```
% Problem 1a

% Area of ceiling = 25 x 40 = 1000 m2
% Area of walls = 4 x 25 x 10 = 1000 m2
% Area of floor = 25 x 40 = 1000 m2
% T_60 = -2*ln(0.0001)*V/(gc* sum(a_i*S_i))
% T_60 = -0.161*V/sum(a_i*S_i)

freq = [250 1000 4000];
s_plywood = [0.22 0.22 0.11];
s_glass = [0.25 0.12 0.04];
s_marble = [0.01 0.01 0.02];
s_carpet = [0.06 0.37 0.65];
alpha_air = [1/130 1/30 1/5];

width = 25;
length = 40;
height = 10;

area_ceiling = width*length;
area_floor = area_ceiling;
area_wall = 2*height*(length+width);

area_plywood = area_ceiling + area_floor/2 + area_wall/10;
area_glass = area_wall/10;
area_marble = 8/10*area_wall + area_floor/2;

volume = 25*10*40;
disp('');
for i=1:3
    T_60(i) = 0.161*volume/( (area_plywood*s_plywood(i) + area_glass*s_glass(i) +
    out = sprintf('Frequency: %i Hz T_60 = %.2f s',freq(i),T_60(i));
    disp(out);
end
disp(' ');
disp('if the church is half its size, assume half the length');
disp(' ');
width = 25;
length = 40/2;
height = 10;


area_ceiling = width*length;
area_floor = area_ceiling;
area_wall = 2*height*(length+width);


area_plywood = area_ceiling + area_floor/2 + area_wall/10;
area_glass = area_wall/10;
area_marble = 8/10*area_wall + area_floor/2;

volume = 25*10*40;

for i=1:3
    T_60(i) = 0.161*volume/( (area_plywood*s_plywood(i) + area_glass*s_glass(i) +
    out = sprintf('Frequency: %i Hz T_60 = %.2f s',freq(i),T_60(i));
    disp(out);
end

disp(' ');
disp('if church were carpeted');
disp(' ');
```

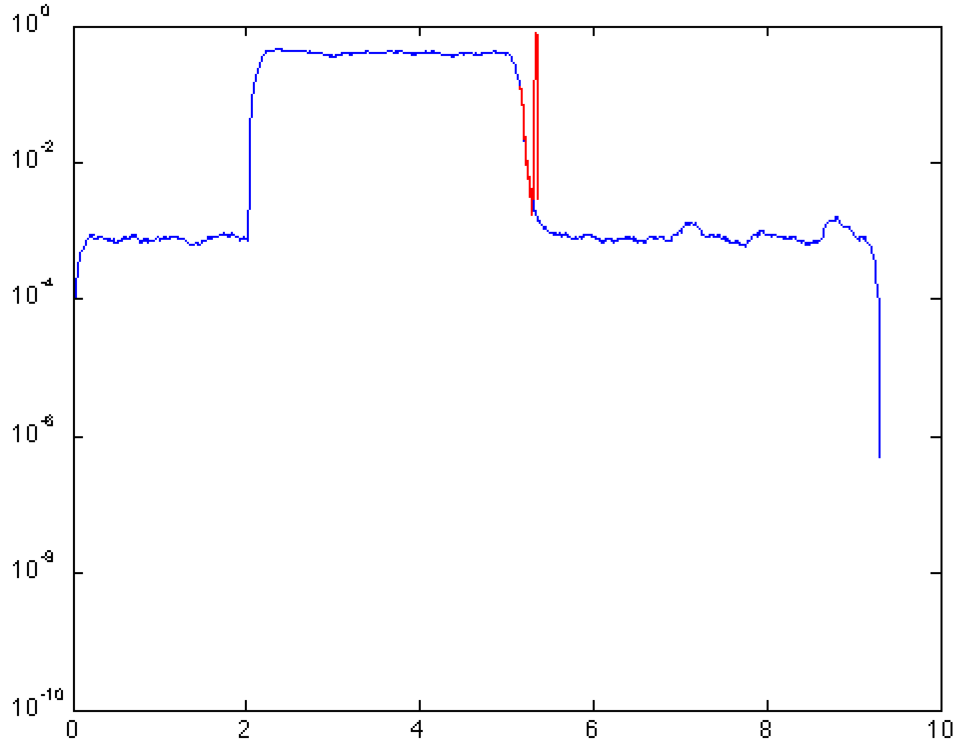
```
width = 25;
length = 40;
height = 10;

area_ceiling = width*length;
area_floor = area_ceiling;
area_wall = 2*height*(length+width);

area_plywood = area_ceiling + area_wall/10;
area_glass = area_wall/10;
area_marble = 8/10*area_wall;
area_carpet = area_floor;

volume = 25*10*40;

for i=1:3
    T_60(i) = 0.161*volume/( (area_plywood*s_plywood(i) + area_glass*s_glass(i) +
    out = sprintf('Frequency: %i Hz T_60 = %.2f s',freq(i),T_60(i));
    disp(out);
end
```

```
Frequency: 250 Hz T_60 = 3.33 s
Frequency: 1000 Hz T_60 = 2.23 s
Frequency: 4000 Hz T_60 = 0.73 s

if the church is half its size, assume half the length

Frequency: 250 Hz T_60 = 5.48 s
Frequency: 1000 Hz T_60 = 2.99 s
Frequency: 4000 Hz T_60 = 0.76 s

if church were carpeted

Frequency: 250 Hz T_60 = 4.37 s
Frequency: 1000 Hz T_60 = 2.65 s
Frequency: 4000 Hz T_60 = 0.75 s
```

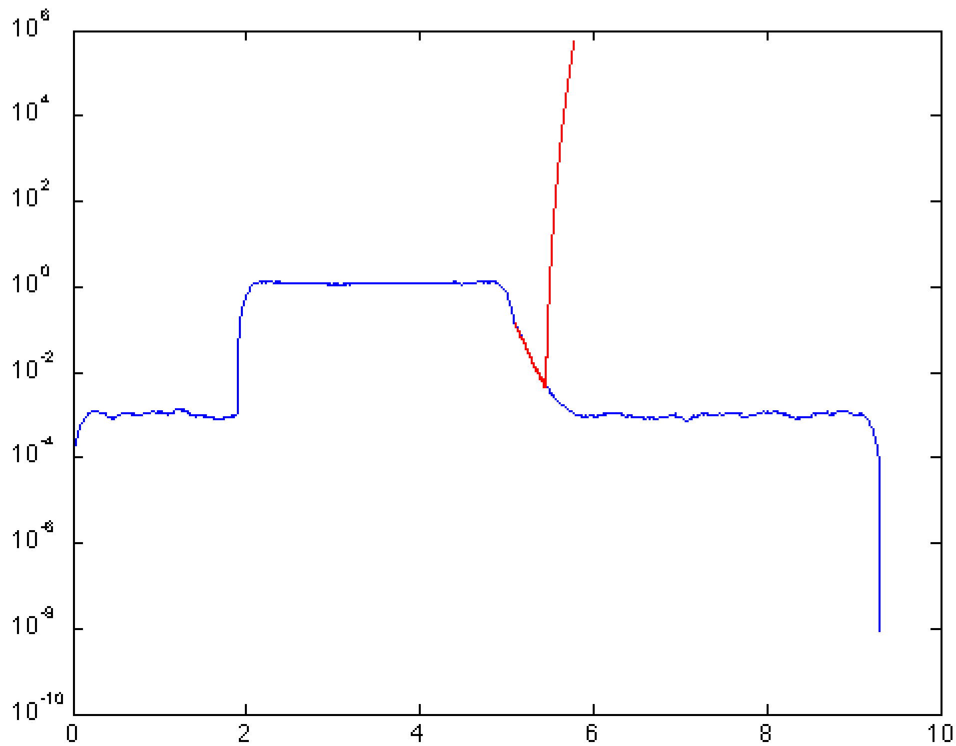
Published with MATLAB® 7.10

```
clear all;
% Problem 2


% Read in file
[x,fs] = wavread('filtered_noise_response.wav');
n = length(x);

% Split into different bursts
for i=1:8
    y(i,:) = x(1+(i-1)/8*n:i/8*n);
    y2(i,:) = y(i,:).*y(i,:);
    e(i,:) = conv(y2(i,:),boxcar(8000));
%      figure();
%      semilogy(e(i,:));
end

% for first burst
start =  [227300 224200 227000 220300 233000 216600 206600 198000 ];
finish = [233700 239500 271400 297000 340200 330900 276800 259800 ];

for i=1:length(start)
    X = start(i):finish(i);
    X_new = start(i):start(i)+2*(finish(i)-start(i));
    p = polyfit(X,e(i,X),50);
    Y = polyval(p,X_new);
    figure();
    time = (1:length(e(i,:)) )/fs;
    semilogy(time,e(i,:));
    hold on;
    plot(X_new/fs,Y,'r');
end


% T60 by inspection of graphs
freq = [16000 8000 4000 2000 1000 500 250 125];
T60 = [0.3 0.7 1.15 1.25 1.65 1.67 2.0 1.8];

for i=1:8
    out = sprintf('Frequency: %i Hz T60 = %.2f s',freq(i),T60(i));
    disp(out)
end


Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
Warning: Negative data ignored
Warning: Negative data ignored
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
```

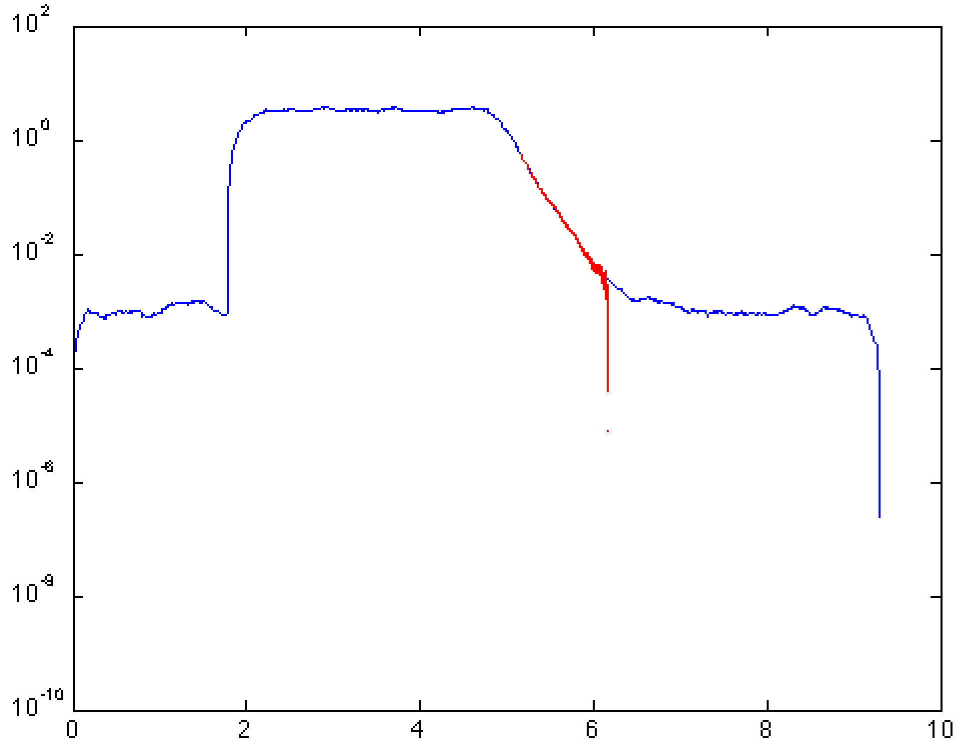
```
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
Warning: Negative data ignored
Warning: Negative data ignored
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
Warning: Negative data ignored
Warning: Negative data ignored
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
Warning: Negative data ignored
Warning: Negative data ignored
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
Warning: Negative data ignored
Warning: Negative data ignored
Warning: Polynomial is badly conditioned. Add points with distinct X
         values, reduce the degree of the polynomial, or try centering
         and scaling as described in HELP POLYFIT.
Warning: Negative data ignored
Warning: Negative data ignored
Frequency: 16000 Hz T60 = 0.30 s
Frequency: 8000 Hz T60 = 0.70 s
Frequency: 4000 Hz T60 = 1.15 s
Frequency: 2000 Hz T60 = 1.25 s
Frequency: 1000 Hz T60 = 1.65 s
Frequency: 500 Hz T60 = 1.67 s
Frequency: 250 Hz T60 = 2.00 s
Frequency: 125 Hz T60 = 1.80 s
```



