Constructing a Time-Varying Multi-Vowel Synthesizer in MATLAB®

Paul J. Coyne, Jr.
Loyola College in Maryland
Department of Electrical Engineering and Engineering Science

Abstract

A digital filter simulation of the vocal cavity, positioned to create a vowel, that was excited by a glottal pulse train with the filter output played through a sound card can be accomplished using MATLAB® with a modest effort. The simulation of a time-varying vocal tract that generated sequential vowels, with a natural transition, required the modification of the digital filter coefficients as the glottal pulse train excited the vocal tract model. This paper presents a procedure for the implementation of a pseudo time-varying digital filter that simulated the vocal tract producing sequential vowels. The procedure allowed the controlled movement of the poles associated with one vowel into the correct position for the second vowel. The pole locations of the vowels were represented in polar form using the formant frequencies, scaled by the folding frequency. The radial locations of the poles were arrived at by matching the spectral analysis of the digital filter simulation with the actual vowel spectrum. Since MATLAB® does not allow time-varying digital filters, the time scale used for the incremental changes in the digital filter coefficients was chosen to be the glottal pulse period, which was set for a male speaker. The transition interval from one vowel to the next was selected so the listener would sense a natural change. Simulation results were generated for a /u/ - /i/ vowel combination. The M-files necessary for the simulation were included as an Appendix.

Introduction

This project is a candidate for inclusion in a course applying Digital Signal Processing techniques at the senior or first year graduate level. This project is an extension to a project in Burrus; et al's book entitled Computer-Based Exercises for Signal Processing Using MATLAB®[1]. In the Applications chapter, a Vowel Synthesis project is included in the Speech Modeling section. A discrete-time system that models the glottal pulse generation by the vocal cords or folds is driven by a periodic train of impulse in MATLAB® with a period which can be set to mimic a man, a woman or a child. This glottal pulse train is then passed through a discrete-time all-pole model of the vocal tract. This model has its origin in a concatenation of lossless tubes as a model of the vocal tract[2]. In this project an eighth-order, all-pole, discrete-time filter is used. The vowel synthesis system is completed by passing the output of the vocal tract model through a model of the radiation of sound passing the lips[3]. This is accomplished using a highpass FIR filter. The final output vector in MATLAB® is then played through a PC sound card using the built-in MATLAB® function sound.

The project I will describe has its origin in a student question. "How can you make MATLAB® produce a natural sequence of vowel sounds?" An obvious answer would involve the generation of separate vowel vectors, which would be concatenated and finally played through a sound card. This has the drawback of producing an abrupt change in sound mimicking a discontinuous change in the vocal tract structure. The goal of this project is to allow MATLAB® to generate multiple vowel sounds by changing the vocal tract digital filter coefficient in small steps thus allowing a natural transition between vowel sounds.

Simulation

Vowel sounds are periodic and an analysis of the frequency spectrum will show a few dominant peaks which are referred to as formant frequencies. Formant frequency locations for vowels are affected by three factors: the overall length of the vocal tract cavity, the locations of constrictions along tract, and the narrowness of the constrictions[4]. The dominant formant frequencies for a male speaker are located in the range 0 Hz to 4 kHz. Thus a sampling frequency of 10kHz is assumed for the input vector to the sound card. The discrete-time model for the generation of a sequence of vowels is shown in Figure 1. Given a 0.1 msec sampling interval, the pitch period or the time between glottal pulse entering the vocal cavity is set for a male speaker at 12 msec, which corresponds to a fundamental frequency of phonation of 83.33Hz. This gives a p(t) function, which has a unit impulse, every 120 samples with zeroes in between the impulses.

