EE 679 Speech Processing

Computing Assignment 1: Signal synthesis

Roll 09d07040 Swrangsar Basumatary Date: Aug 22 2012

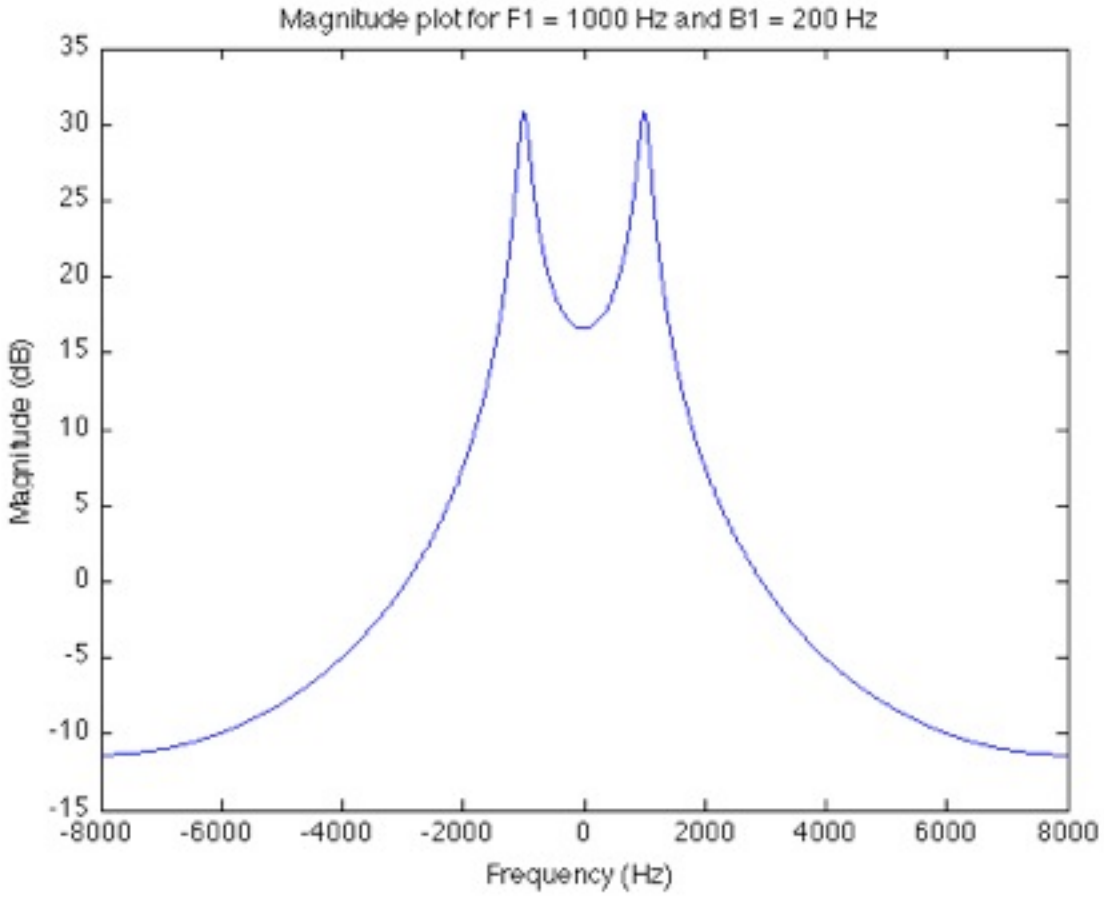
1)
The transfer function H(z) = (z*z)/((z-a)(z-a*)), where a = r*exp(jθ).
r = exp(-pi*B1/Fs);
θ = 2*pi*F1/Fs;

The MATLAB code:

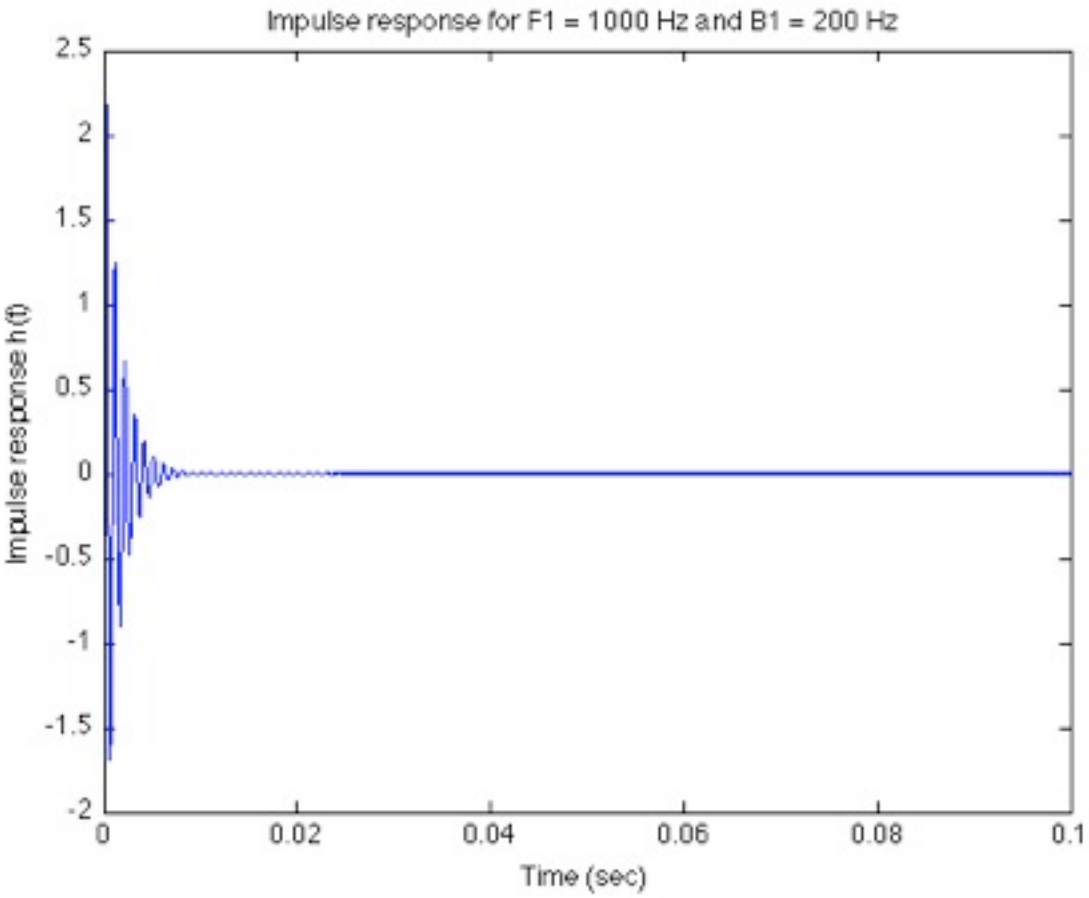
```
function [impulseResponse] = singleFormantResonator(F1, B1)
%SINGLEFORMATRESONATOR Creates a new filter for a single formant
resonator.
%  SINGLEFORMANTRESONATOR(F1, B1) creates a new transfer
function H(z)
%  for a digital filter for a single formant resonator. It uses
the input
%  formant frequency(F1), bandwidth(B1) and sampling
frequency(Fs) to do
%  this. It also plots the magnitude response (dB magnitude vs
frequency)
%  and impulse response.

Fs = 16000; % 16 kHz
poleRadius = exp(-(pi*B1/Fs));
poleTheta = 2*pi*F1/Fs;
pole1 = poleRadius*exp(1i*poleTheta);
pole2 = poleRadius*exp(-1i*poleTheta);

M = 8000;
step = 1;
f = -M:step:M;
r = 1;
z = r*exp(1i*2*pi*f/Fs);
H = (z.*z)./((z-pole1+eps).*(z-pole2+eps));
mag = 20*log10(abs(H));
figure, plot(f, mag);
title(['Magnitude plot for F1 = ', num2str(F1), ' Hz and B1 = ',
num2str(B1), ' Hz']);
xlabel('Frequency (Hz)');
```

```
ylabel('Magnitude (dB)');

impulseResponse = real(ifft(ifftshift(H))); % do ifft on non-
centered Fourier transform
n = 0:1:((M*2)/step);
t = n/Fs;
figure, plot(t, impulseResponse);
title(['Impulse response for F1 = ', num2str(F1), ' Hz and B1 =
', num2str(B1), ' Hz']);
xlabel('Time (sec)');
ylabel('Impulse response h(t)');
```

The magnitude plot:



The impulse response:



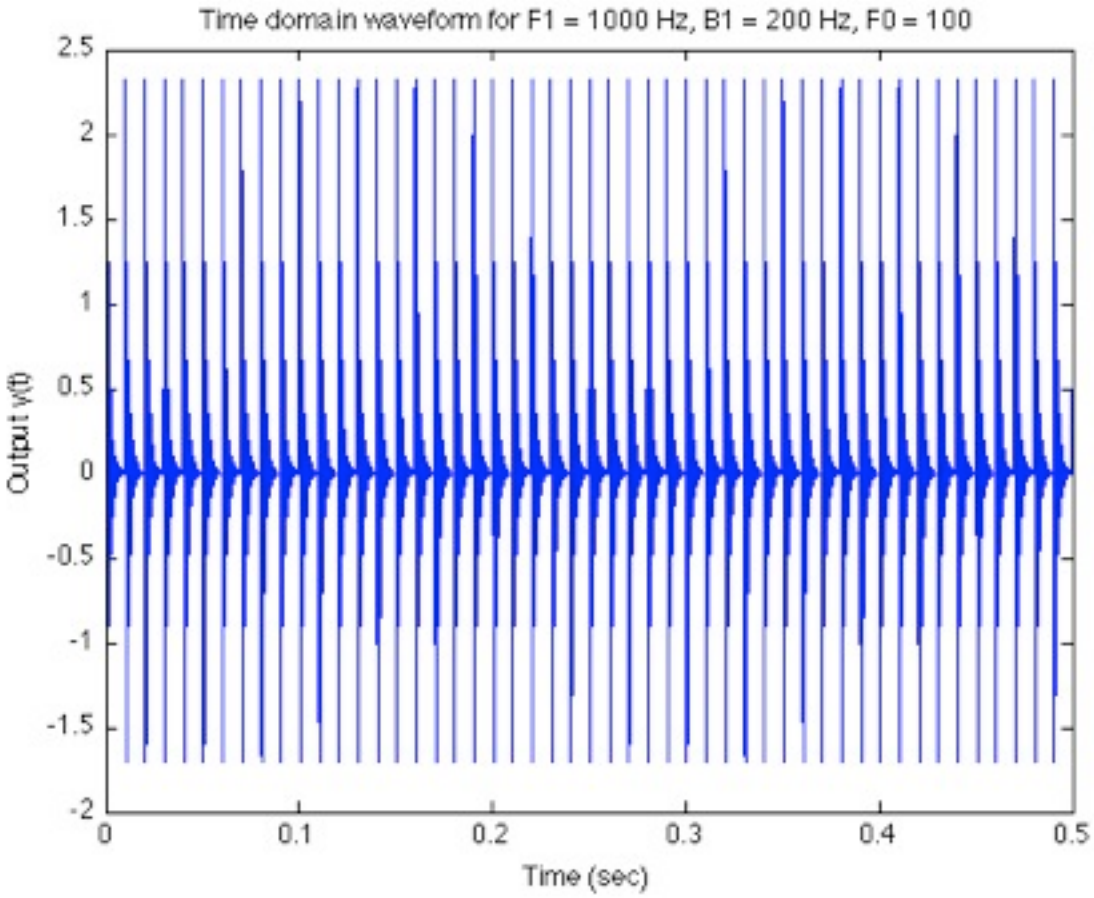
2)
Exciting the above resonator (filter) with a source given by an impulse train of F0 = 100 Hz.