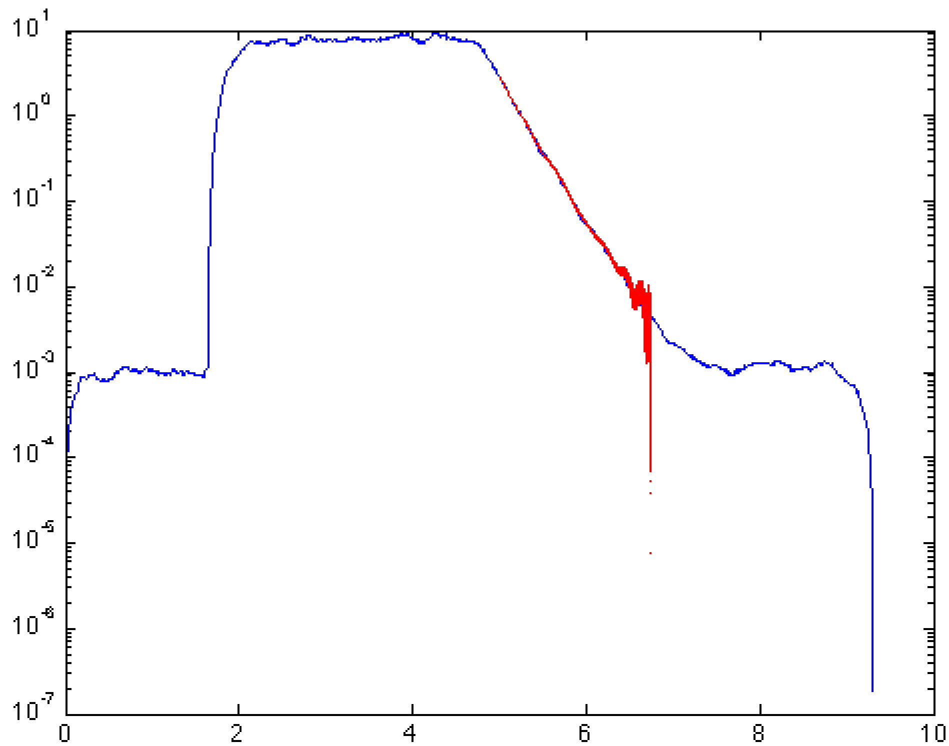
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
2
4
6
8
10



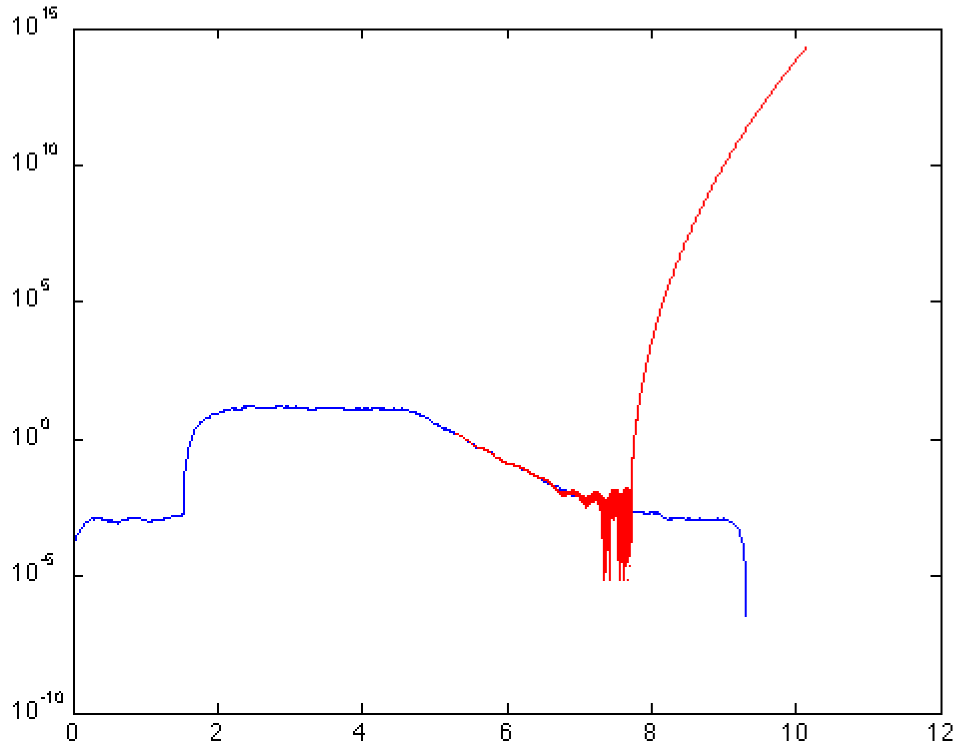
10^{6}
10^{4}
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
2
4
6
8
10



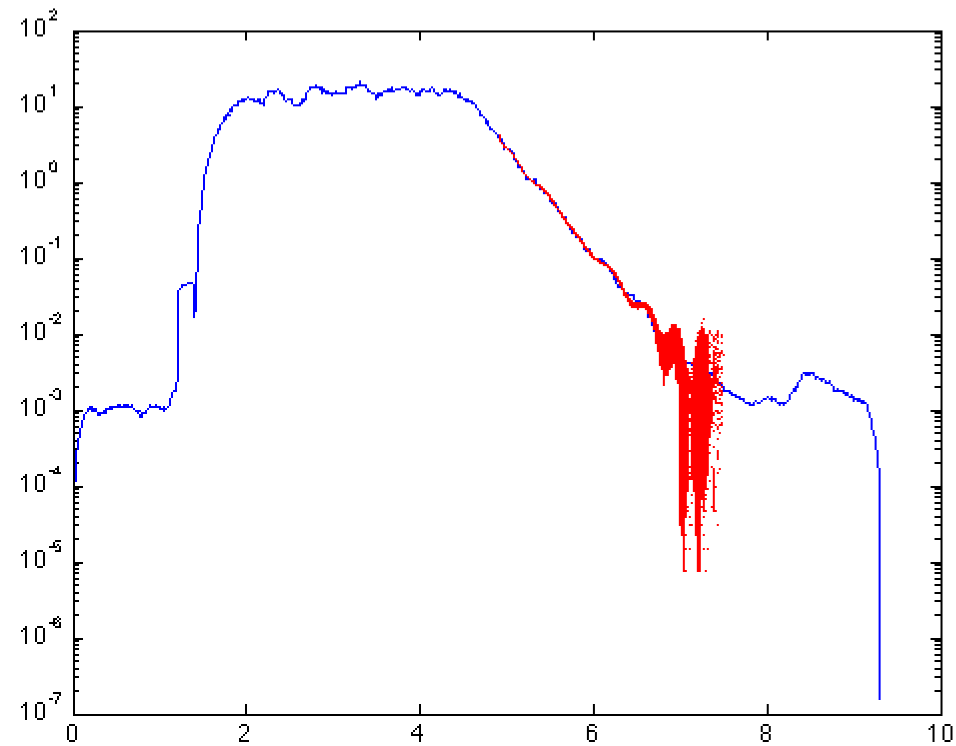
10^2
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
0
2
4
6
8
10



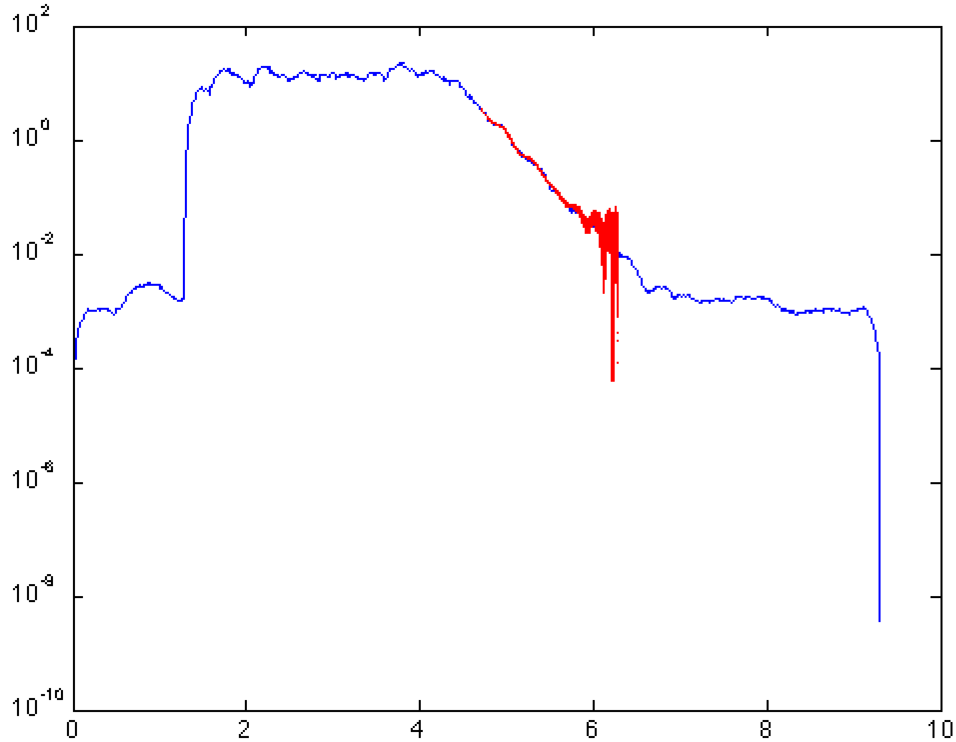
10^1
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
10^-6
10^-7
0
2
4
6
8
10