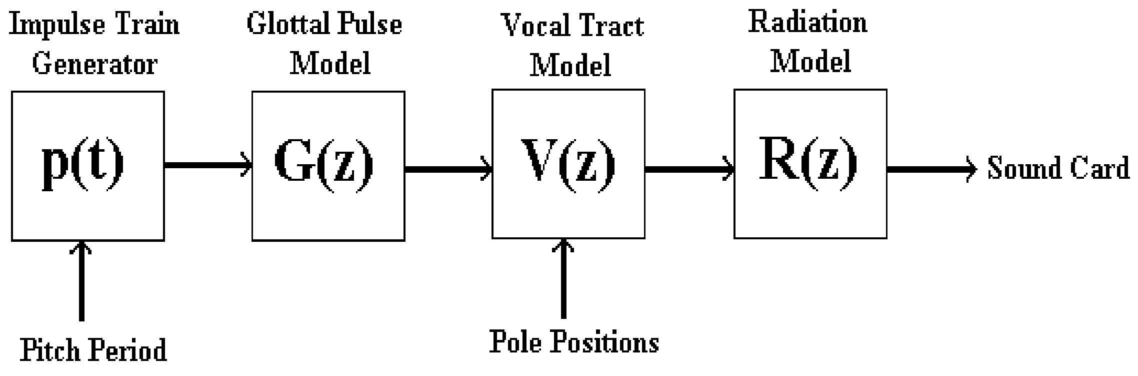

Figure 1: Discrete-Time Model For Vowel Production

The first project in the Speech Modeling section deals with glottal pulse models. This simulation employs the Exponential Model. The transfer function is given by

$$G(z) = -a\ e\ \ln(a)\ z^{-1}\ (\ 1 - a\ z^{-1})^{-2} \qquad (1),$$

where e = 2.71828… is the natural logarithm base and a is the parameter which adjusts the real double-pole location. In this project a is set equal to 0.88. An image of a single glottal pulse, with a = 0.88, is shown in Figure 2. There is no need to replicate the pulse many times to allow a multi-second speech segment, since the vocal tract digital filter coefficients is adjusted after each period of the glottal pulse.