The code:

```
function sourceFilterSystem(F1, B1, F0)
%SOURCEFILTERSYSTEM Models a filter system based on a single
formant resonator.

Fs = 16000;
impulseResponse = singleFormantResonator(F1, B1);
impulseResponse = impulseResponse(1:(Fs/2)+1);
length(impulseResponse)

duration = 2;
itlen = (duration*Fs);
impulseTrain = zeros(1,itlen+1);
for k = 1:round(Fs/F0):itlen+1
    impulseTrain(k) = 1;
```

```
end
y = conv(impulseResponse, impulseTrain, 'same');
n = 0:length(y)-1;
t = n/Fs;
figure, plot(t, y);
title(['Time domain waveform for F1 = ', num2str(F1), ' Hz, B1 =
', num2str(B1), ' Hz, F0 = ', num2str(F0)]);
xlabel('Time (sec)');
ylabel('Output y(t)');

sound(y, Fs);
```

The time domain waveform:

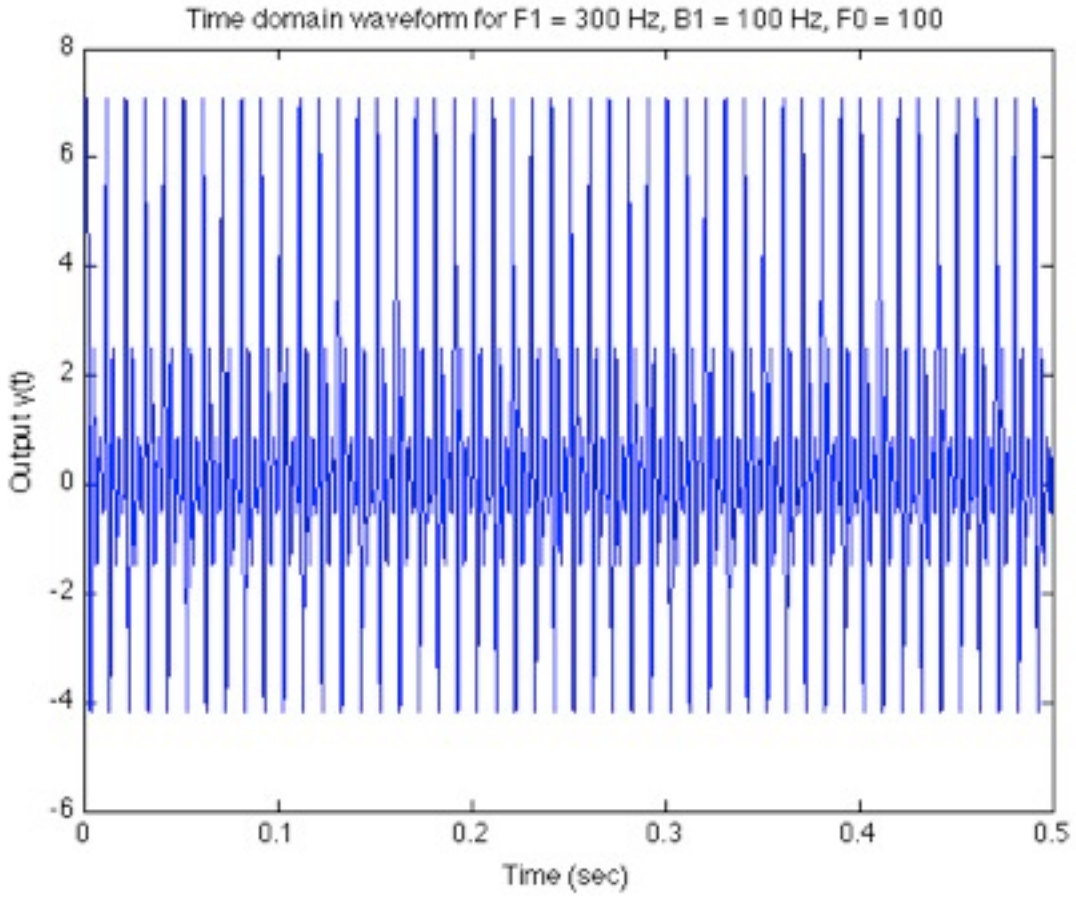


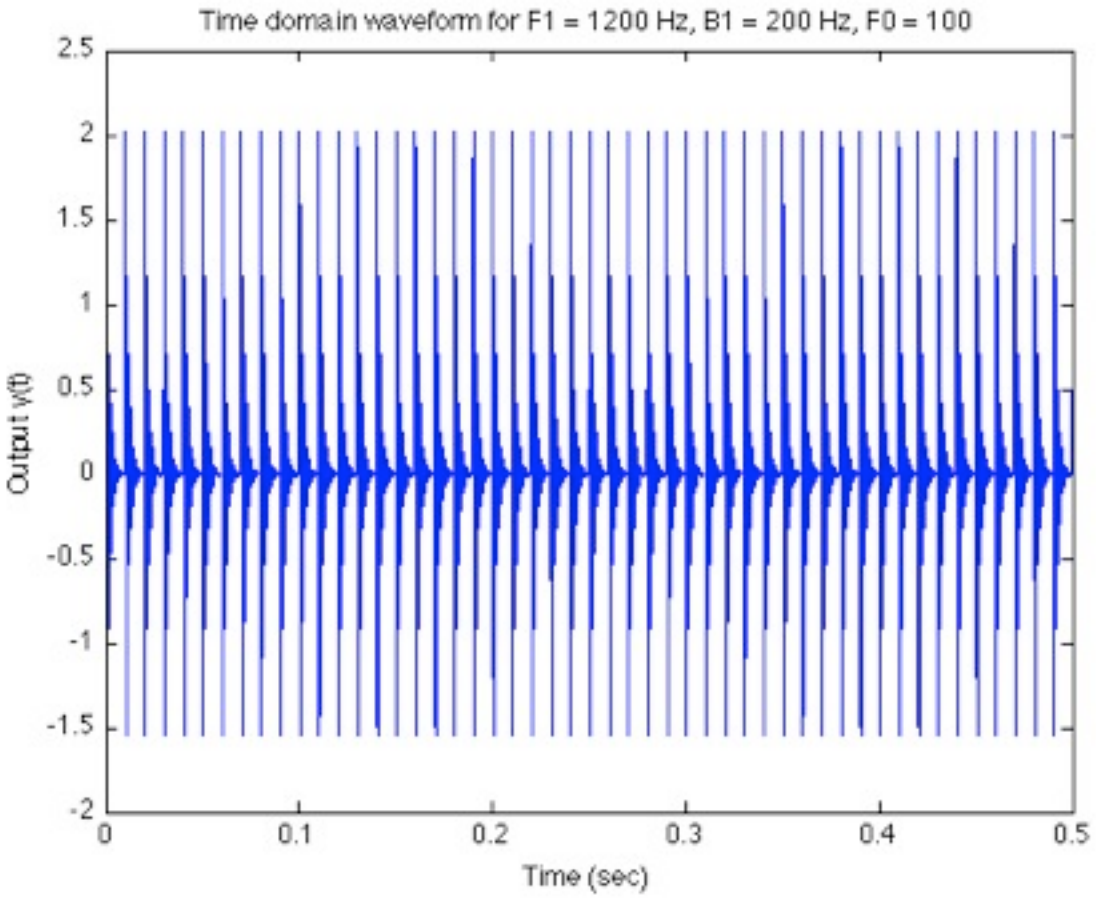
Comment:

It has a good sound quality. We hear a buzzing kind of sound with a pitch of 100hz.

3)