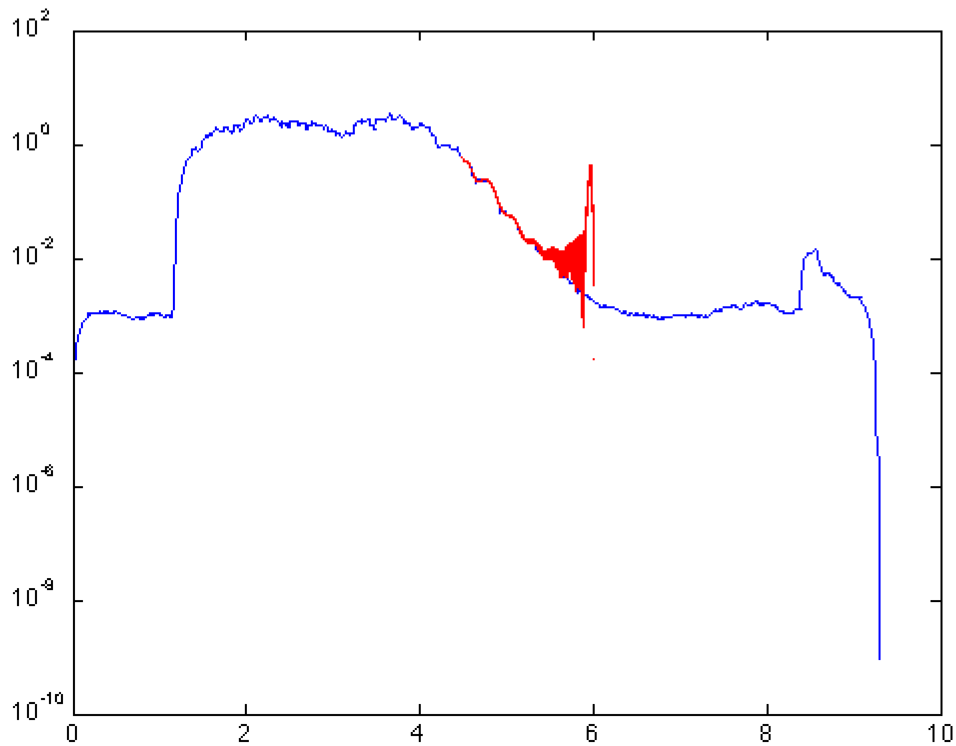
10^{15}
10^{10}
10^{5}
10^{0}
10^{-5}
10^{-10}
0
2
4
6
8
10
12



10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
2
4
6
8
10



10^2
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
0
2
4
6
8
10



10^2
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
0
2
4
6
8
10

Published with MATLAB® 7.10

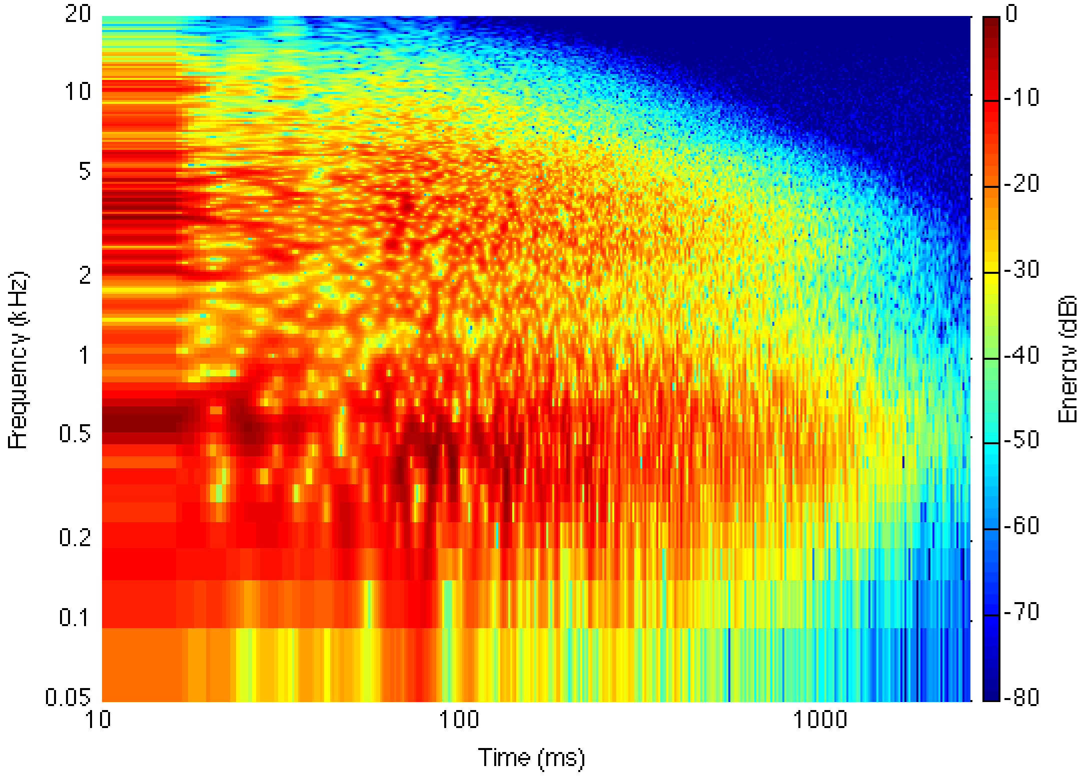
```
clear all;
% problem 3a

[y,fs] = wavread('balloon_Kevin.wav');
n = 0:length(y)-1;
n = n./fs;

impulse_start = 0.3849*fs;

y_25s = y(impulse_start:impulse_start+2.5*fs);
y_300ms = y(impulse_start:impulse_start+0.3*fs);

figure();
ftgram(y_25s, fs, 'rir', 'nskip', 32, 'waveform', false, 'dbrange', 80);

figure();
ftgram(y_300ms, fs, 'rir', 'nskip', 32, 'waveform', false, 'dbrange', 80);

% problem 3b

w_c = 125*2.^[0:7];
w_c = w_c/fs*2;

[b,a] = butter(3,(w_c(1)*w_c(2))^0.5,'low');
figure();
freqz(b,a);
title(w_c(1)*fs/2);

for i=2:length(w_c)-1
    [b, a] = butter(3,[(w_c(i-1)*w_c(i))^0.5 (w_c(i)*w_c(i+1))^0.5]);
    figure();
    freqz(b,a);
    title(w_c(i)*fs/2);
end

[b,a] = butter(3,(w_c(7)*w_c(8))^0.5,'high');
figure();
freqz(b,a);
title(w_c(8)*fs/2);

% problem 3c

freq =       [125     250     500     1000     2000     4000     8000     16000];

% filter 125Hz
[b,a] = butter(3,(w_c(1)*w_c(2))^0.5,'low');
f = filtfilt(b, a, y);
f2 = abs(f(:,1).*f(:,2));
e = conv(f2,boxcar(5000));
figure();
plot([1:length(e)]/fs,20*log10(e));
ylim([-150 50]);
xlim([0 5]);
hold all;
T60_Kevin(1) = lateDecayT60(e,fs);
decayRate_Kevin(1) = lateDecayRolloff(e,fs);



% filter 250-8k
for i=2:length(w_c)-1
    [b, a] = butter(3,[(w_c(i-1)*w_c(i))^0.5 (w_c(i)*w_c(i+1))^0.5]);
    f = filtfilt(b, a, y);
```

```
    f2 = abs(f(:,1).*f(:,2));
    e = conv(f2,boxcar(5000));
    plot([1:length(e)]/fs,20*log10(e));
    T60_Kevin(i) = lateDecayT60(e,fs);
    decayRate_Kevin(i) = lateDecayRolloff(e,fs);
end

% filter 16k
[b,a] = butter(3,(w_c(7)*w_c(8))^0.5,'high');
f = filtfilt(b, a, y);
f2 = abs(f(:,1).*f(:,2));
e = conv(f2,boxcar(5000));
plot([1:length(e)]/fs,20*log10(e));
legend('125Hz','250Hz','500Hz','1000Hz','2000Hz','4000Hz','8000Hz','16000Hz');
title('Kevins Baloon');
T60_Kevin(8) = lateDecayT60(e,fs);
decayRate_Kevin(8) = lateDecayRolloff(e,fs);

figure();
semilogx(freq,T60_Kevin);
xlabel('Frequency (Hz)');
ylabel('T60(s)');
title('Kevins Baloon');
figure();
semilogx(freq,decayRate_Kevin,'r');
xlabel('Frequency (Hz)');
ylabel('Decay Rate');
title('Kevins Baloon');


% repeat for different balloon pop



[y,fs] = wavread('balloon_Haying.wav');

% filter 125Hz
[b,a] = butter(3,(w_c(1)*w_c(2))^0.5,'low');
f = filtfilt(b, a, y);
f2 = abs(f(:,1).*f(:,2));
e = conv(f2,boxcar(5000));
figure();
plot([1:length(e)]/fs,20*log10(e));
ylim([-150 50]);
xlim([0 5]);
hold all;
T60_Haying(1) = lateDecayT60(e,fs);
decayRate_Haying(1) = lateDecayRolloff(e,fs);


% filter 250-8k
for i=2:length(w_c)-1
    [b, a] = butter(3,[(w_c(i-1)*w_c(i))^0.5 (w_c(i)*w_c(i+1))^0.5]);
    f = filtfilt(b, a, y);
    f2 = abs(f(:,1).*f(:,2));
    e = conv(f2,boxcar(5000));
    plot([1:length(e)]/fs,20*log10(e));
    T60_Haying(i) = lateDecayT60(e,fs);
    decayRate_Haying(i) = lateDecayRolloff(e,fs);
end

% filter 16k
[b,a] = butter(3,(w_c(7)*w_c(8))^0.5,'high');
f = filtfilt(b, a, y);
f2 = abs(f(:,1).*f(:,2));
```

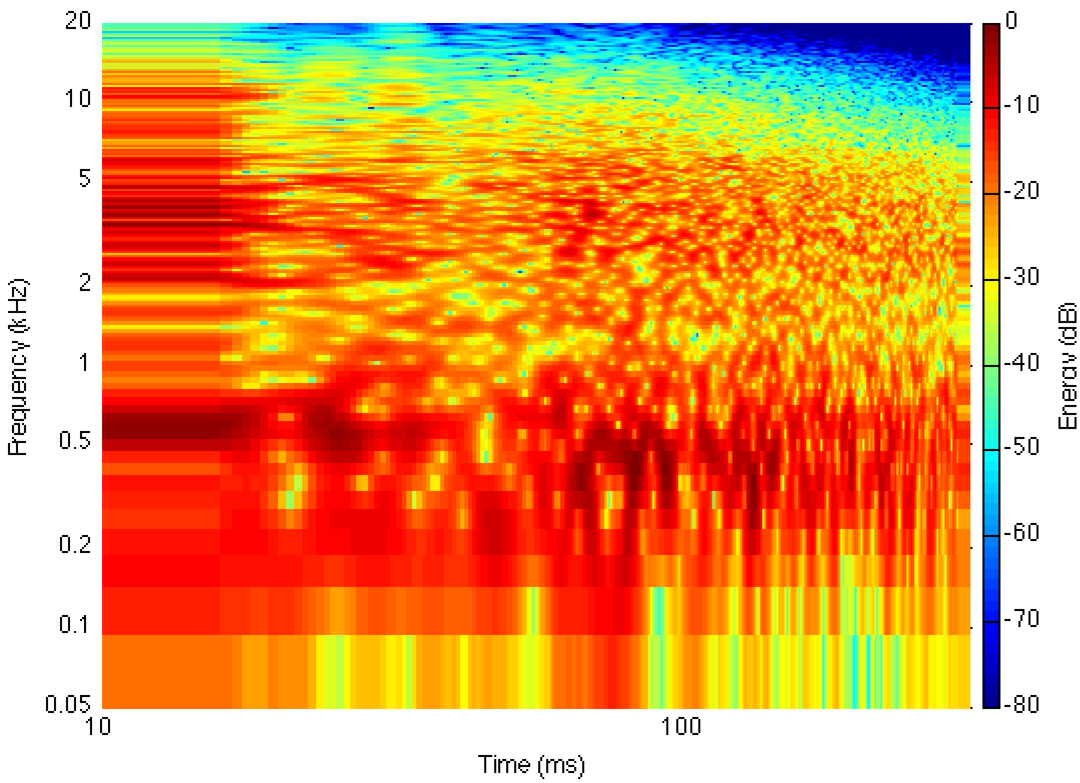
```
e = conv(f2,boxcar(5000));
plot([1:length(e)]/fs,20*log10(e));
legend('125Hz','250Hz','500Hz','1000Hz','2000Hz','4000Hz','8000Hz','16000Hz');
title('Hayings Baloon');
T60_Haying(8) = lateDecayT60(e,fs);
decayRate_Haying(8) = lateDecayRolloff(e,fs);


figure();
semilogx(freq,T60_Haying);
xlabel('Frequency (Hz)');
ylabel('T60(s)');
title('Hayings Baloon');
figure();
semilogx(freq,decayRate_Haying,'r');
xlabel('Frequency (Hz)');
ylabel('Decay Rate');
title('Hayings Baloon');

% Comparing the T-60 of the 2 balloons, we find that both values are
% following the same pattern, confirming that they were both in very
% similar spaces



% Problem 3d

% by inspection
freq = [125    250    500    1000    2000    4000    8000    16000 ];
SNR  = [1.39e4 1.58e5 9.24e5 4.43e5  4.77e5  1.07e6  2.43e5  5.13e4  ]
```

Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index

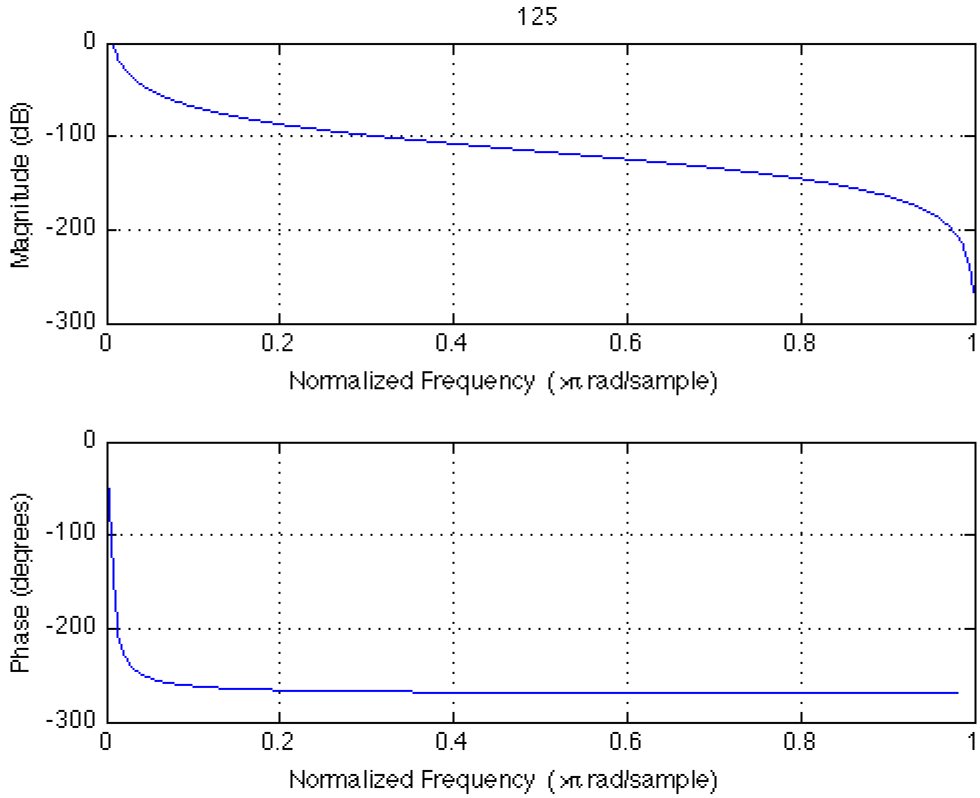
```
SNR =

  Columns 1 through 5

       13900       158000       924000       443000       477000

  Columns 6 through 8

     1070000       243000        51300
```

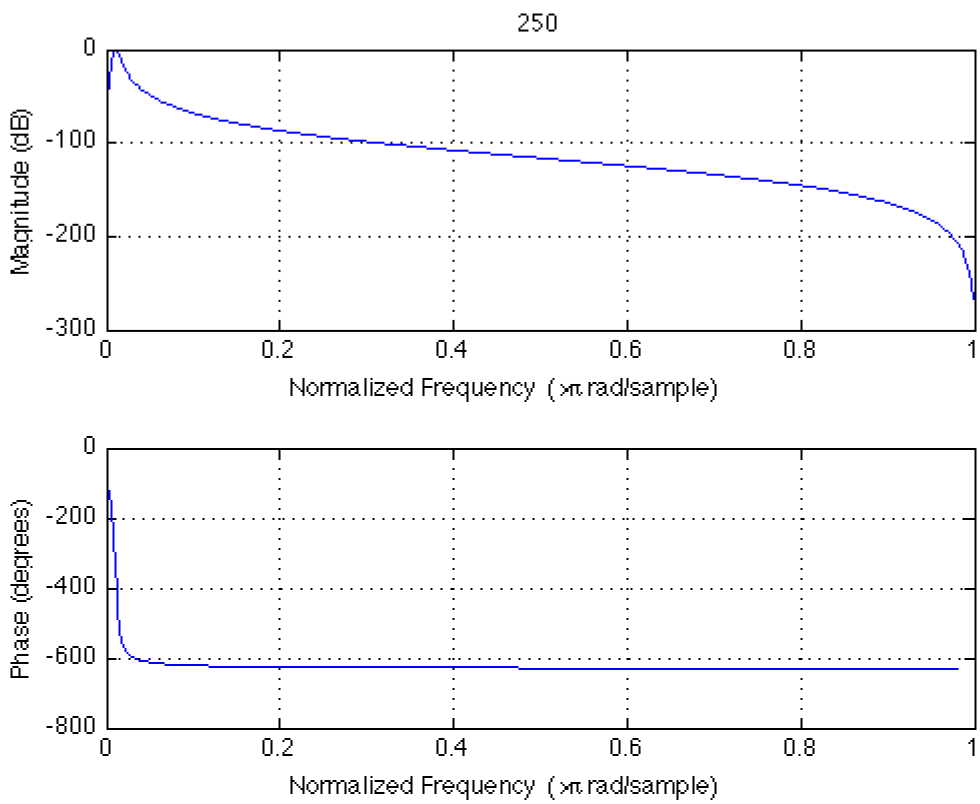


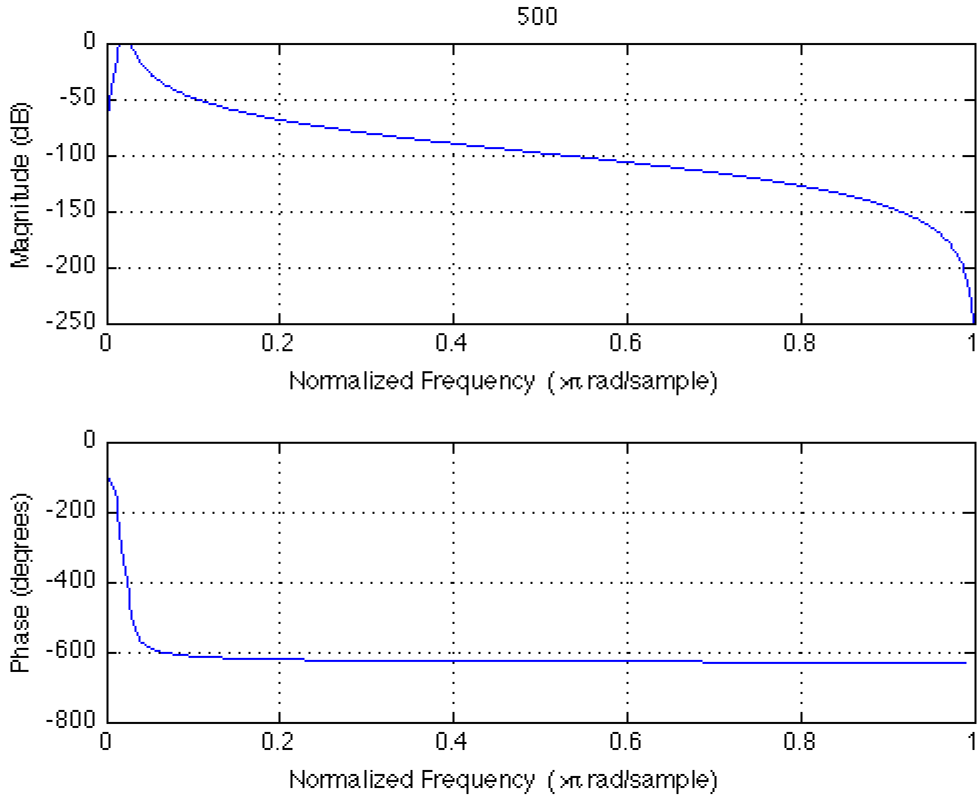
Frequency (kHz)
Time (ms)
Energy (dB)

