Kevin Quizhpi

DSP Design

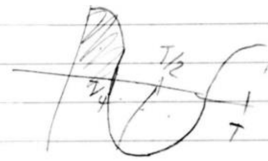
Kdq2

9/28/17

# Project 2: Dynamic Range Control
# Compressors, Limiters, Expanders and Noise Gates

Calculating the mean-square average can be done by integrating the input function squared over a period. After working through it by hand it does simplify to $A^2/2$

$$\overline{|x(t)|} = \frac{4A}{T}\int_0^{T/4} \cos(2\pi f t)$$




$$\frac{2 \cdot 2A}{2\pi f T}\int_0^{T/2} 2\pi f \cos(2\pi f t)\,dt$$

$$\frac{2A}{\pi f T} \sin(2\pi f t)\Big|_0^{T/4}$$

$$\left[\sin \frac{2\pi}{4} fT - \sin(0)\right]$$

$$\frac{2A}{\pi}\left[1 - 0\right]$$

To find the absolute average I integrated the input function over a quarter period and multiplied by 4.

$$\overline{x^2(t)} = \frac{A^2}{T}\int_0^T \frac{1}{2}\,dt + \frac{A^2}{2T}\int_0^T \cos(4\pi f t)\,dt$$

$$\frac{A^2}{2T}\left[T-0\right] \qquad \frac{A^2}{2 \cdot T 4\pi f}\int_0 4\pi f \cos(4\pi f t)\,dt$$

$$\frac{A^2}{2} \qquad \frac{A^2}{8T\pi f} \sin(4\pi f t)\Big|_0^T$$

goes to 0

$$\overline{x^2(t)} = \frac{A^2}{2}$$

# Table of Contents



```
% Kevin Quizhpi
% DSP Design
% Project 2
% 9/26/17
```

# Part A

```
% Establishing anonymous funtion for input x(t)
X = @(t,A,f) A*cos(2*pi*f*t);


MeanSqAvg = @(A) A^2 /2;
AbsAvg = @(A) 2*A /pi;
```

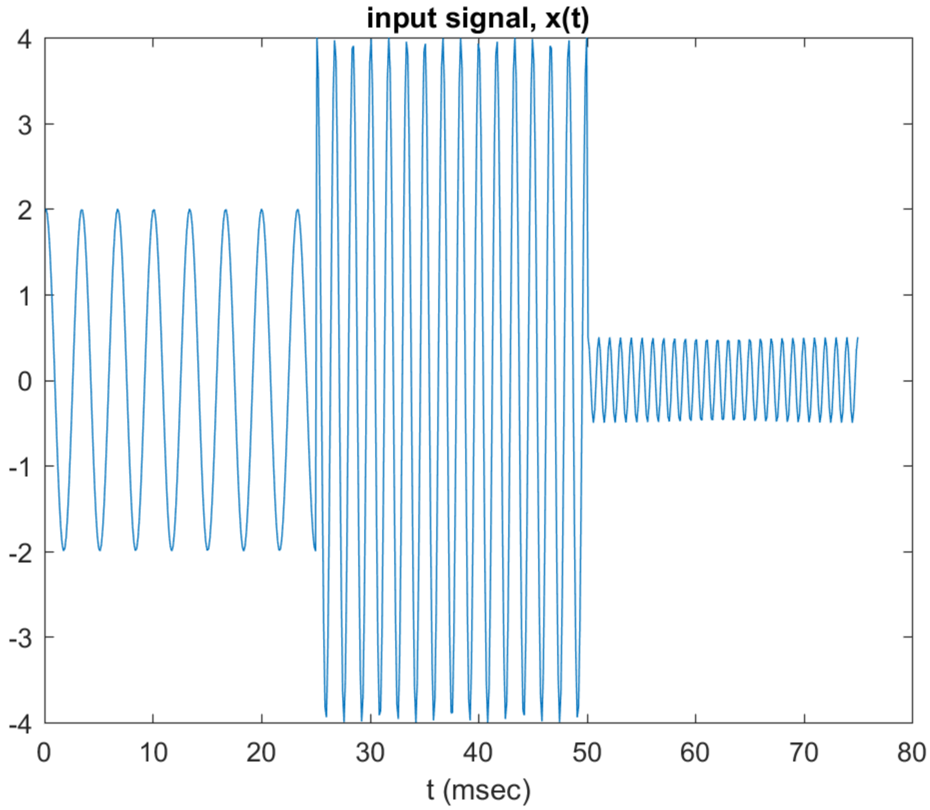
# Part B

```
% Constants used for parts c-g
fs = 8000;
Ts = 1/fs;
ep = 0.1;
LbAtck = ep ^ (Ts/(2/1000));
LbRels = ep ^ (Ts/(1/100));
```