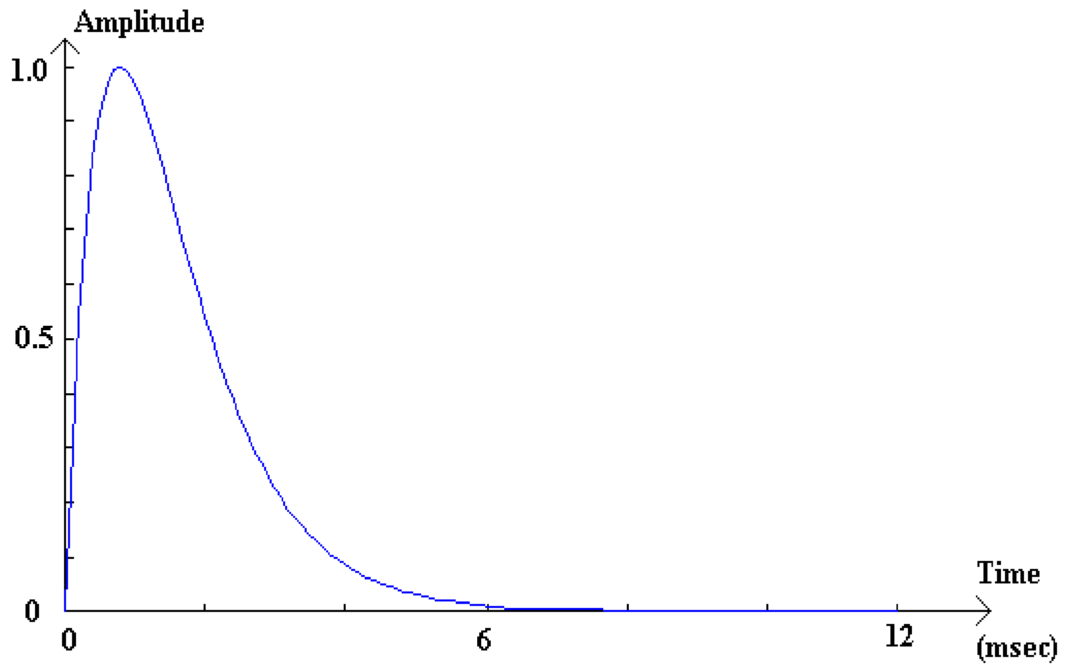

Figure 2: Exponential Model Glottal Pulse

The information needed to construct the vocal tract model consists of the locations of the four largest peaks in the spectrum of a segment of the vowel sound. This allows the angular location of each of the four complex conjugate pole-pairs poles to be specified. A radial location for each pole-pair is arrived at by matching the spectrum of the simulation with available spectrums of actual vowel sounds. Given the pole locations, the folding frequency, and a z-plane scale factor the MATLAB® function zp2tf is used to create a set of filter coefficients to model the vocal tract. An example vocal tract frequency response for the vowel /u/ which appears in the word “boot” is shown in Figure 3.

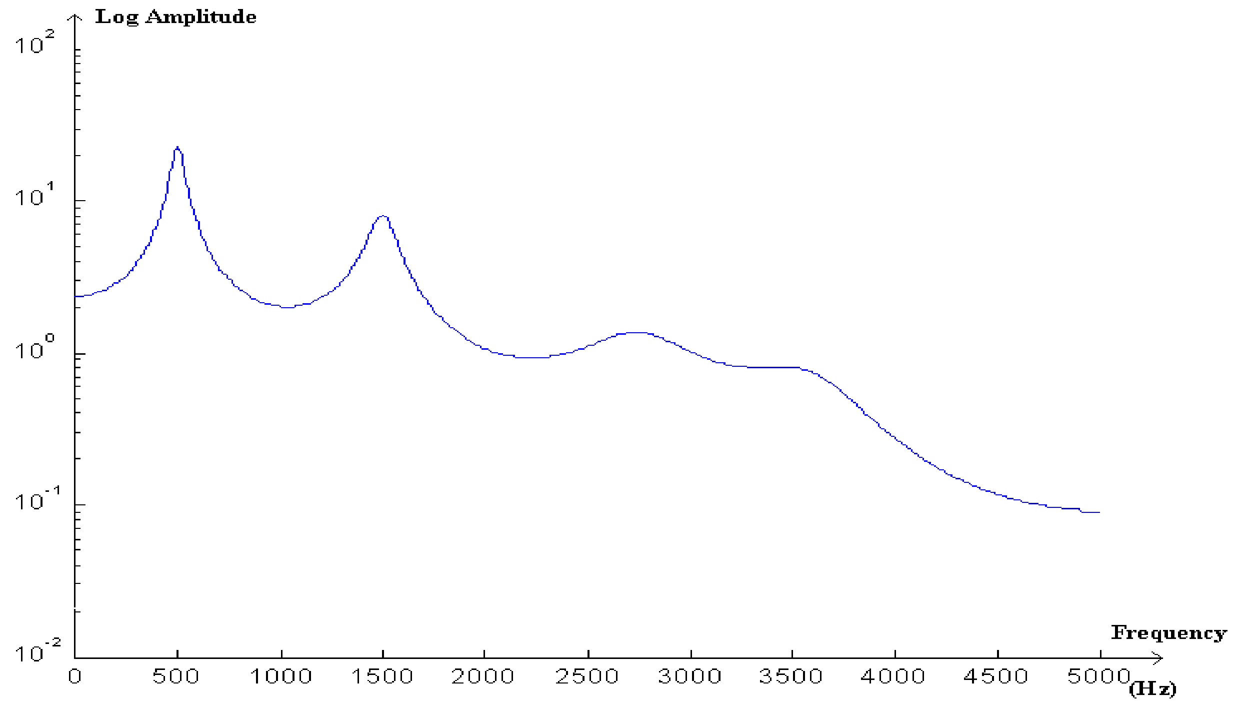

Figure 3: Magnitude Spectrum of Vowel /u/.

