % affricates
         tt(length(tt)+1) = tt(length(tt))+500;
         [y x] = size(vals);
         vals(y+1:y+2,:) = [segments(id, :); segments(id, :)];
         vals(y+1:y+2, 30:31) = [0, 0; 0, 0;];
         tt(length(tt)+1) = tt(length(tt))+200;
   else  time1 = 100; time2 = 600; fric = 10; stop = 1;% values for voiced stops
   end

   tt(length(tt)+1) = tt(length(tt))+time1;

   [y x] = size(vals);
   vals(y+1:y+2, :) = [segments(id, :); segments(id, :)];

   if stop == 1
      if i == length(utt)                    %suppress release of final stops and when desired with '-'
         vals(y+1:y+2, 29:30) = [50 0; 20 0];
      elseif i ~= length(utt) & utt(i+1) == '-'
         vals(y+1:y+2, 29:30) = [50 0; 20 0];
      else
```

```
            vals(y+1, 30) = fric;
         end;
      else
         vals(y+1, 30) = fric;
      end;

      if vals(y+1, 5) > 250
         nas = 1;
      end;

      nextt = time2;

      stop = 0;

   elseif id == 53 %h
      if i == 1
         tt(length(tt)+1) = 0;
      else
         tt(length(tt)+1) = tt(length(tt))+nextt;
      end

      tt(length(tt)+1) = tt(length(tt))+2000;

      [y x] = size(vals);

      vals(y+1:y+2, :) = [segments(getid(utt(i+1)), :); segments(getid(utt(i+1)), :)];
      vals(y+1:y+2, 6) = [300; 300];
      vals(y+1:y+2, 29) = [50; 50];
      vals(y+1:y+2, 31) = [0; 0];

      nextt = 500;

   elseif id == 54 %aspiration
      tt(length(tt)) = tt(length(tt))+300;
      nextt = nextt+300;
   end;
end;

totalt = tt(length(tt))+nextt;
```

```
display(['Set Simulation time to ' num2str(totalt)]);

tt = tt';
vals(:, 33) = [];
A2 = vals(:, 23);
A3 = vals(:, 24);
A4 = vals(:, 25);
A5 = vals(:, 26);
A6 = vals(:, 27);
AB = vals(:, 28);
AH = vals(:, 29);
AF = vals(:, 30);
AV = vals(:, 31);
AVS = vals(:, 32);

F = vals(:, 1:11);
BW = vals(:, 12:22);

if vowels == 1;
   ttf = [0; round(totalt/2); totalt];
   F0 = [120; 130; 140];
elseif vowels >= 2
   ttf = [0; round(totalt/(vowels+1)); 2*round(totalt/(vowels+1)); vowels*round(totalt/(vowels+1)); totalt];
   F0 = [120; 140; 130; 130; 100];
end;


function out = getid(tofind)  % function that gets numerical position of ASCII phonemes

index = 'wyrlmnNfvTDszSZcjbdgptki I e E A a > ^ o U u * @ & !h"-';
%        1   5    0    5    0
out = find(index==tofind);
```

References:

Klatt, D. (1980). Software for a cascade / parallel formant synthesizer. *Journal of the Acoustical Society of America 67(3).* 971-995