# Part C

```
% We create the signal x(t) that is made up of a concatenation of
 three
% signals each of 25ms length each.
fo = [0.3 0.6 1]*1000;
Ao = [2 4 0.5];
to = [0 25 50]/1000;
Smp = 0.025*fs;     % number of samples per 25ms section

% Function used to extract real world time from the current sample
time = @(n,t0)  n/(Smp-1) *0.025 + t0;
```

```
x = zeros(1,Smp*3);

for i =1:3
    for n = 1:Smp
        t = time(n-1,to(i));
        x(n + 200*(i-1)) = X(t,Ao(i),fo(i));
    end
end


% Mean absolute values
x1MA = AbsAvg(Ao(1));       % 1.2732
x2MA = AbsAvg(Ao(2));       % 2.5465
x3MA = AbsAvg(Ao(3));       % 0.3183

n = 1:length(x);
% Plot of created input signal
figure;
plot(n/8,x(n));
title('input signal, x(t)');
xlabel('t (msec)');
```

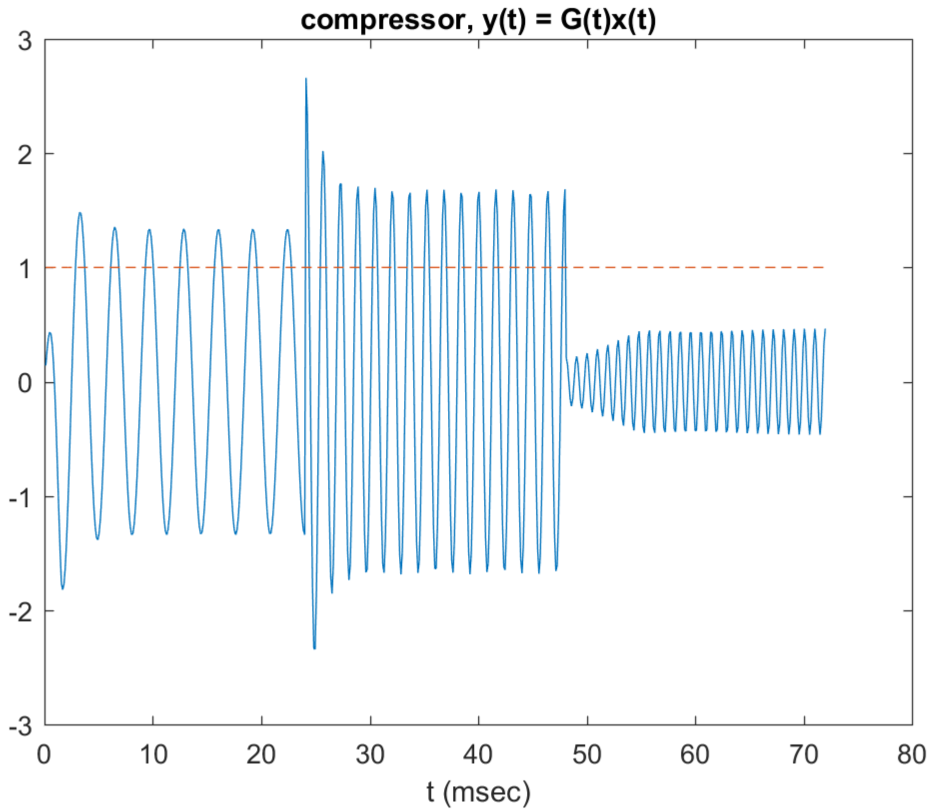


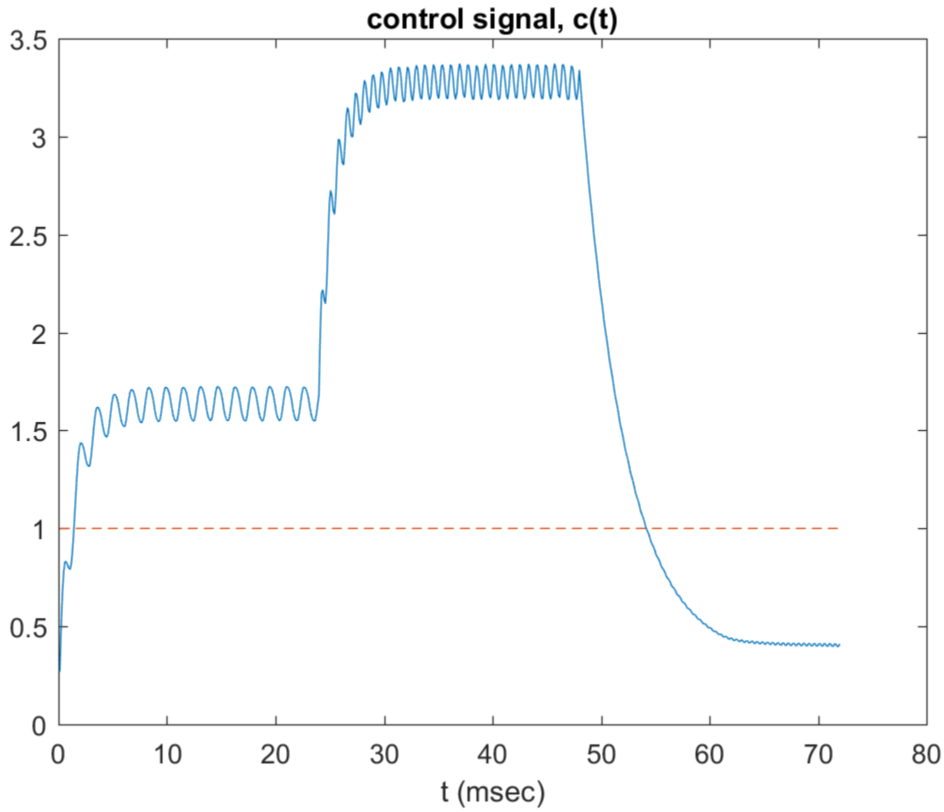

# Part D - Compressor

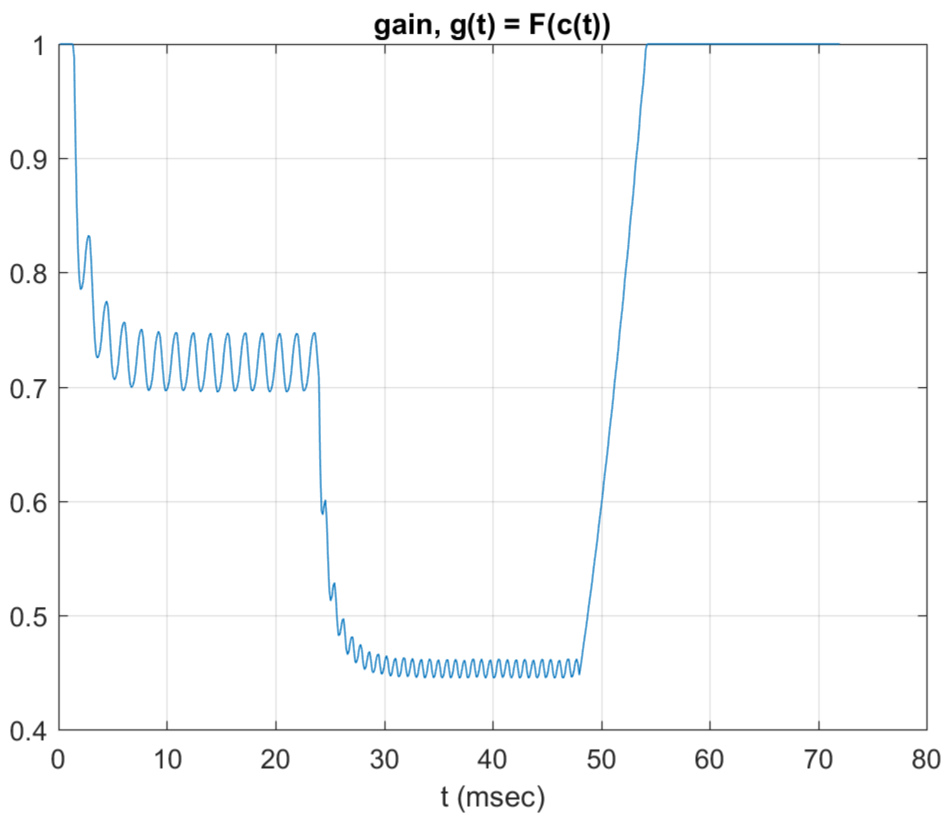
```
% For the compressor p should have a value between 1/4 & 1/2
```