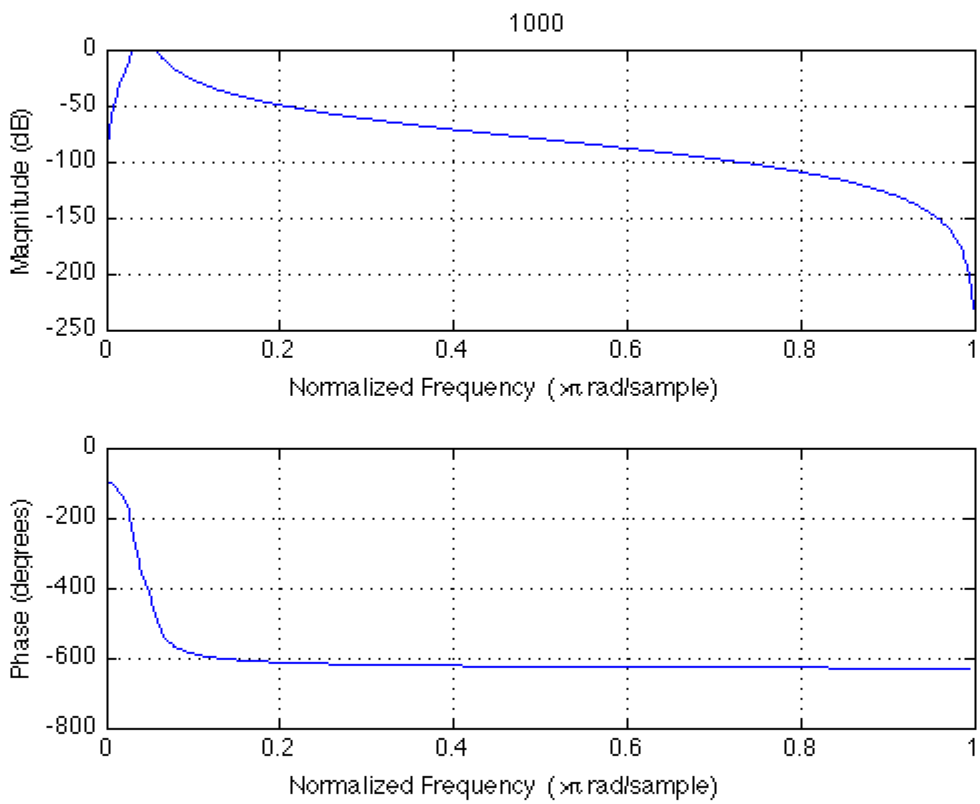


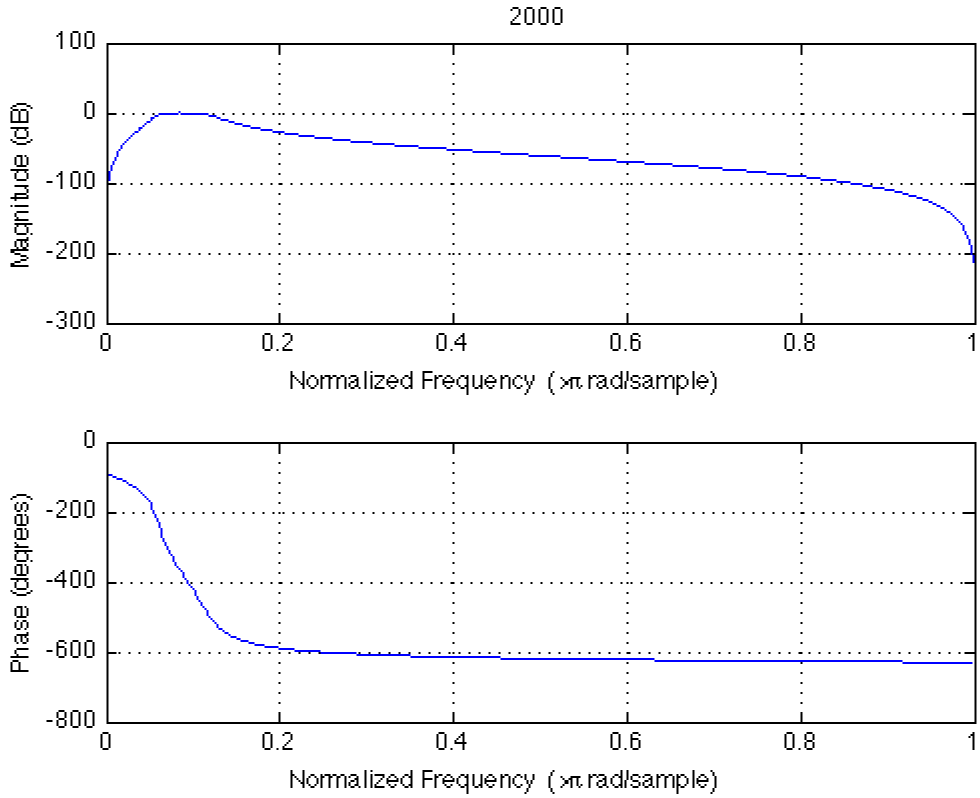
Frequency (kHz)
Time (ms)
Energy (dB)

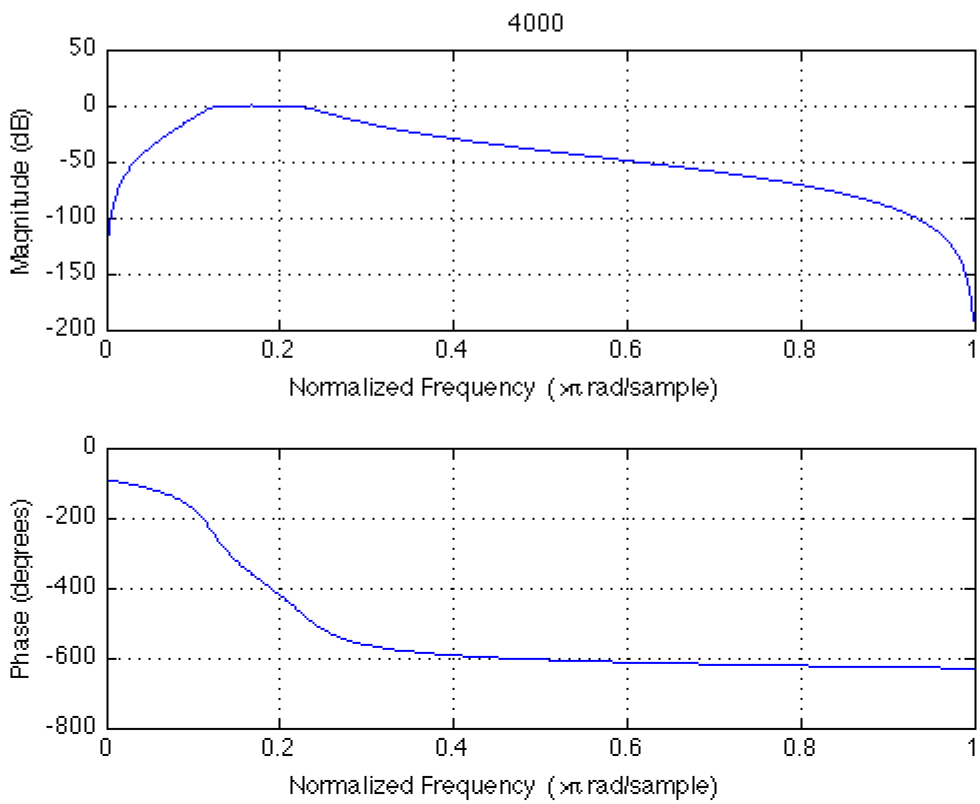

125
Magnitude (dB)
0
-100
-200
-300
Phase (degrees)
0
-100
-200
-300
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)
Normalized Frequency (×π rad/sample)

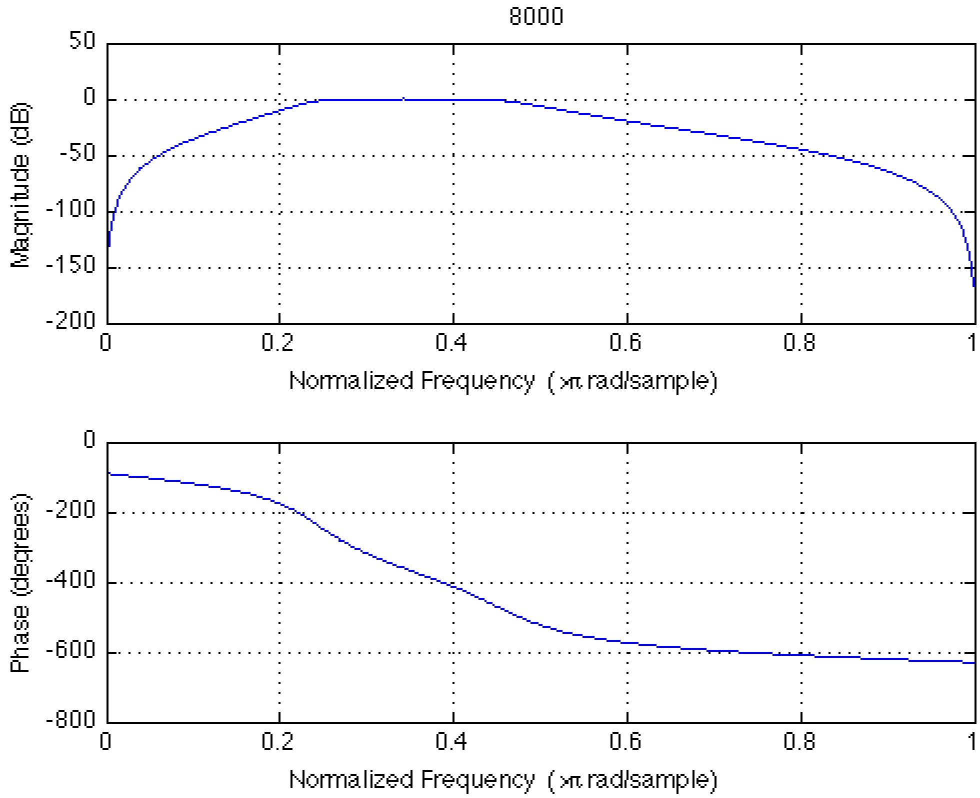

250
Magnitude (dB)
0
-100
-200
-300
Phase (degrees)
0
-200
-400
-600
-800
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)
Normalized Frequency (×π rad/sample)

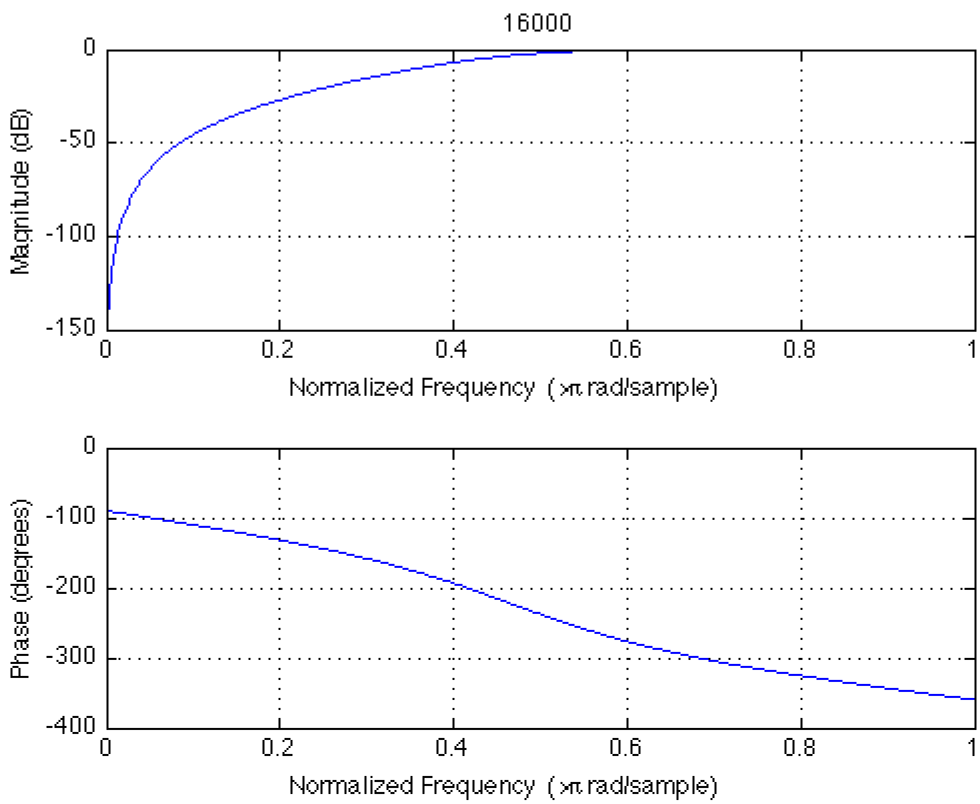

500
Magnitude (dB)
0
-50
-100
-150
-200
-250
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-200
-400
-600
-800
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)