(a) F1=300Hz,B1=100Hz; F1=1200Hz,B1=200Hz

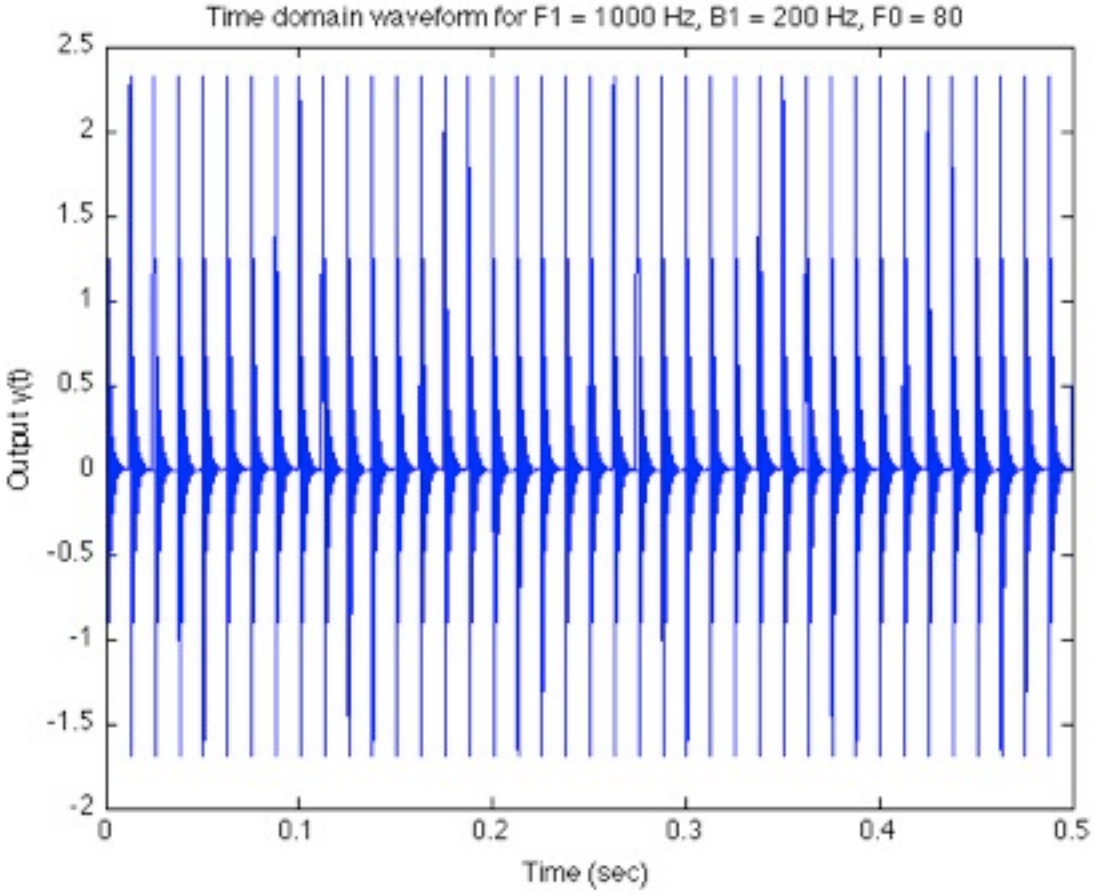


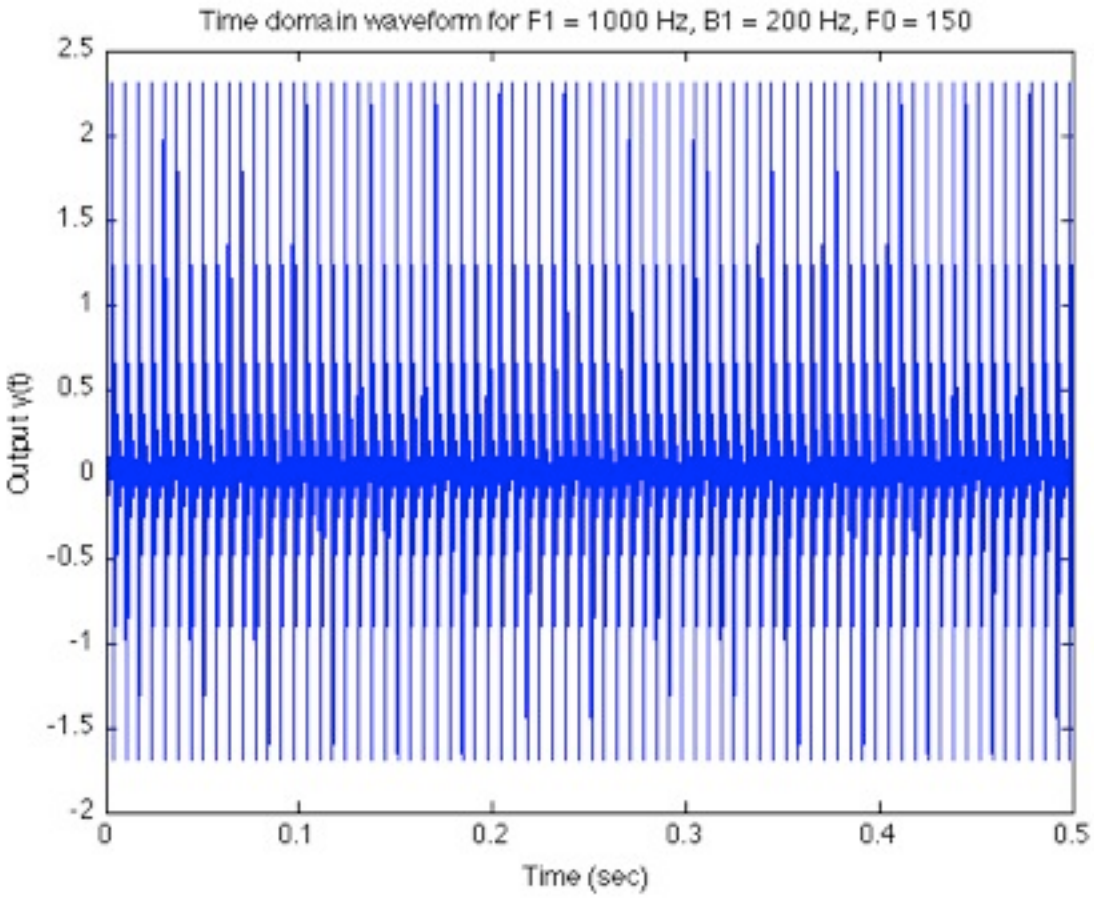


Comment:

The filter with the higher bandwidth i.e. the 2nd one has a clearer and a brighter sound because its formant is high which gives it more brightness/timbre and the higher bandwidth also lets most of the energy near the formant frequency pass through the filter giving higher amplitude or higher intensity. The second waveform is also clearer.

(b) F0=80Hz;F0=150Hz


Time domain waveform for F1 = 1000 Hz, B1 = 200 Hz, F0 = 80
Output w(t)
Time (sec)
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.1
0.2
0.3
0.4
0.5



Comment:

The 2nd sound has a higher pitch (F0) of 150 Hz compared to 80 Hz of the other one. The harmonics 960 and 1040 Hz of the first sound lies within +/-40 Hz of the formant frequency 1000 Hz (F1). While the harmonic 1050 lies a 50 Hz distance away from F1. So the first sound gets amplified more than the second one. The second waveform has more chances of the impulse response overlapping (because of higher pitch).

4)
Vowel F1, F2, F3
/a/ 730, 1090, 2440
/i/ 270, 2290, 3010
/u/ 300, 870, 2240

Code for the resonator function:

```
function [impulseResponse] = vowelResonator(vowel)
%VOWELRESONATOR Creates a new filter WITH 3 formant frequencies.
%  VOWELRESONATOR('VOWEL') creates a new transfer function H(z)
```

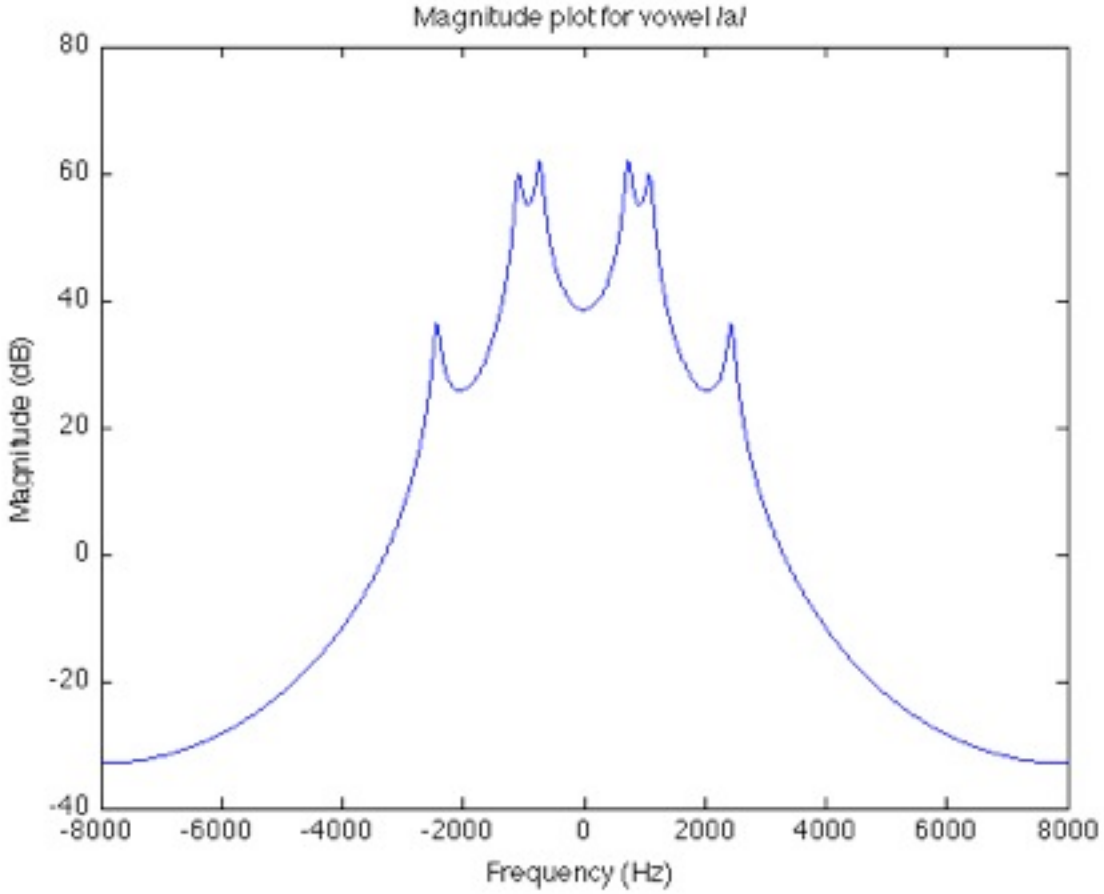
```
%  for a digital filter with three formants. It uses the input
%  string 'VOWEL' to do
%  this. It also plots the magnitude response (dB magnitude vs
frequency)
%  and impulse response.

switch vowel;
    case 'a'
        F1 = 730; F2 =1090; F3 = 2440;
    case 'i'
        F1 = 270; F2 = 2290; F3 = 3010;
    case 'u'
        F1 = 300; F2 = 870; F3 = 2240;
    otherwise
        error('Unknown or undefined vowel.')
end

Bw = 100;
Fs = 16000; % 16 kHz
poleRadius = exp(-(pi*Bw/Fs));
poleTheta = 2*pi*F1/Fs;
pole1 = poleRadius*exp(1i*poleTheta);
pole2 = poleRadius*exp(-1i*poleTheta);

M = 8000;
step = 1;
f = -M:step:M;
r = 1;
z = r*exp(1i*2*pi*f/Fs);
H1 = (z.*z)./((z-pole1+eps).*(z-pole2+eps));
poleTheta = 2*pi*F2/Fs;
pole1 = poleRadius*exp(1i*poleTheta);
pole2 = poleRadius*exp(-1i*poleTheta);
H2 = (z.*z)./((z-pole1+eps).*(z-pole2+eps));
poleTheta = 2*pi*F3/Fs;
pole1 = poleRadius*exp(1i*poleTheta);
pole2 = poleRadius*exp(-1i*poleTheta);
H3 = (z.*z)./((z-pole1+eps).*(z-pole2+eps));
H = H1.*H2.*H3;


% mag = 20*log10(abs(H));
% figure, plot(f, mag);
% title(['Magnitude plot for vowel /', vowel, '/']);
% xlabel('Frequency (Hz)');
% ylabel('Magnitude (dB)');
```