The adjustment of the vocal tract parameters, which create the time-varying effect, was accomplished using a first-order discrete-time system whose initial state corresponds to the initial pole angle or pole radius and whose final state corresponds to the final location. Equation (2) illustrates the system using the generic variable p for position.

$$p = p_f + (p_i - p_f) * b^{(120 * m)} \qquad (2)$$

The variable p_f was the final position, p_i was the initial position, b allowed the rate of movement to be adjusted, and 120*m corresponded to the increment in discrete-time between adjustments, with m counting the glottal pulse periods. The pole positions were adjusted after every glottal pulse cycle, which is 12 msec or 120 units of discrete-time given the sampling interval. The parameter b was chosen so that the vowel movement would take place over about 60 msec or by the fifth glottal pulse cycle the poles would be at the new location. This corresponds to a value or b = 0.994.

The final phase of the simulation passed the complete vocal tract output through a model of the radiation of sound passing the lips. The model suggested in the Vowel Synthesis project is given by Equation (3).

$$R(z) = 1 - z^{-1} \qquad (3)$$

The radiation model corresponds to a discrete-time system with a zero at z = 1 or a high-pass filter.

The simulation initialization, the cycling through each glottal pulse and the sound generation procedures are outlined below.

1. Initialize:

    (a) Create 120 msec pitch period impulse sequence, p(t).

    (b) Create glottal pulse, gp88, by passing p(t) through G(z), with a=0.88.

    (c) Create initial vocal tract transfer function, V(z), using the initial vowel pole positions.

    (d) Pass gp88 through V(z), creating output, y, and the final state of registers, Z_f.

2. Simulation:

    (a) Allows m_i cycles of the first vowel.

    (b) Transition to the second vowel from m=1 to m=5.

    (c) The second vowel lasts until m_f.

    (d) Given gp88, V(z), and Z_f from the initialization;

```
For $m = -m_i$ to $m_f$
    Move poles and update V(z)
    $Z_i = Z_f$
    Filter gp88 using V(z) and $Z_i$, to form $y_m$ and $Z_f$.
    Concatenate $y_m$ onto y
End
```

    (e) Pass y through R(z)

3. Play y through the sound card using a sampling frequency of 10 kHz.

The transition region for the vowel pair /u/ as in “boot” to /i/ as in “eve” is shown in Figure 4. The transition into the second vowel begins at time = 12 msec and is essentially over by 60 msec. This corresponds to m=1 to m=5 in the For Loop above. The MATLAB® script file use to generate the complete vector shown in Figure 4 is included as an Appendix.