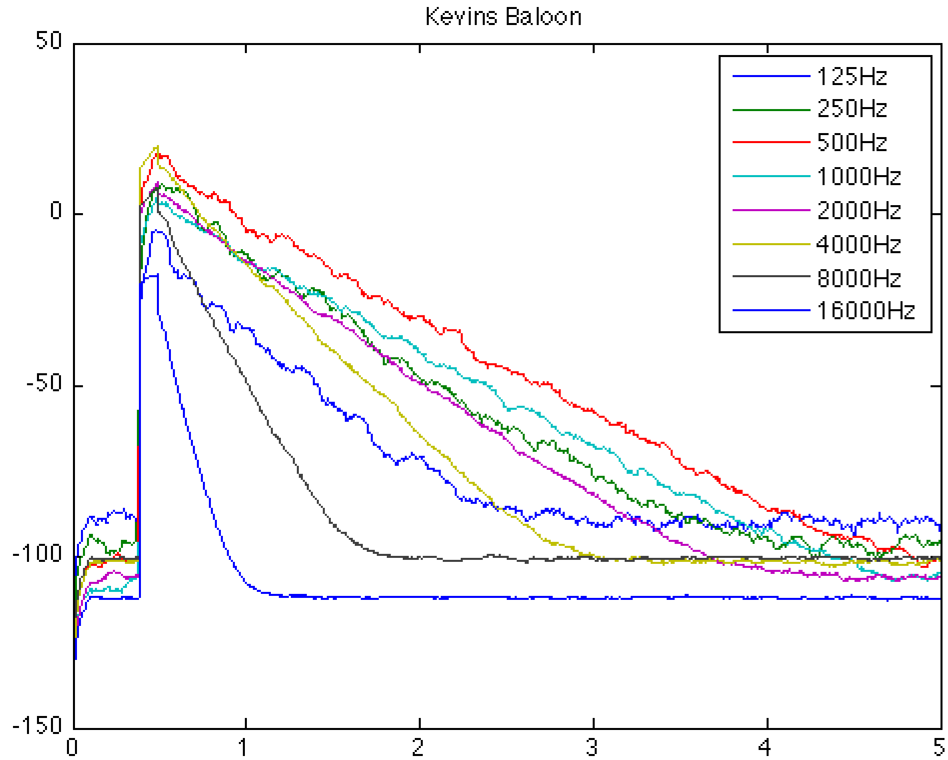

1000
Magnitude (dB)
0
-50
-100
-150
-200
-250
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-200
-400
-600
-800
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)

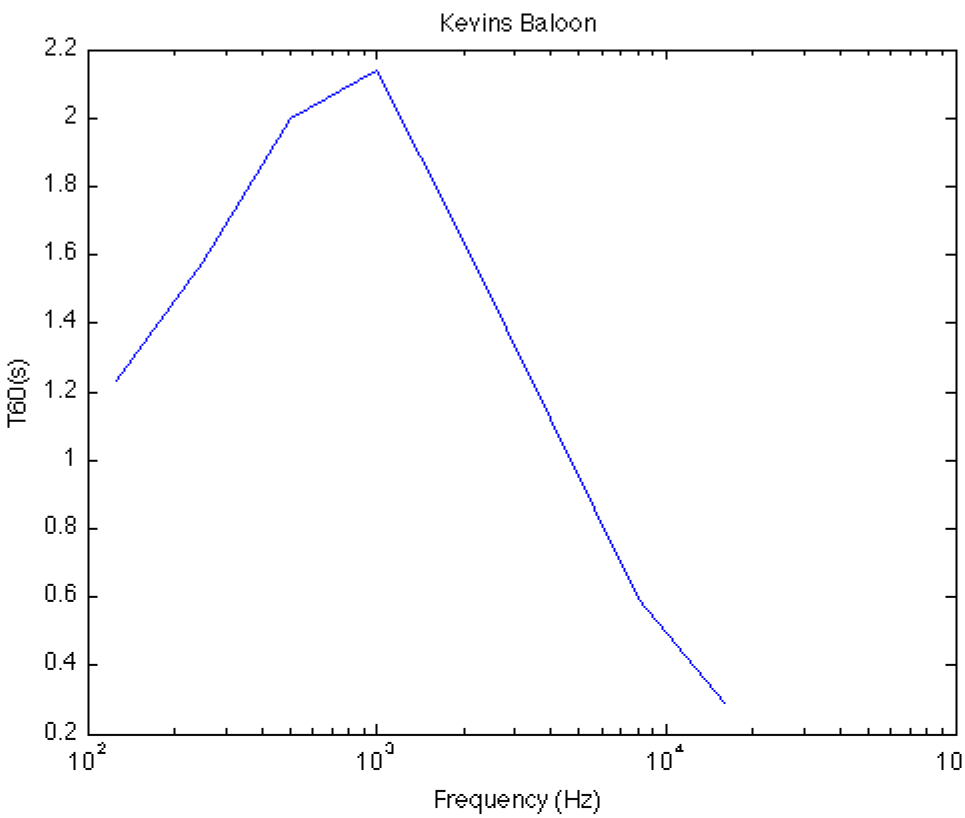

2000
Magnitude (dB)
100
0
-100
-200
-300
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-200
-400
-600
-800
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)

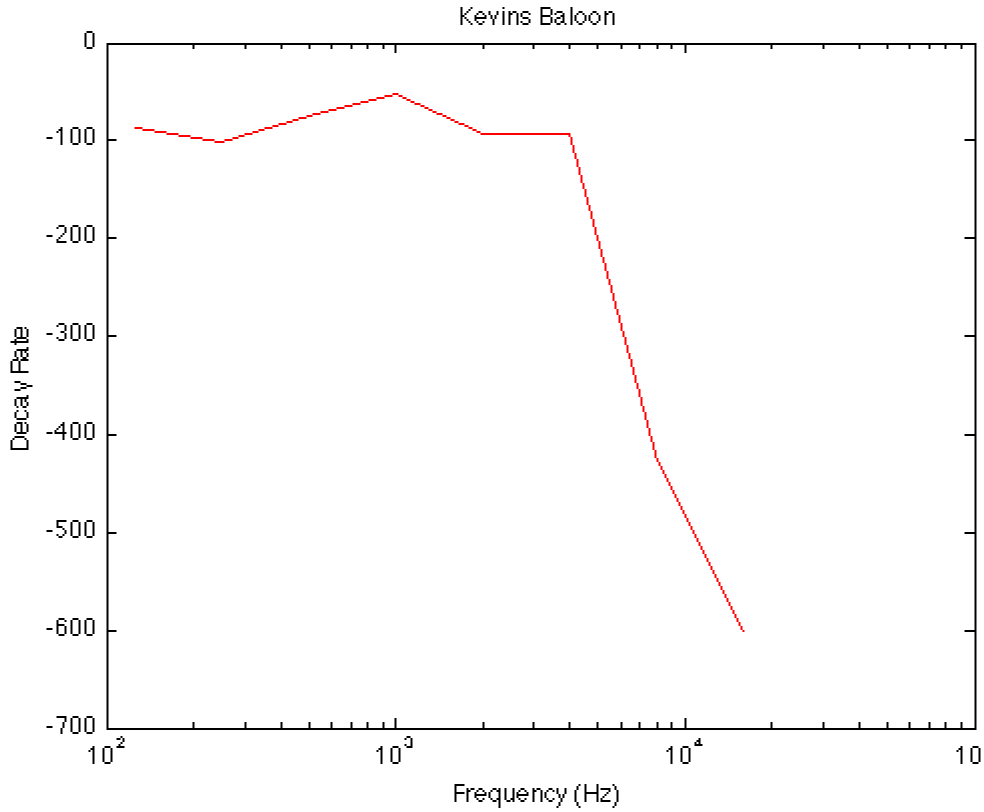

4000
Magnitude (dB)
50
0
-50
-100
-150
-200
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-200
-400
-600
-800
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)

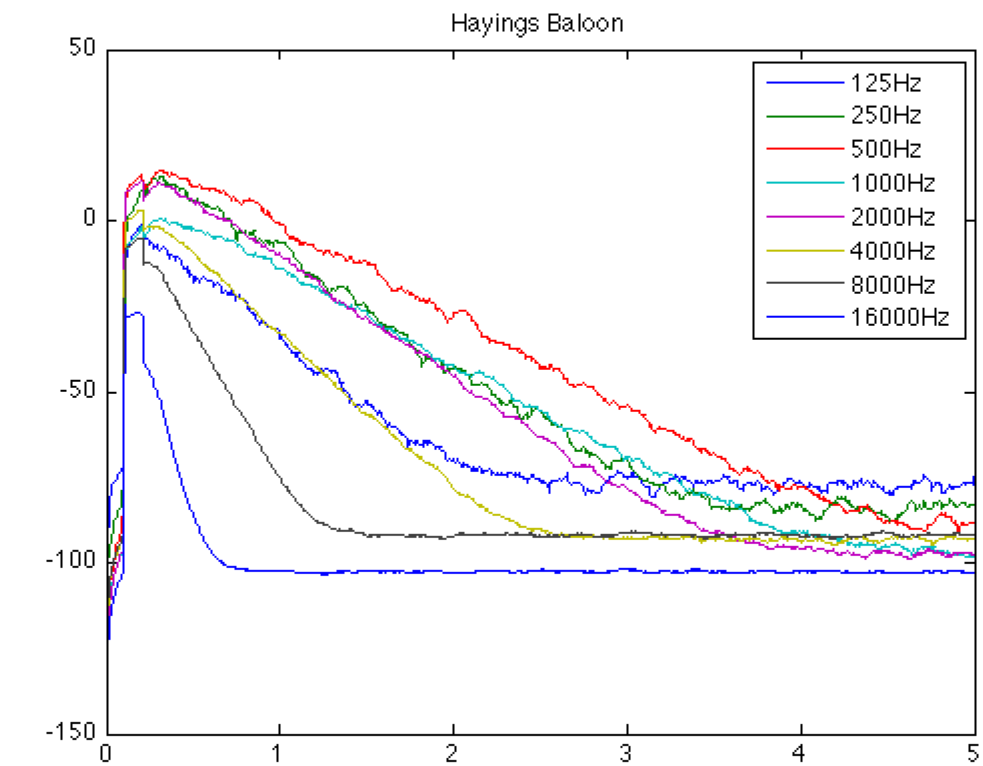

8000
Magnitude (dB)
50
0
-50
-100
-150
-200
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-200
-400
-600
-800
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)