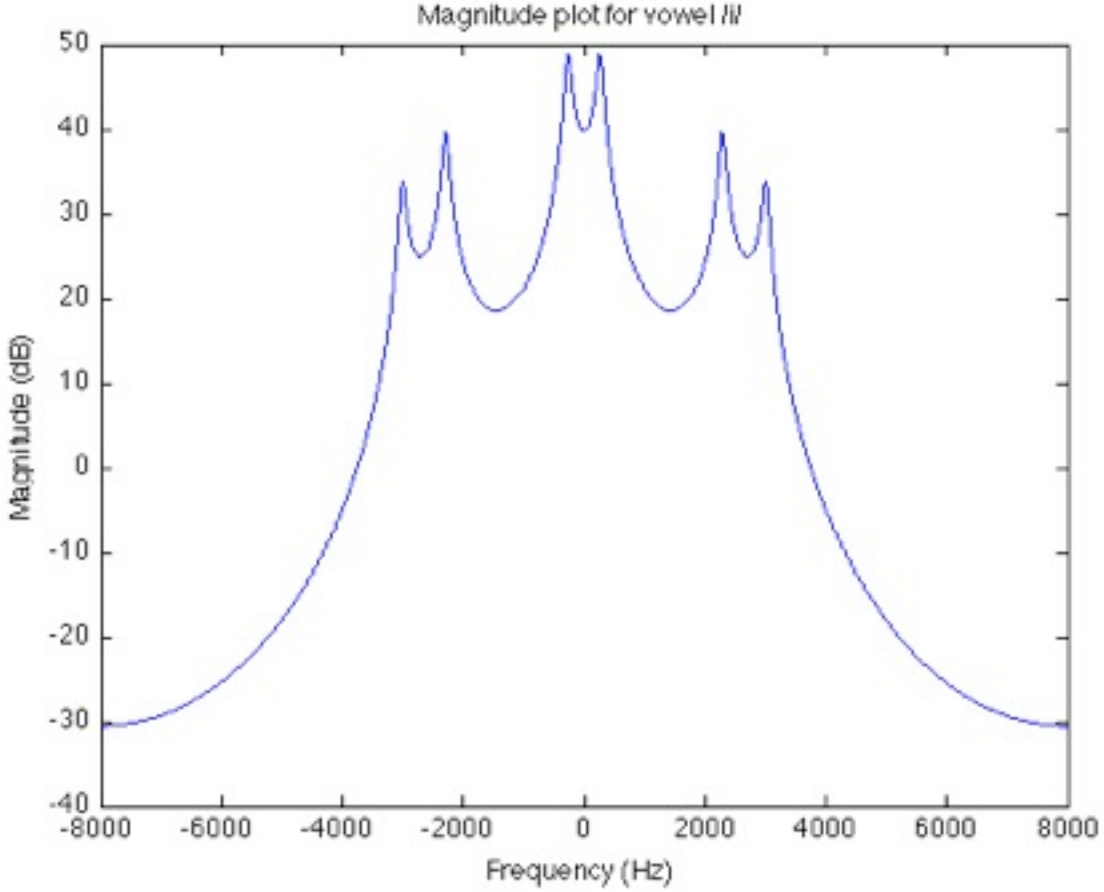
```
impulseResponse = real(ifft(ifftshift(H))); % do ifft on non-
centered Fourier transform
% n = 0:1:((M*2)/step);
% t = n/Fs;
% figure, plot(t, impulseResponse);
% title(['Impulse response for vowel /', vowel, '/']);
% xlabel('Time (sec)');
% ylabel('Impulse response h(t)');
```

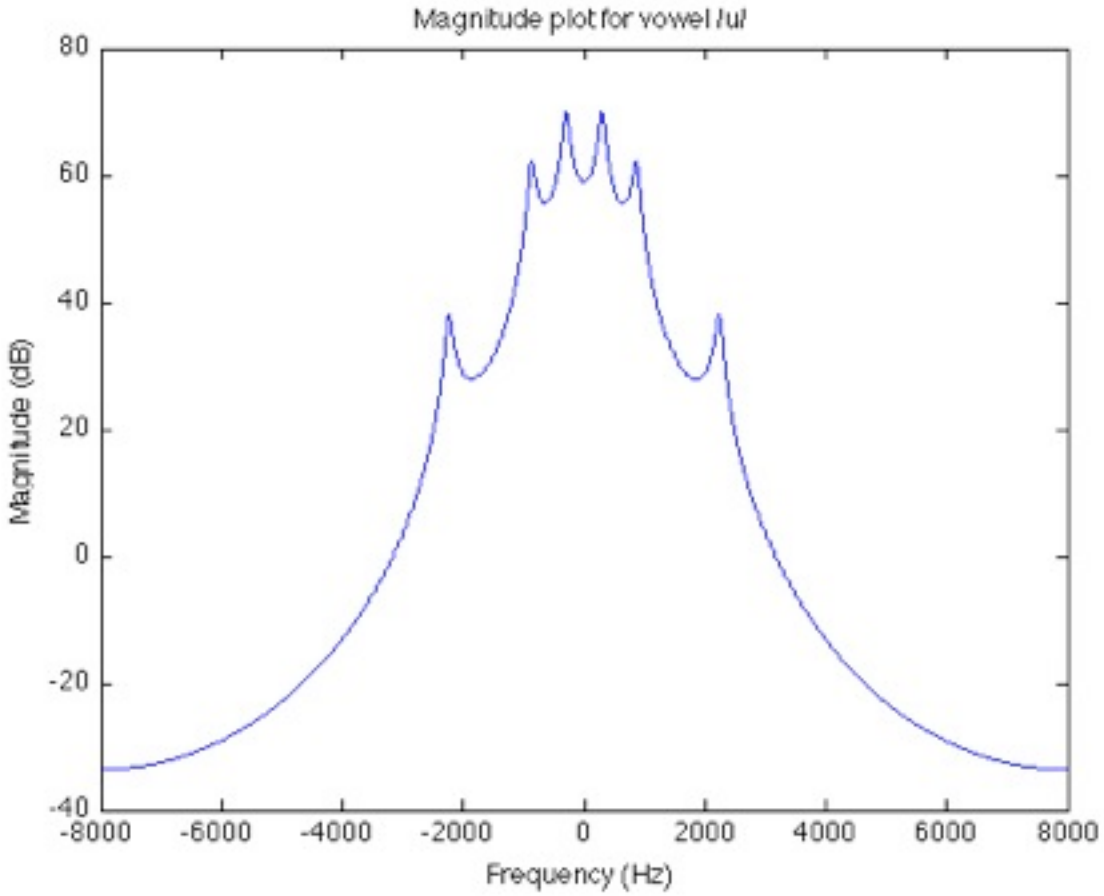
Code for the filter system excited by impulse train:

```
function y = vowelFilterSystem(vowel, F0)
%VOWELFILTERSYSTEM Models a vowel filter system based on a given
pitch.

Fs = 16000;
impulseResponse = vowelResonator(vowel);
impulseResponse = impulseResponse(1:Fs);
% length(impulseResponse)

duration = 2;
step = round(Fs/F0);
numStep = ceil(duration*Fs/step);
itlen = (numStep*step);
impulseTrain = zeros(1,itlen);
for k = 1:step:itlen
    impulseTrain(k) = 1;
end
y = conv(impulseResponse, impulseTrain, 'same');
y = y(1:Fs/2);
n = 0:length(y)-1;
t = n/Fs;
figure, plot(t, y);
title(['Time domain waveform for vowel /', vowel, '/ and F0 = ',
num2str(F0)]);
xlabel('Time (sec)');
ylabel('Output y(t)');

sound(y, Fs);
```

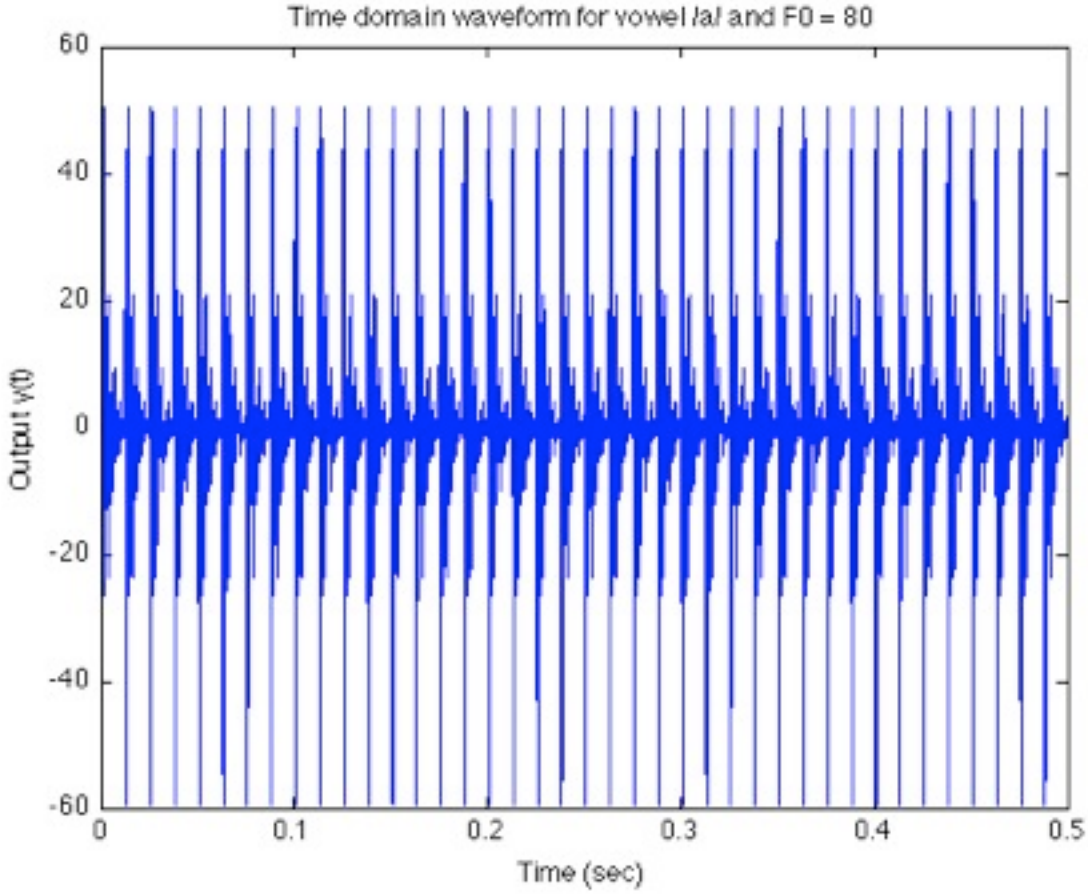
Mag plots for the vowel resonators:

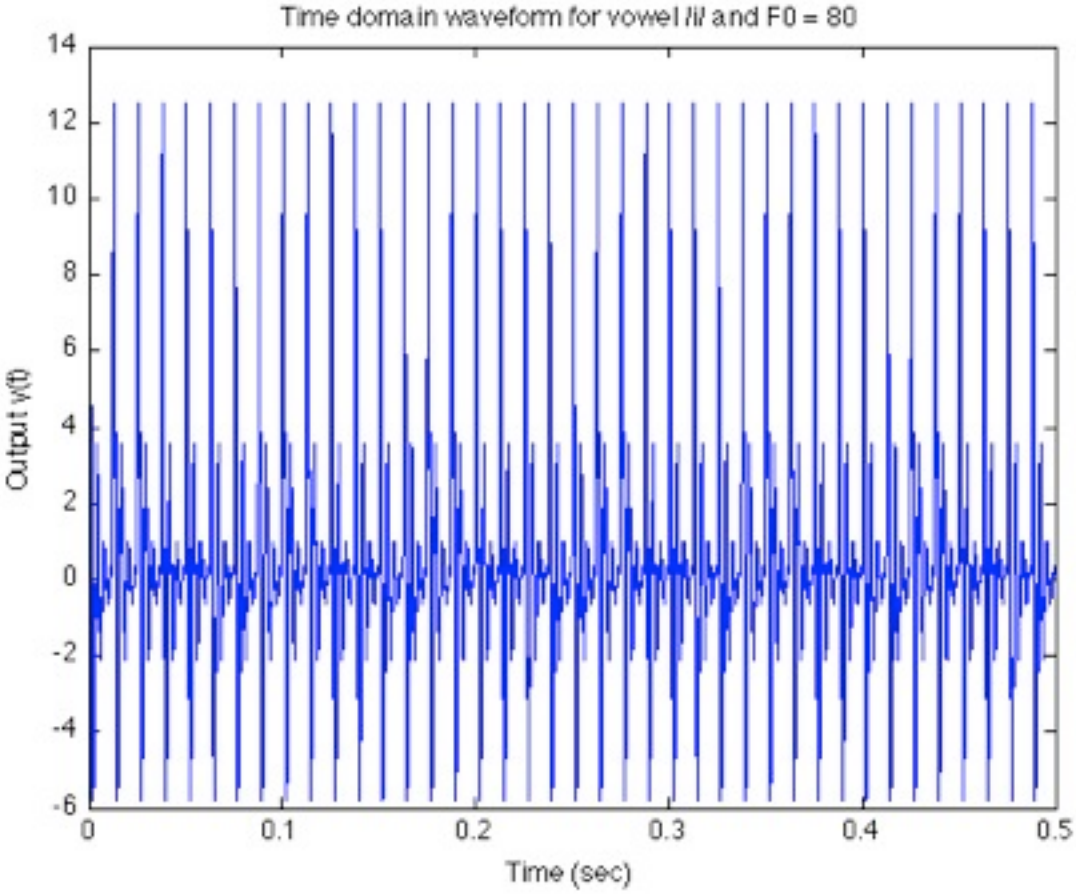

Magnitude plot for vowel /a/
Magnitude (dB)
80
60
40
20
0
-20
-40
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000
Frequency (Hz)