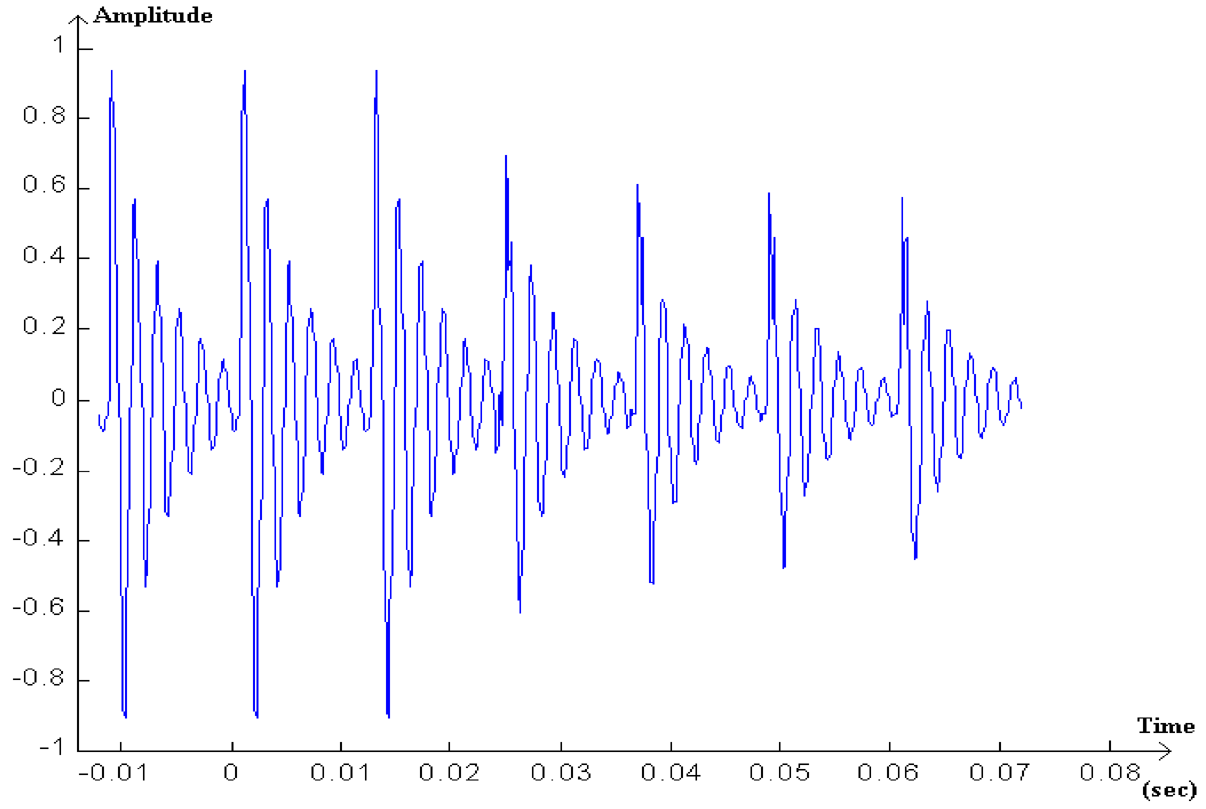

Figure 4: Speech Segment During Transition From /u/ To /i/.

The first two cycle in Figure 4 represents the steady state vowel, /u/, output while the last cycle represents the steady state vowel, /i/. The size difference would correspond to a loudness difference and the waveform differences would differentiate the vowel sounds.

Conclusion

The simulation of a time-varying vocal tract that generates sequential vowels, with a natural transition, requires the modification of the vocal tract digital filter coefficients as the glottal pulse train excites the vocal tract model. A procedure for the implementation of a pseudo time-varying digital filter that simulates the vocal tract producing a controlled movement of the poles associated with one vowel into the correct position for the second vowel has been presented. Since MATLAB® does not allow time-varying digital filters, the key decision was the time interval used for the incremental changes in the vocal tract digital filter coefficients. This was chosen to be the glottal pulse period, which was set for a male speaker. The transition interval from one vowel to the next was selected so the listener would sense a natural change. Simulation results were presented for a /u/ - /i/ vowel transition.

Bibliography

1. Computer-Based Exercises for Signal Processing Using MATLAB®, by C. S. Burrus, J. H. McClellan, A. V. Oppenheim, T. W. Parks, R. W. Scafer, and H. W. Schuessler, Prentice-Hall, 1994, p. 336.

2. Discrete-Time Processing of Speech Signals, J. R. Deller, Jr., J. G. Proakis, and John H. L. Hansen, Prentice-Hall, 1987, p. 193.

3. Ibid., p. 195.

4. Ibid., p. 121.

PAUL COYNE

Paul J. Coyne, Jr. is a Professor of Electrical Engineering and Engineering Science and Associate Director of the Master of Engineering Science Program at Loyola College in Baltimore, Maryland. Dr. Coyne received a Bachelor of Electrical Engineering in 1970, a Masters of Electrical Engineering in 1973 and a Ph.D. in Applied Science in 1976 all from the University of Delaware. Dr. Coyne has been at Loyola College since 1976.