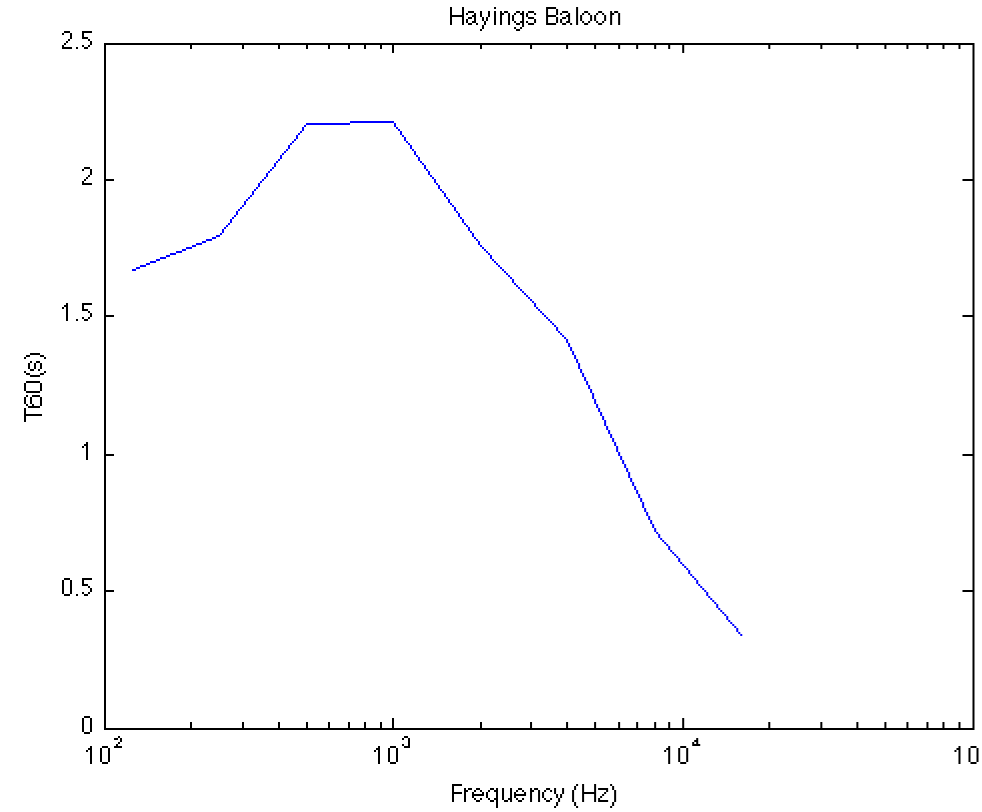

16000
Magnitude (dB)
0
-50
-100
-150
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-100
-200
-300
-400
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π rad/sample)

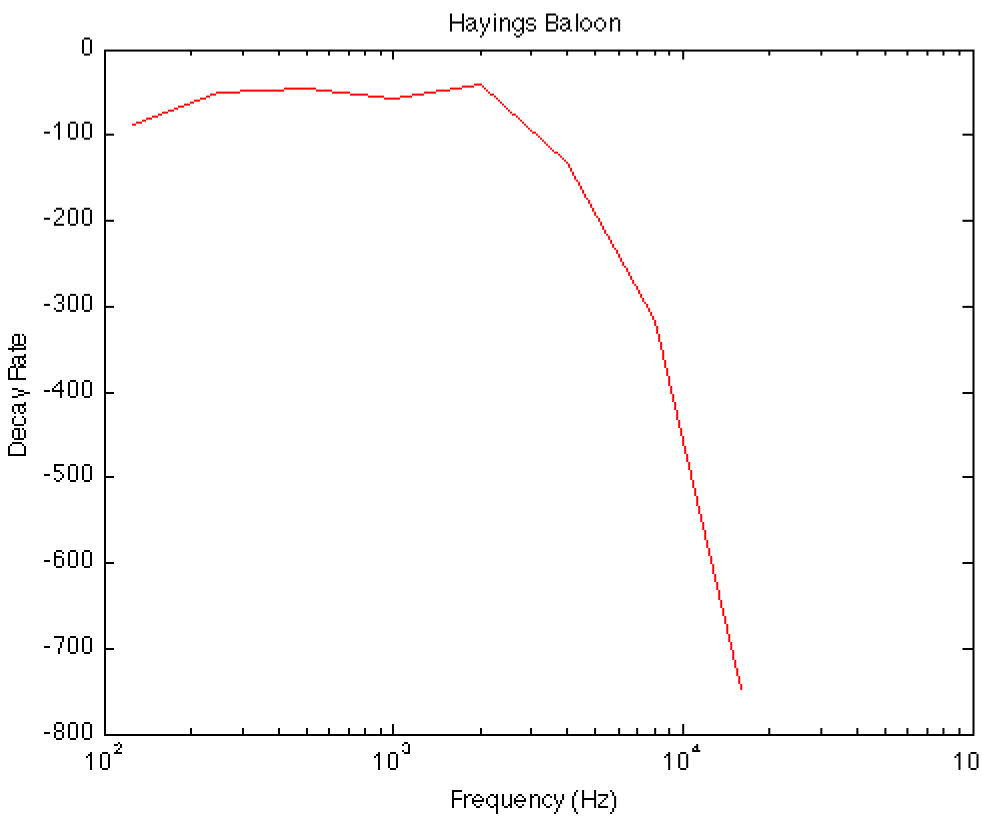

Kevins Baloon
125Hz
250Hz
500Hz
1000Hz
2000Hz
4000Hz
8000Hz
16000Hz
50
0
-50
-100
-150
0
1
2
3
4
5


Kevins Baloon
T60(s)
Frequency (Hz)
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
10^2
10^3
10^4
10^5


Kevins Baloon
Decay Rate
0
-100
-200
-300
-400
-500
-600
-700
10^2
10^3
10^4
10^5
Frequency (Hz)


Hayings Baloon
50
0
-50
-100
-150
0
1
2
3
4
5
125Hz
250Hz
500Hz
1000Hz
2000Hz
4000Hz
8000Hz
16000Hz


Hayings Baloon
T60(s)
Frequency (Hz)
2.5
2
1.5
1
0.5
0
10^2
10^3
10^4
10^5


Hayings Baloon
Decay Rate
Frequency (Hz)
0
-100
-200
-300
-400
-500
-600
-700
-800
10^2
10^3
10^4
10^5

Published with MATLAB® 7.10

```
function [ slope ] = lateDecaySlope( signal,fs )
%FINDT60 Find T60 of an input signal's smoothed energy envelope
%   For input, give signal and fs. It outputs the time taken to decay to
%   -60dB of the maximum.


% find maximum
[signal_max, t_max] = max(signal);


% find 60 dB decay
cropped_signal = signal(t_max:length(signal));

[signal_5,t_5] = min(abs(cropped_signal-signal_max*0.5623));

[signal_65, t_65] = min(abs(cropped_signal-signal_max*5.6234e-04));


% scale to seconds
slope = 20*log(signal_5/signal_65) / (t_5-t_65) * fs;

end
```

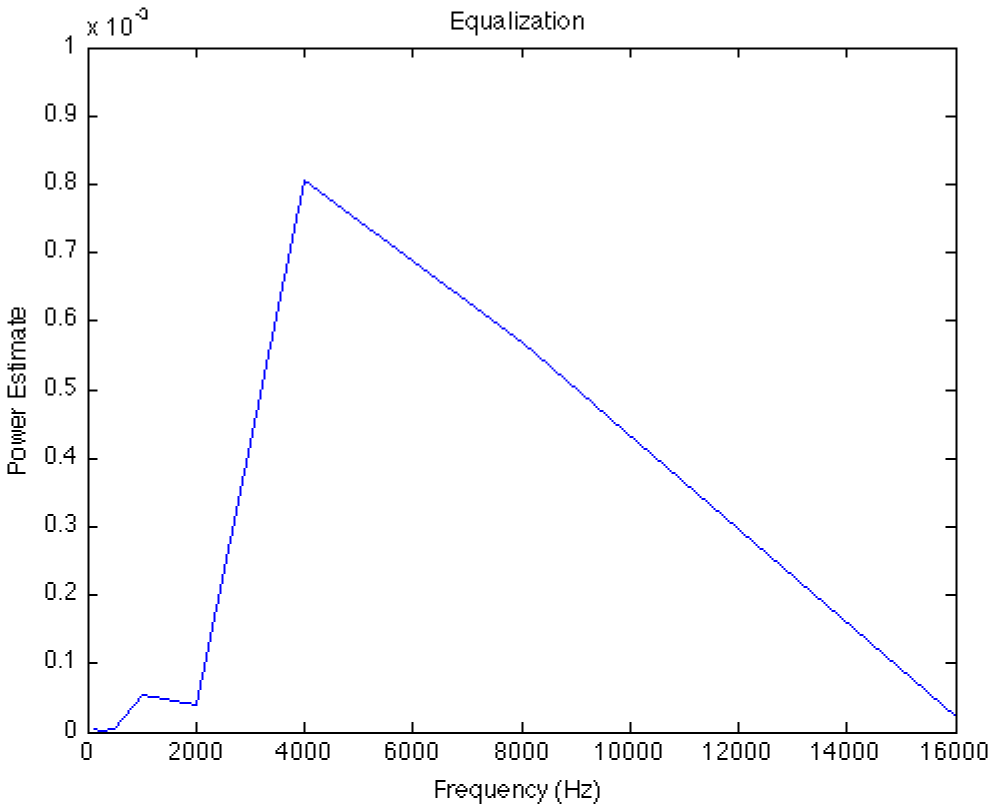
Published with MATLAB® 7.10

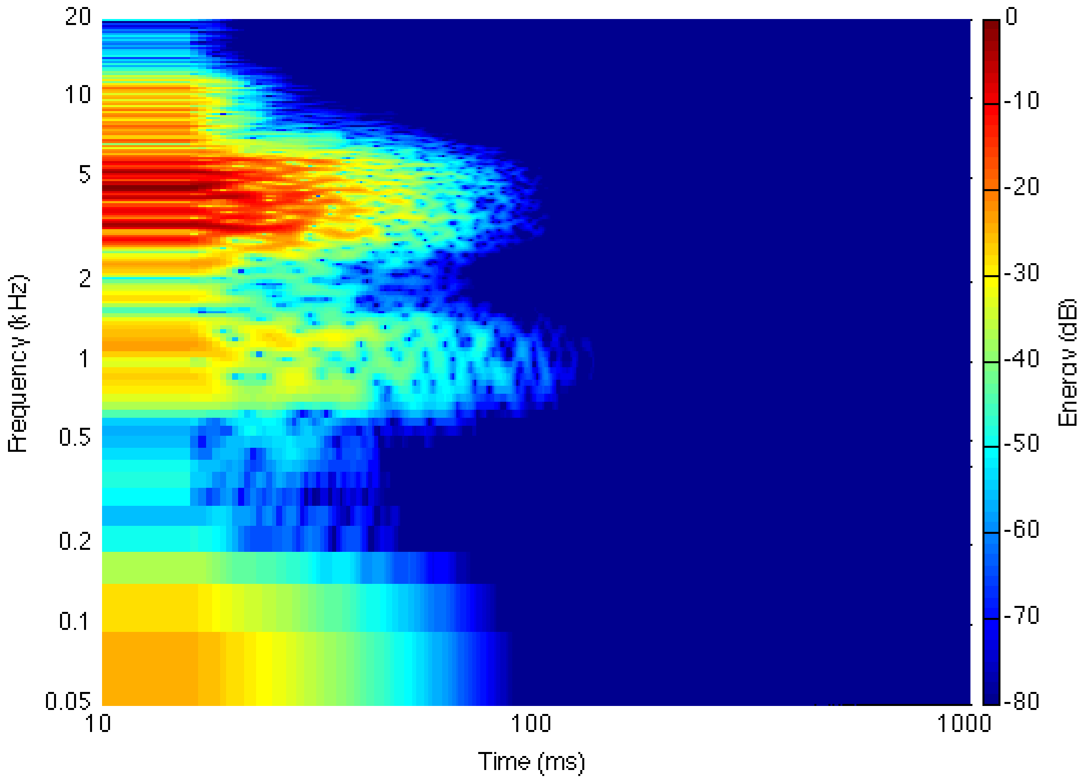
```matlab
function [ t60 ] = lateDecayT60( signal,fs )
%FINDT60 Find T60 of an input signal's smoothed energy envelope
%   For input, give signal and fs. It outputs the time taken to decay to
%   -60dB of the maximum.


% find maximum
[signal_max, t_max] = max(signal);


% find 60 dB decay
cropped_signal = signal(t_max:length(signal));

[signal_5,t_5] = min(abs(cropped_signal-signal_max*0.5623));

[signal_65, t_65] = min(abs(cropped_signal-signal_max*5.6234e-04));


% scale to seconds
t60 = (t_65 - t_5) / fs;

end
```