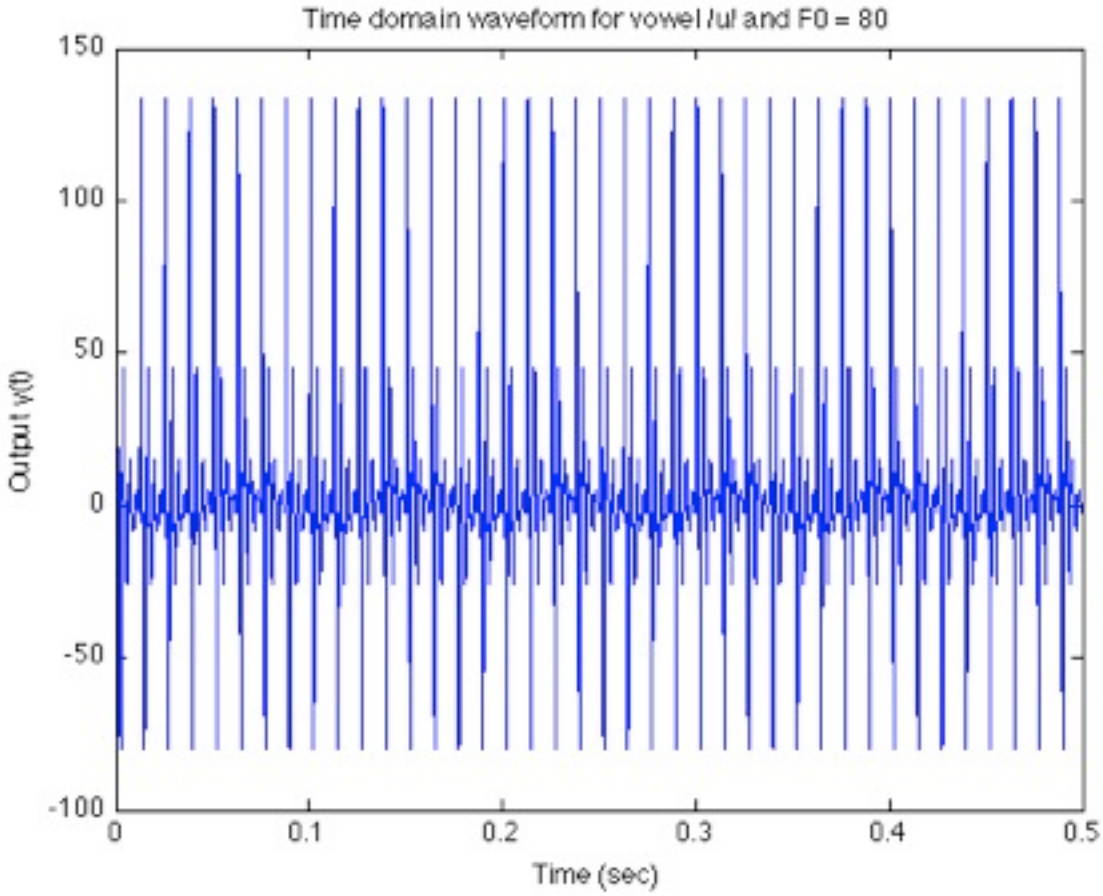

Magnitude plot for vowel /i/
Magnitude (dB)
Frequency (Hz)
50
40
30
20
10
0
-10
-20
-30
-40
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000



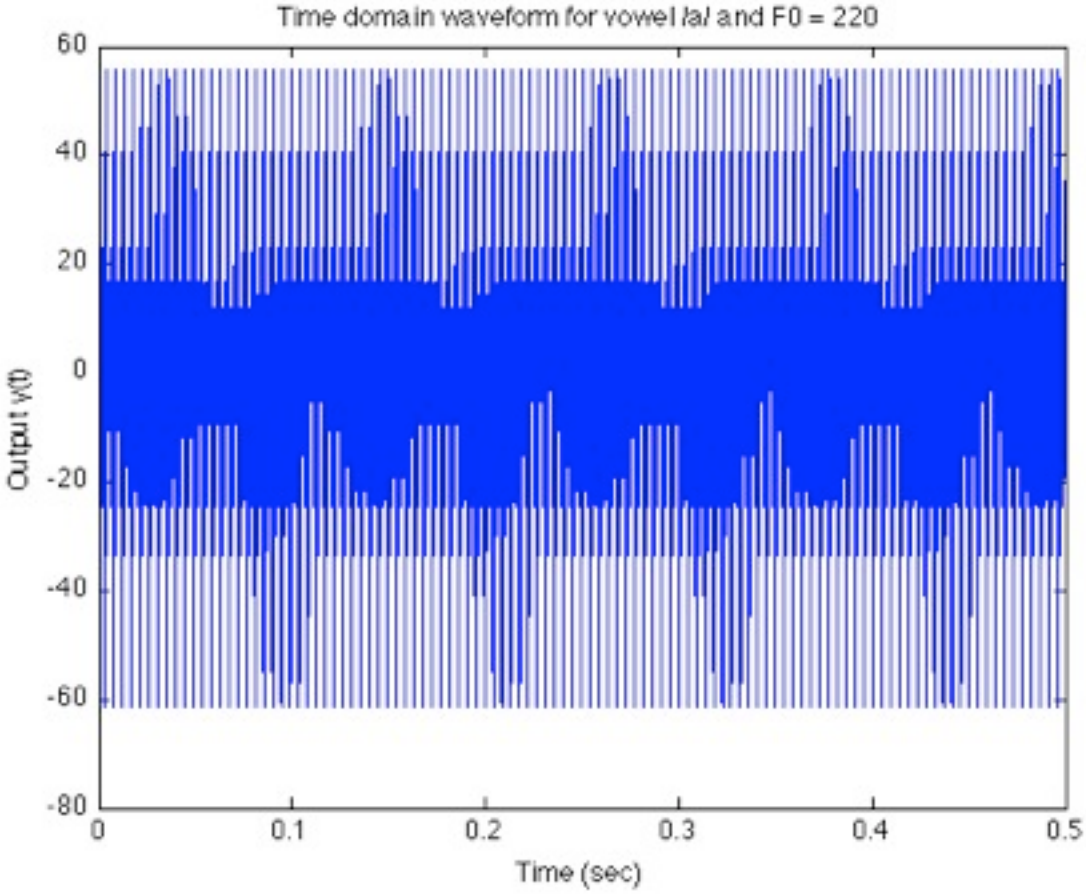
Time domain plots at 80 Hz pitch :

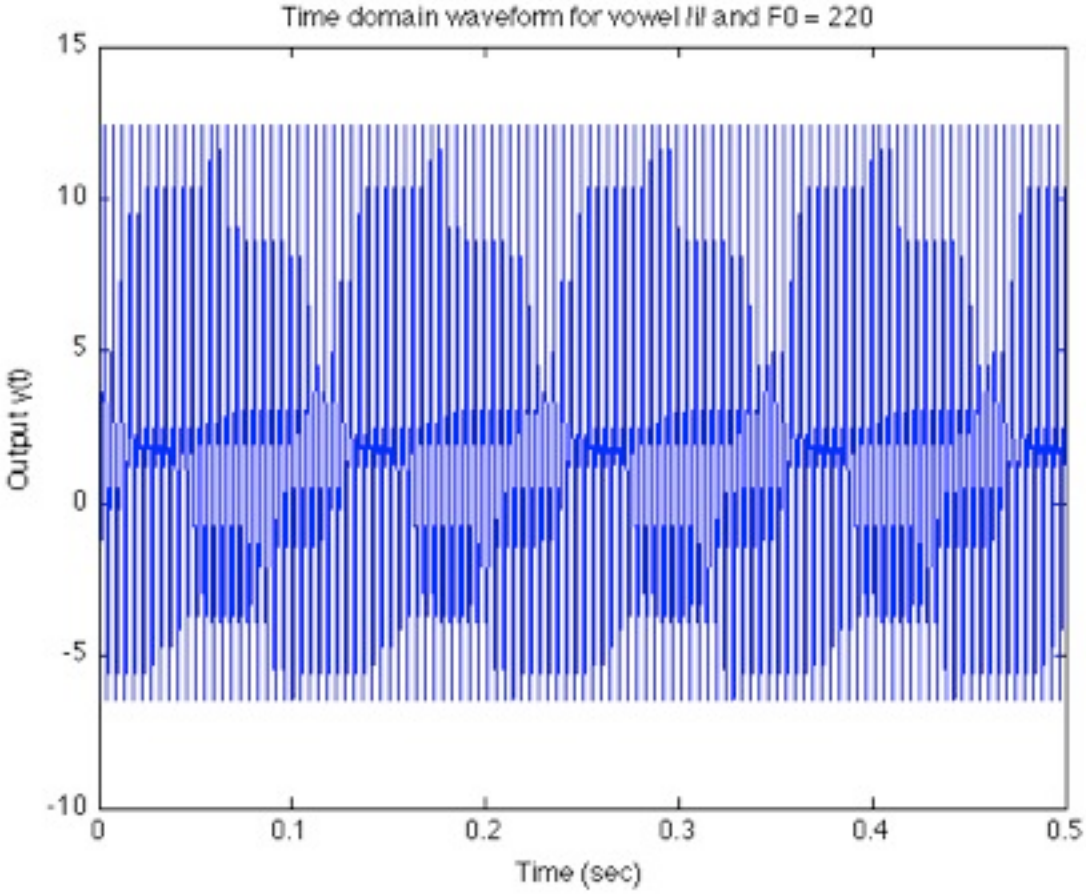

Time domain waveform for vowel /a/ and F0 = 80
60
40
20
0
-20
-40
-60
Output y(t)
0
0.1
0.2
0.3
0.4
0.5
Time (sec)