Appendix

```
% Script file that simulates the vowel sound /u/ and
%     transition to vowel sound /i/.
% Calls user defined functions gp, allpoles and move.
imp=[1 zeros(1,119)]; % create impulse
[b,a]=gp(0.88); % generate glottal pulse filter coefficients
gp88=filter(b,a,imp); % create a single glottal pulse period, 120 long
Ff=5000; % folding frequency is 5000 Hz
r1i=0.98; % initial radius for pole 1
a1i=500.; % initial angle for pole 1
r2i=0.96;  % initial radius for pole 2
a2i=1500.0; % initial angle for pole 2
r3i=0.85; % initial radius for pole 3
a3i=2750.0; % initial angle for pole 3
r4i=0.85; % initial radius for pole 4
a4i=3600.0; % initial angle for pole 4
r1f=0.98; % final radius for pole  1
a1f=500.0; % final angle for pole 1
r2f=0.9; % final radius for pole 2
a2f=2100.0; % final angle for pole 2
r3f=0.89; % final radius for pole 3
a3f=2700.0; % final angle for pole 3
r4f=0.92; % final radius for pole 4
a4f=3500.0; % final angle for pole 4
sf=1.0; % scale factor for z-plane
[b,a]=allpoles(Ff,r1i,a1i,r2i,a2i,r3i,a3i,r4i,a4i,sf);% initial all-pole
filter
[y,Zf]=filter(b,a,gp88);% initial filter response
% transition to second vowel begins at m=1 and lasts until m=5.
```

```
for n=-450:1:450
   [r1,a1]=move(r1i,r1f,a1i,a1f,n);
   [r2,a2]=move(r2i,r2f,a2i,a2f,n);
   [r3,a3]=move(r3i,r3f,a3i,a3f,n);
   [r4,a4]=move(r4i,r4f,a4i,a4f,n);
   [b,a]=allpoles(Ff,r1,a1,r2,a2,r3,a3,r4,a4,sf);
   Zi=Zf;
   [yn,Zf]=filter(b,a,gp88,Zi);
   y=[y yn];
end
y=conv(y,[1 -1]);% model of sound passing lips
sound(y,10000)% a sampling frequency of 10,000 Hz. is assumed

function [b,a]= gp(ap)
% returns digital filter coefficients using parameter ap.
e=2.71828;
b=[0 -ap*e*log(ap)];
a=[1 -2*ap ap^2];

function [b,a] = allpoles(Ff,p1,a1,p2,a2,p3,a3,p4,a4,sf)
% Ff = folding frequency
% p# = radius of pole pair # from origin
% a# =  frequency location of pole pair #
% sf = pole-zero plot scale factor
% b = vector of input weighting coefficients
% a = vector of output weighting coefficients
%% [b,a] = allpoles(p1,a1,p2,a2,p3,a3,p4,a4)
p1a=p1*(cos(a1*pi/Ff)+sin(a1*pi/Ff)*j);
p1b=conj(p1a);
p2a=p2*(cos(a2*pi/Ff)+sin(a2*pi/Ff)*j);
p2b=conj(p2a);
p3a=p3*(cos(a3*pi/Ff)+sin(a3*pi/Ff)*j);
p3b=conj(p3a);
p4a=p4*(cos(a4*pi/Ff)+sin(a4*pi/Ff)*j);
p4b=conj(p4a);
[b,a]=zp2tf([],[p1a;p1b;p2a;p2b;p3a;p3b;p4a;p4b],sf);

function [r,a]=move(ri,rf,ai,af,m)
% m < 0 leaves poles at initial location
% At m=1 the transition begins
% ri and ai are the initial radius and angle of a pole
% rf and af are the final radius and angle of a pole
n=m*120;
if n<= 0
   r=ri;
   a=ai;
else
   r=rf+(ri-rf)*(0.994^n);
   a=af+(ai-af)*(0.994^n);
end
```