Published with MATLAB® 7.10

```
clear all;
% problem 4a

[y,fs] = wavread('balloon_Kevin.wav');

freq =       [125     250     500     1000    2000    4000    8000    16000];

w_c = 125*2.^[0:7];
w_c = w_c/fs*2;

% filter 125 Hz
[b,a] = butter(3,(w_c(1)*w_c(2))^0.5,'low');
f = filtfilt(b, a, y);
f2 = abs(f(:,1).*f(:,2));
powerEstimate(1) = findT5_level(f2);

% filter 250-8k
for i=2:length(w_c)-1
    [b, a] = butter(3,[(w_c(i-1)*w_c(i))^0.5 (w_c(i)*w_c(i+1))^0.5]);
    f = filtfilt(b, a, y);
    f2 = abs(f(:,1).*f(:,2));
    powerEstimate(i) = findT5_level(f2);
end

% filter 16k
[b,a] = butter(3,(w_c(7)*w_c(8))^0.5,'high');
f = filtfilt(b, a, y);
f2 = abs(f(:,1).*f(:,2));
powerEstimate(8) = findT5_level(f2);

figure();
plot(freq,powerEstimate);
xlabel('Frequency (Hz)');
ylabel('Power Estimate');
title('Equalization');

% problem 4b
% to synthesize the IR, we will take white noise, filter it through the
% different octave wide filters, multiply by the power of each band and
% decay exponentially with the decay rate


decayRate =[-86.1249 -100.6026  -73.9751  -52.3742  -93.4506  -93.8774 -427.3904 -


% filter 125 Hz
[b,a] = butter(3,(w_c(1)*w_c(2))^0.5,'low');
noise = rand(fs,1);
ir(1,:) = powerEstimate(1)*filtfilt(b, a, noise);
window(1,:) = exp(decayRate(1)*[1/fs:1/fs:1]);


% filter 250-8k
for i=2:length(w_c)-1
    [b, a] = butter(3,[(w_c(i-1)*w_c(i))^0.5 (w_c(i)*w_c(i+1))^0.5]);
    noise = rand(fs,1);
    ir(i,:) = powerEstimate(i)*filtfilt(b, a, noise);
    window(i,:) = exp(decayRate(i)*[1/fs:1/fs:1]);

end

% filter 16k
[b,a] = butter(3,(w_c(7)*w_c(8))^0.5,'high');
```

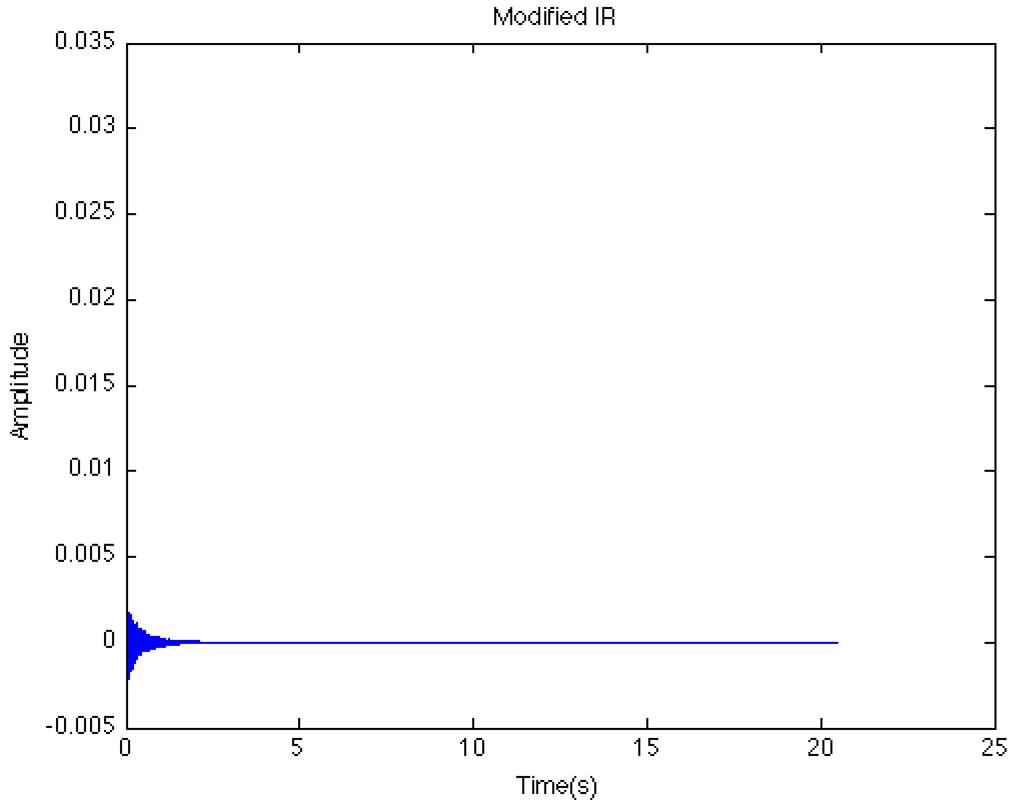
```
noise = rand(fs,1);
ir(8,:) = powerEstimate(8)*filtfilt(b, a, noise);
window(8,:) = exp(decayRate(8)*[1/fs:1/fs:1]);

final_ir=0;
for i=1:8
    final_ir = final_ir + ( ir(i,:).*window(i,:) );
end

figure();
ftgram(final_ir, fs, 'rir', 'nskip', 32, 'waveform', false, 'dbrange', 80);
```

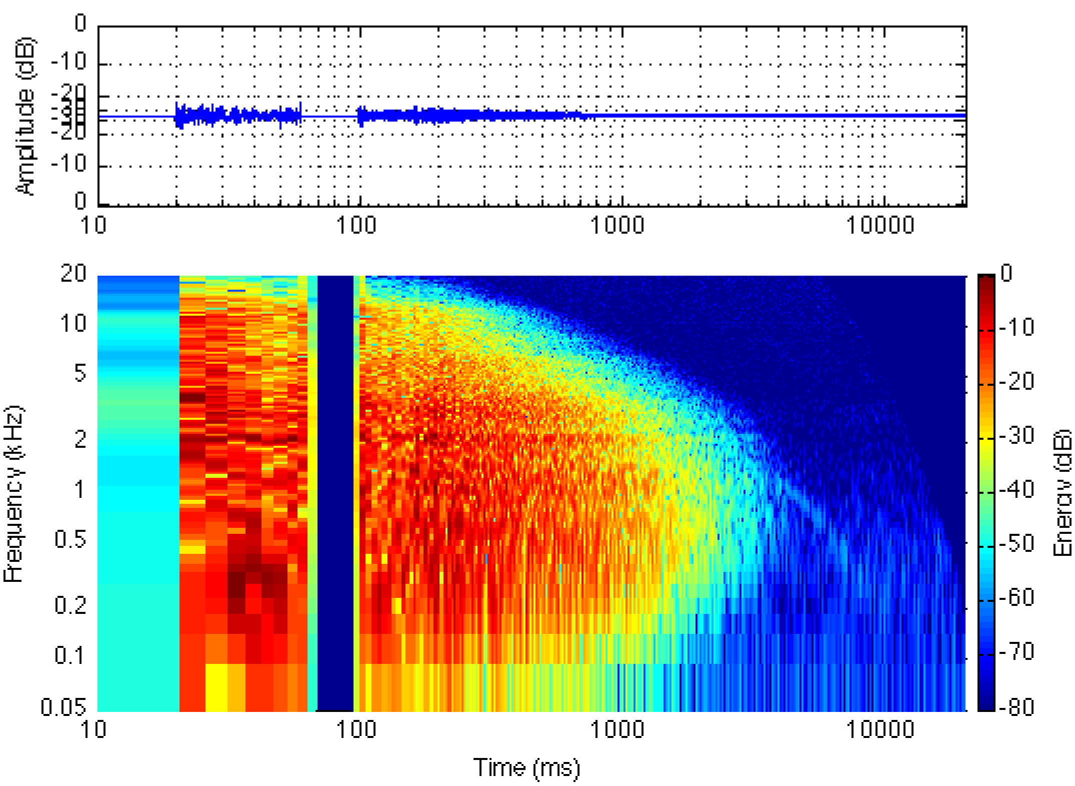




Published with MATLAB® 7.10

```
% Problem 5a

% exponential sweep IR estimate

[rs, fs] = wavread('ssx_20_response.wav');
[ss, fs] = wavread('ssx_48_20.wav');
cs = real(ifft(1./fft(ss)));
cs = cs(2^20-2^16+1:end);
irhatx = real(ifft((fft(cs, 2^21)*[1 1]).*fft(rs, 2^21)));
% ftgram(irhatx(2^20+[1:5*fs],1), fs, 'rir');

t = [1:length(irhatx)]/fs;

irhatx = irhatx(23.236*fs:length(irhatx),1);
t = [1:length(irhatx)]/fs;

% plot(t,irhatx);
[maxm,tr] = max(irhatx);
tr = 0.01;
e1 = sum(irhatx([1:fs*tr]).^2);
new_ir = [e1^0.5 zeros(1,fs*tr-1) irhatx(fs*tr:length(irhatx))'];

tr2 = 2374/fs;
e2 = sum(irhatx([fs*tr:fs*tr2]).^2);

new_ir = [new_ir(1:tr2*fs) e2^0.5 zeros(1,fs*tr2-fs*tr-1) irhatx(fs*tr2:length(irh

tL = tr2+tr;

rest = [1+zeros(1,fs*tr2) zeros(1,fs*tL-fs*tr2) 0.5*(1+cos(2*pi*[fs*tL:fs*tL+0.002

w = [rest 1+zeros(1,length(new_ir)-length(rest))];

new_ir = new_ir.*w;


t = [1:length(new_ir)]/fs;
plot(t,new_ir);
title('Modified IR');
xlabel('Time(s)');
ylabel('Amplitude');

figure();
ftgram(new_ir(1,:), fs, 'rir');

% Problem 5b
[y,fs] = wavread('vocal recordings/TomsDiner_full.wav');

irhatx = irhatx(1.2e05:length(irhatx));
new_ir = new_ir(1.2e05:length(irhatx));


% y_oldIR = fftfilt(irhatx,y(:,1));
% y_newIR = fftfilt(y(:,1),new_ir);


% wavwrite(y_oldIR,'vocal recordings/TomsDiner_full_old.wav',fs);
% wavwrite(y_newIR,'vocal recordings/TomsDiner_full_new.wav',fs);
```


Modified IR
Amplitude
Time(s)
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
-0.005
0
5
10
15
20
25


Amplitude (dB)
Frequency (kHz)
Energy (dB)
Time (ms)
10
100
1000
10000
20
10
5
2
1
0.5
0.2
0.1
0.05
0
-10
-20
-30
-40
-50
-60
-70
-80

Published with MATLAB® 7.10