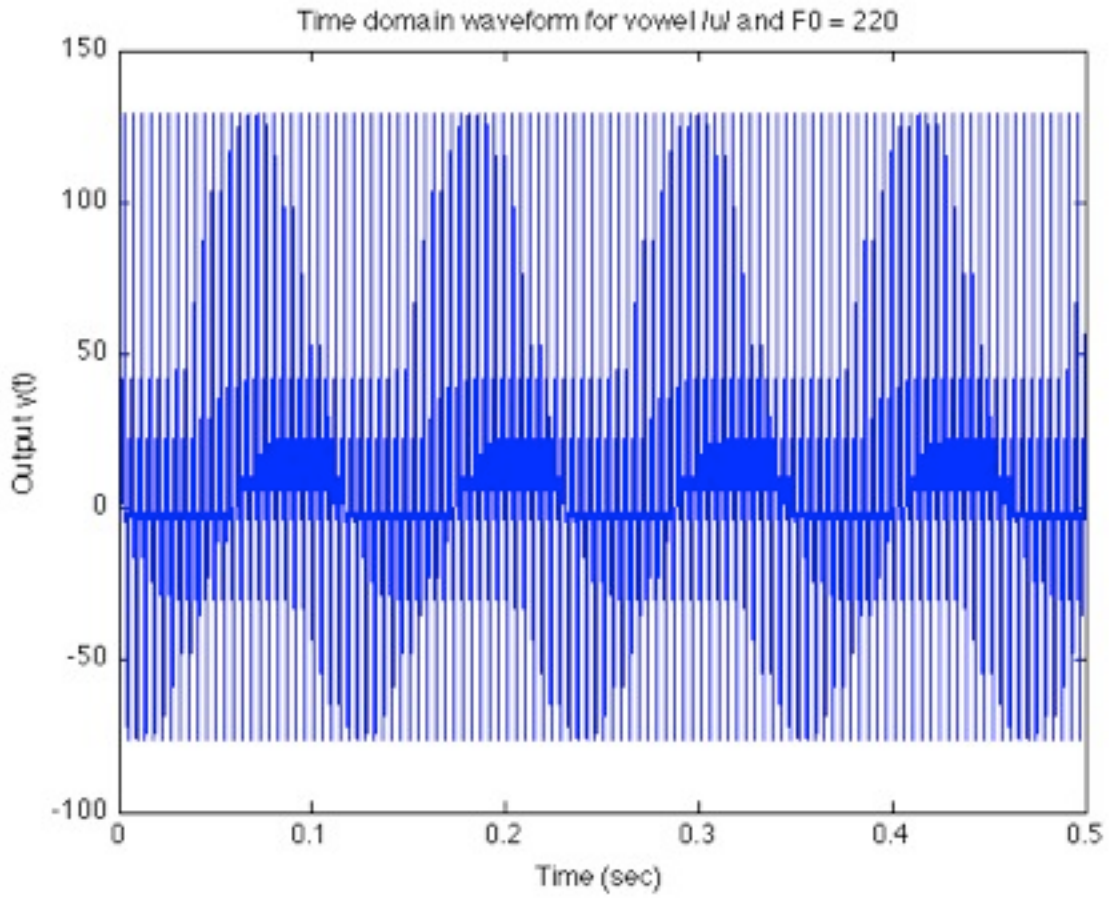

Time domain waveform for vowel /i/ and F0 = 80
Output w(t)
Time (sec)
14
12
10
8
6
4
2
0
-2
-4
-6
0
0.1
0.2
0.3
0.4
0.5



Time domain plots at 220 Hz pitch :


Time domain waveform for vowel /a/ and F0 = 220
60
40
20
0
-20
-40
-60
-80
Output y(t)
0
0.1
0.2
0.3
0.4
0.5
Time (sec)


Time domain waveform for vowel /i/ and F0 = 220
15
10
5
0
-5
-10
Output y(t)
0
0.1
0.2
0.3
0.4
0.5
Time (sec)



5)

Signal analysis:

Code for rectangular window filter:

```
function signal = rectangularVowelFilter(vowel, F0,
windowLength)
%RECTANGULARVOWELFILTER Models a vowel filter system based on a
given pitch.
% WINDOWLENGTH value must be in milliseconds.

Fs = 16000;
signal = vowelFilterSystem(vowel, F0);
rwindow = zeros(1,Fs);
T=1/Fs;
windowPeriod = windowLength/1000;
M = round(windowPeriod/T);
rwindow(1:M) = 1;
signal = signal.*rwindow;
```

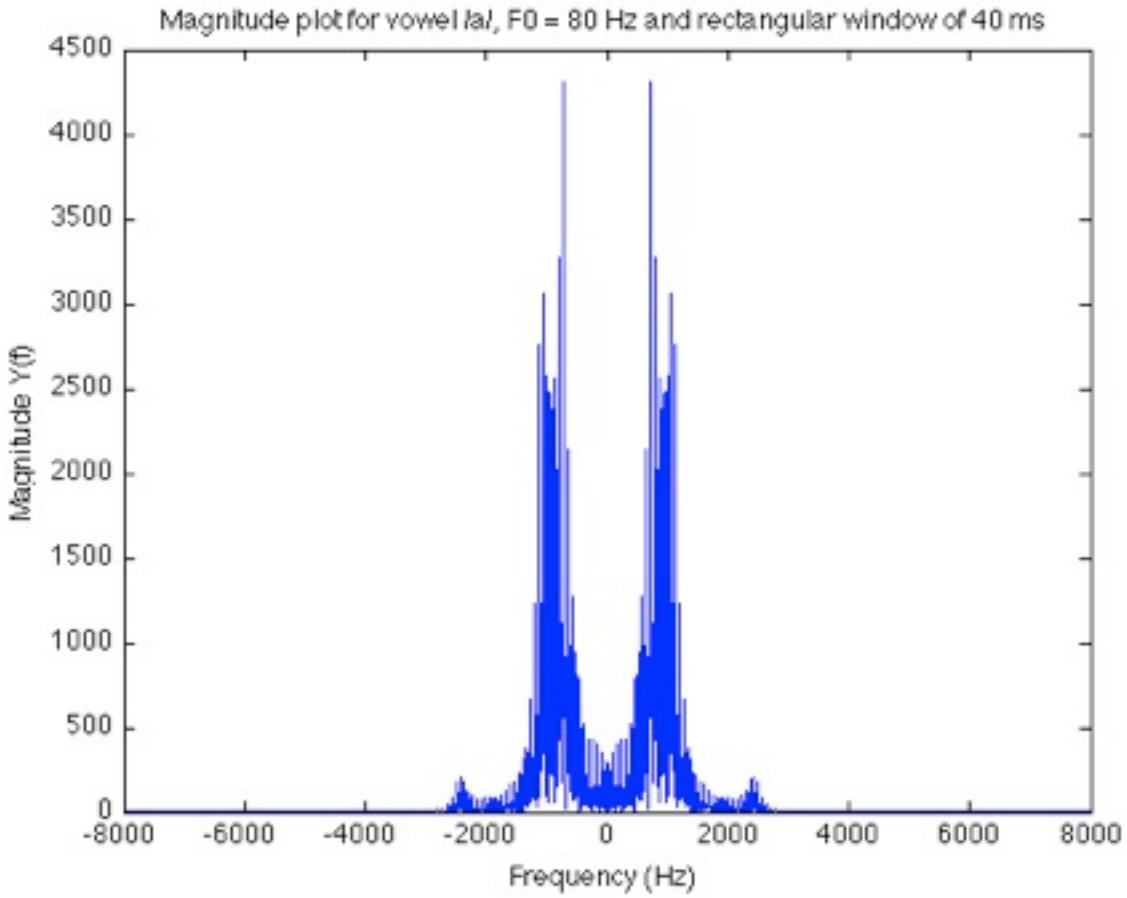
```
Y = fftshift(fft(signal));
n = -length(Y)/2:(length(Y)/2)-1;
freq = n*(Fs/length(Y));
figure, plot(freq, abs(Y));
title(['Magnitude plot for vowel /', vowel, '/, F0 = ',
num2str(F0), ' Hz and rectangular window of ',
num2str(windowLength), ' ms']);
xlabel('Frequency (Hz)');
ylabel('Magnitude Y(f)');
```

Code for Hamming window filter:

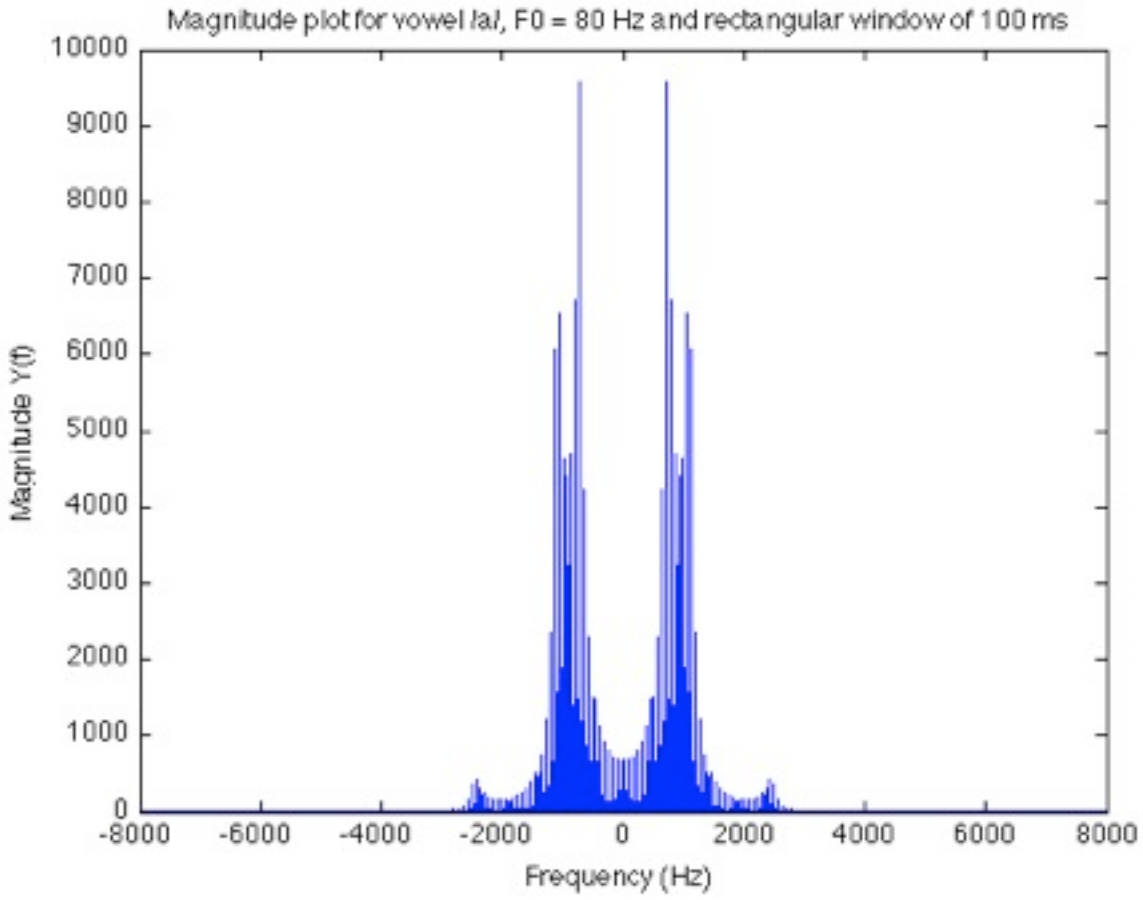
```
function signal = hammingVowelFilter(vowel, F0, windowLength)
%HAMMINGVOWELFILTER Models a vowel filter system based on a
given pitch.

Fs = 16000;
signal = vowelFilterSystem(vowel, F0);
rwindow = zeros(1,Fs);
T=1/Fs;
windowPeriod = windowLength/1000;
M = round(windowPeriod/T);
for k=1:M+1
    rwindow(k) = 0.54-(0.46*(cos(2*pi*(k-1)/M)));
end
signal = signal.*rwindow;

Y = fftshift(fft(signal));
n = -length(Y)/2:(length(Y)/2)-1;
freq = n*(Fs/length(Y));
figure, plot(freq, abs(Y));
title(['Magnitude plot for vowel /', vowel, '/, F0 = ',
num2str(F0), ' Hz and Hamming window of ',
num2str(windowLength), ' ms']);
xlabel('Frequency (Hz)');
ylabel('Magnitude Y(f)');
```

Magnitude plots for /a/ with F0 = 80 Hz:

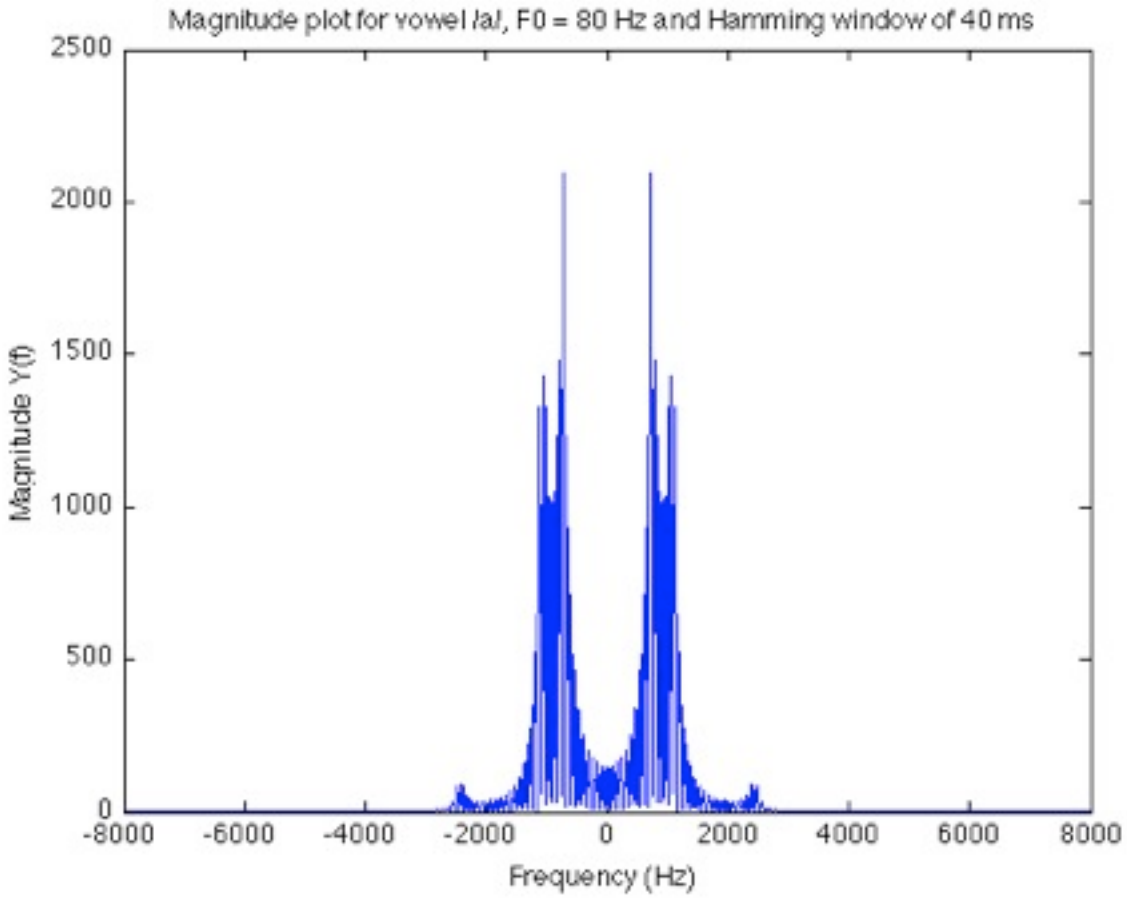
Using rectangular window of 40 ms:



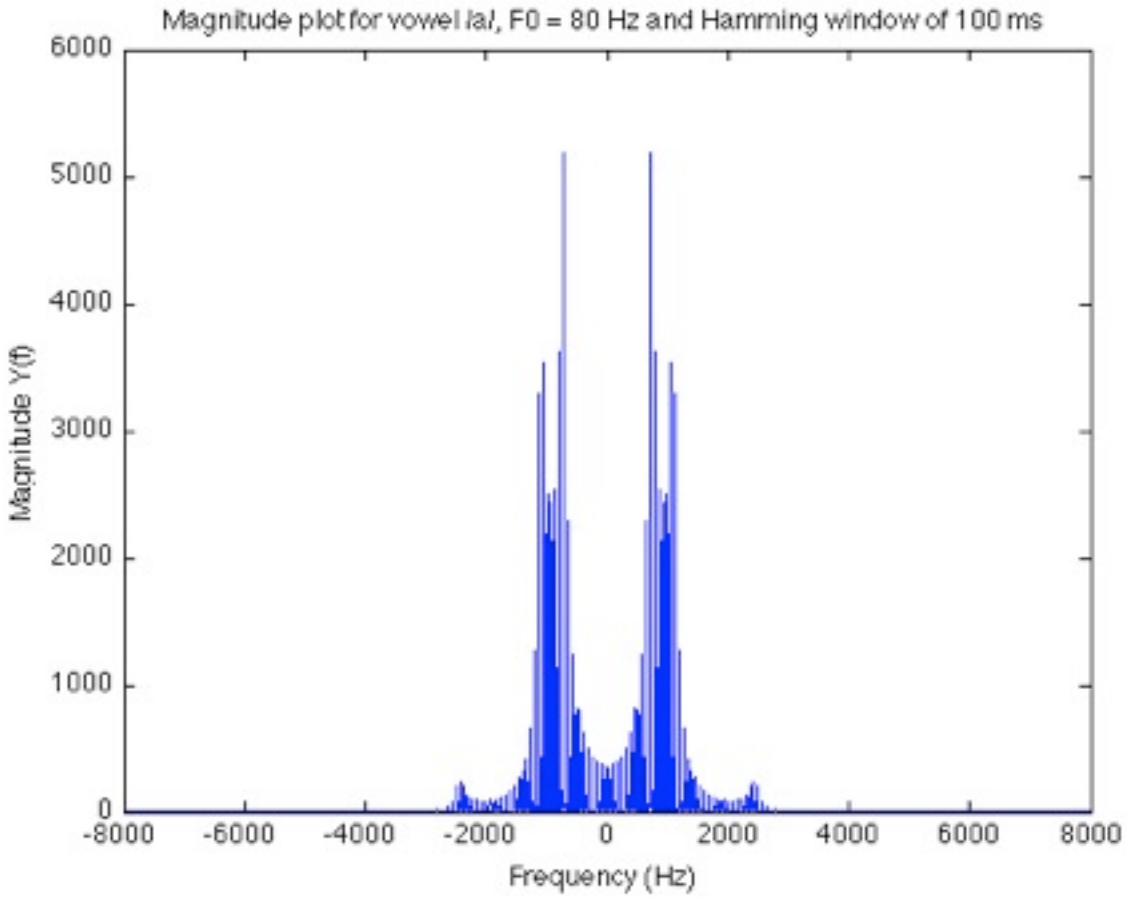
Using rectangular window of 100 ms:



Using Hamming window of 40 ms:

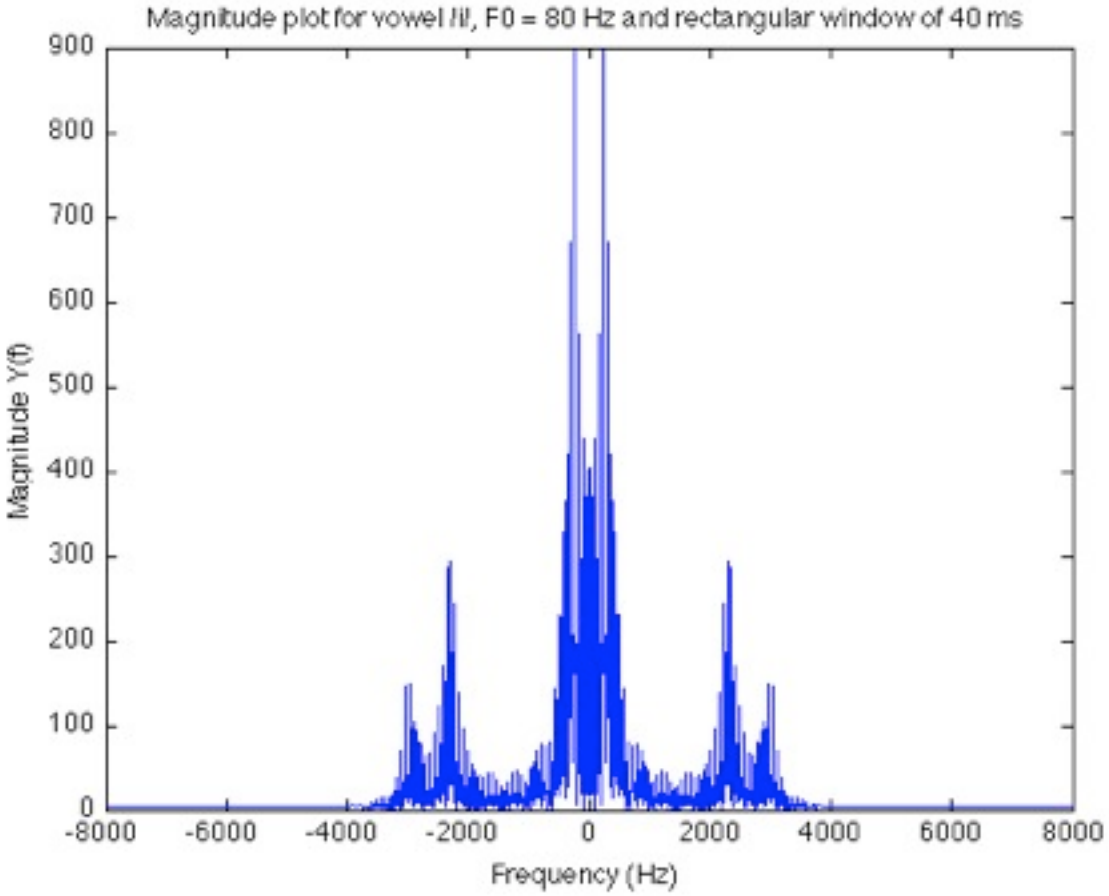


Using Hamming window of 100 ms:

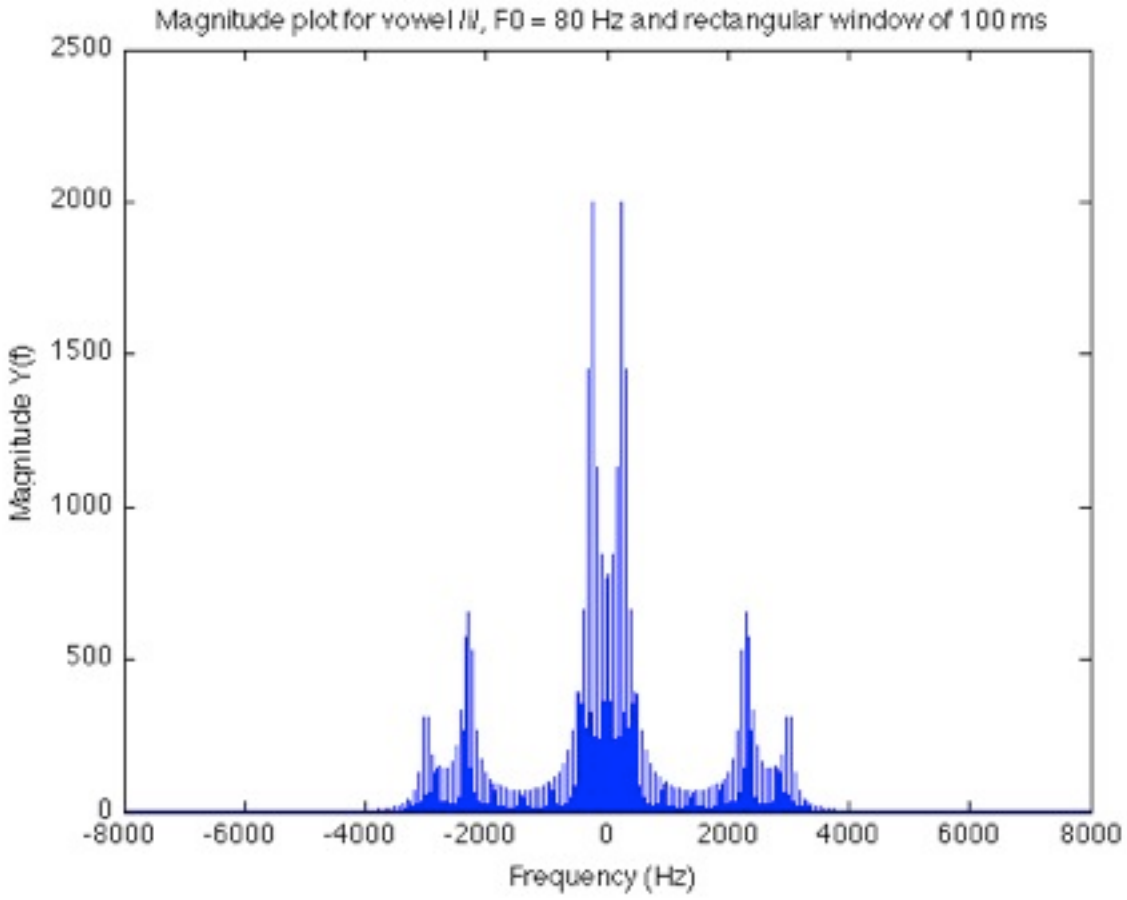


Magnitude plots for /i/ with F0 = 80 Hz:

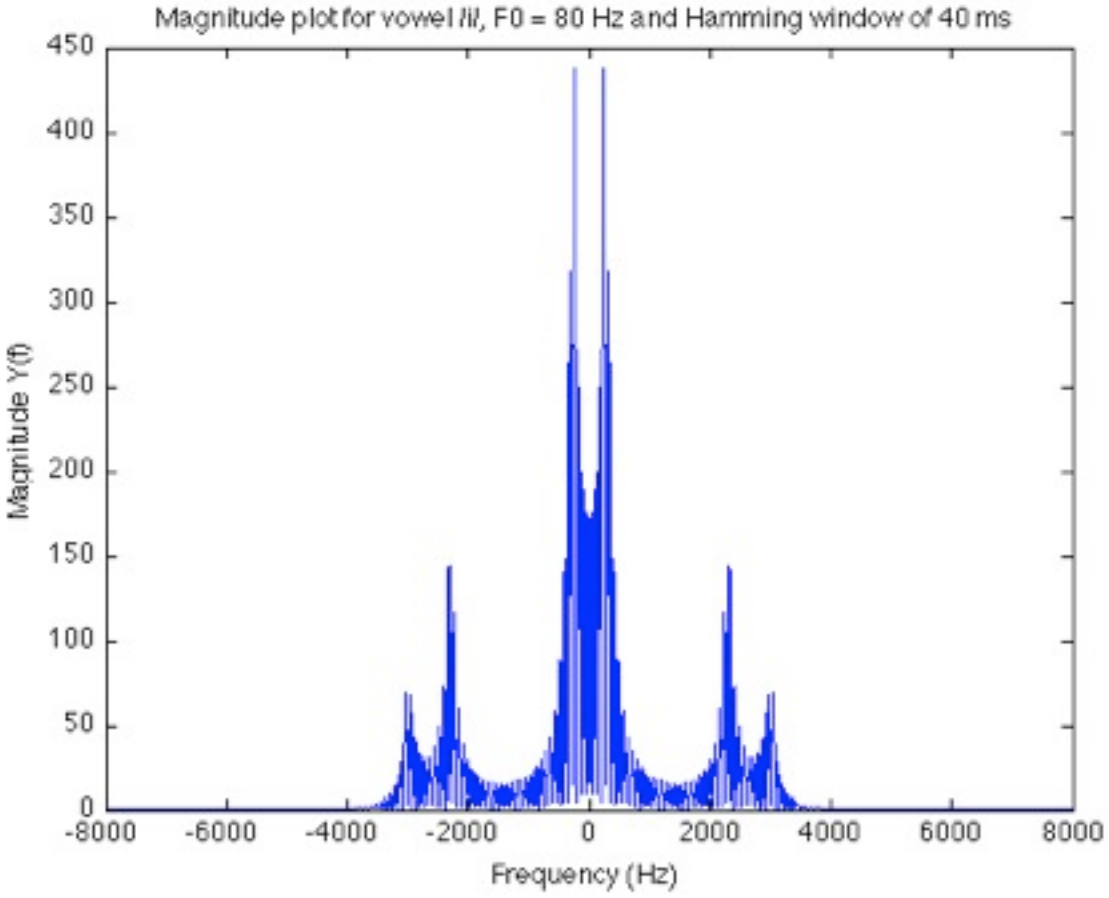
Using rectangular window of 40 ms:



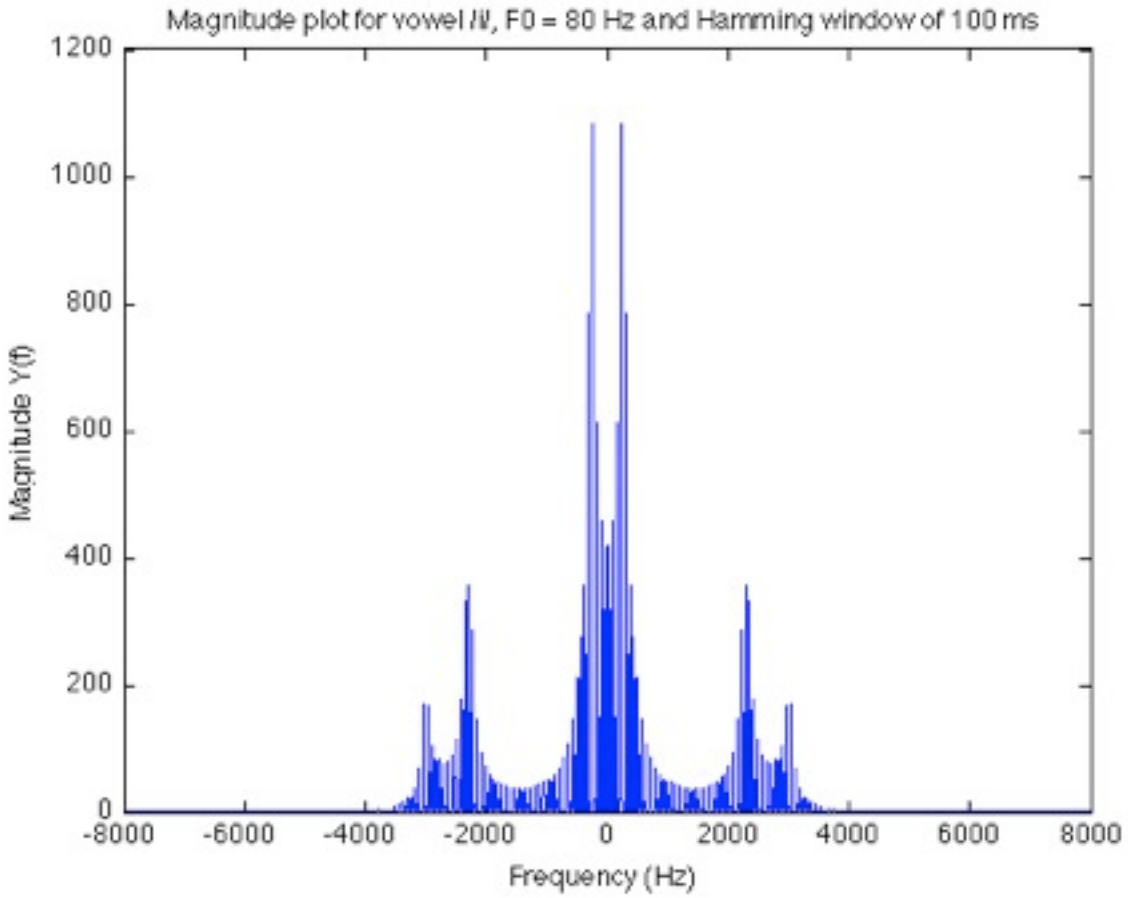
Using rectangular window of 100 ms:



Using Hamming window of 40 ms:



Using Hamming window of 100 ms:



Comment:

Rectangular windows of 40 ms and 100 ms both give the same kind of output. Hamming windows give better plots because in the frequency domain they have smaller side-lobes due to their smoother ends. Hamming windows of 40 ms give higher peaks than Hamming windows of 100 ms. Windows with longer intervals give better estimates because their frequency spectrum is narrower.