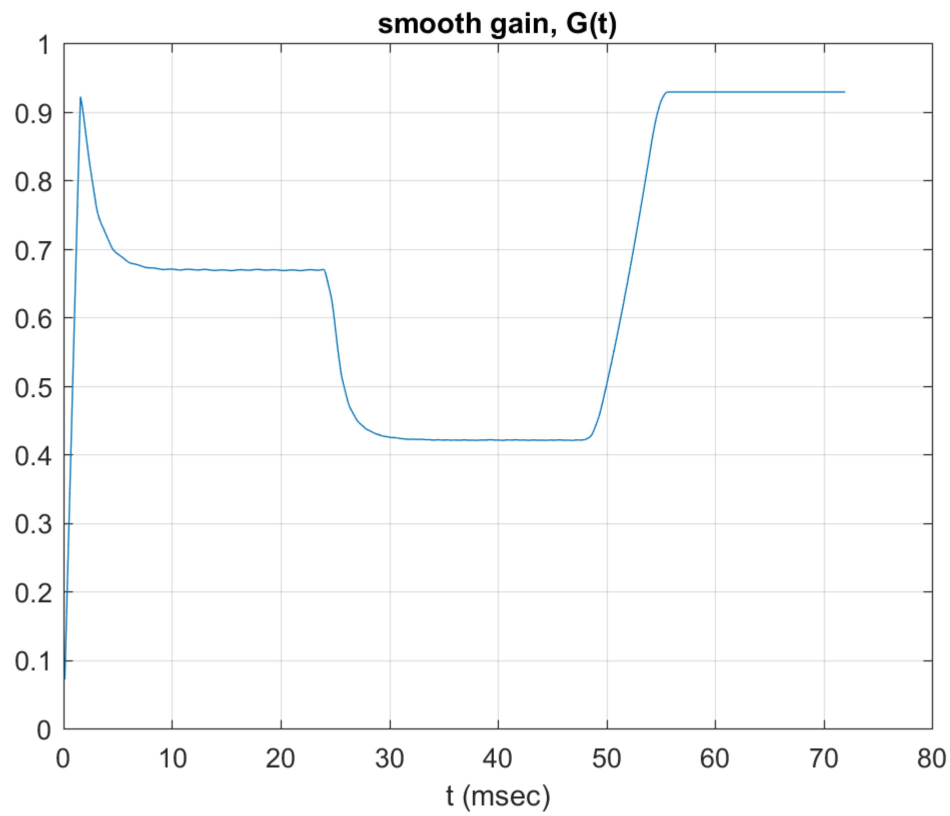
```
p = 1/3;
c0 = 1;
D = 0;

% I've chosen to use the FIR smoothing filter and it requires the use
 of a
% circular buffer of length L
L = ceil((1+LbAtck)/(1-LbAtck));

MvAvgBuf = zeros(1,L);
% Function used to obtain the index of circular buffer
pt = @(i) mod(i,L-1) + 1;
MvAvgSum = 0;
MvAvgOld = 0;
cPrev = 0;

% Initializing arrays of output signals
c = zeros(1,length(x));
g = zeros(1,length(x));
G = zeros(1,length(x));
y = zeros(1,length(x));

% Level detector

cN = @(xn,cPrev) (LbAtck* cPrev + (1-LbAtck)*abs(xn)).*(abs(xn) >=
 cPrev) ...
    + (LbRels*cPrev + (1 - LbRels).*abs(xn)).*(abs(xn) < cPrev);

% Gain Processor functions both Compressor and Expander

gCom = @(c) (c/c0)^(p-1)*(c>= c0) + 1*(c <=c0);

gExp = @(c) (c/c0)^(p-1)*(c <=c0) + 1*(c>= c0);

% Since we are compressing the signal I will use gCom
for i = 1:length(x)

    xn = x(i);
    c(i) = cN(xn,cPrev);
    cPrev = c(i);
    g(i) = gCom(c(i));
    MvAvgOld = MvAvgBuf(pt(i-1));
    MvAvgBuf(pt(i-1)) = g(i);
    MvAvgSum = MvAvgSum + g(i) - MvAvgOld;
    G(i) = MvAvgSum/L;
    y(i) = G(i)*xn;

end

figure;
plot(n*3/25,y, n*3/25, ones(1,length(n))*c0,'--')
title('compressor, y(t) = G(t)x(t)');
xlabel('t (msec)');
```

```
figure;
plot(n*3/25,c, n*3/25, ones(1,length(n))*c0,'--')
title('control signal, c(t)');
xlabel('t (msec)');

figure;
plot(n*3/25,g)
title('gain, g(t) = F(c(t))');
xlabel('t (msec)');
grid on;

figure;
plot(n*3/25,G)
title('smooth gain, G(t)');
xlabel('t (msec)');
grid on;
```




control signal, c(t)
3.5
3
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
t (msec)

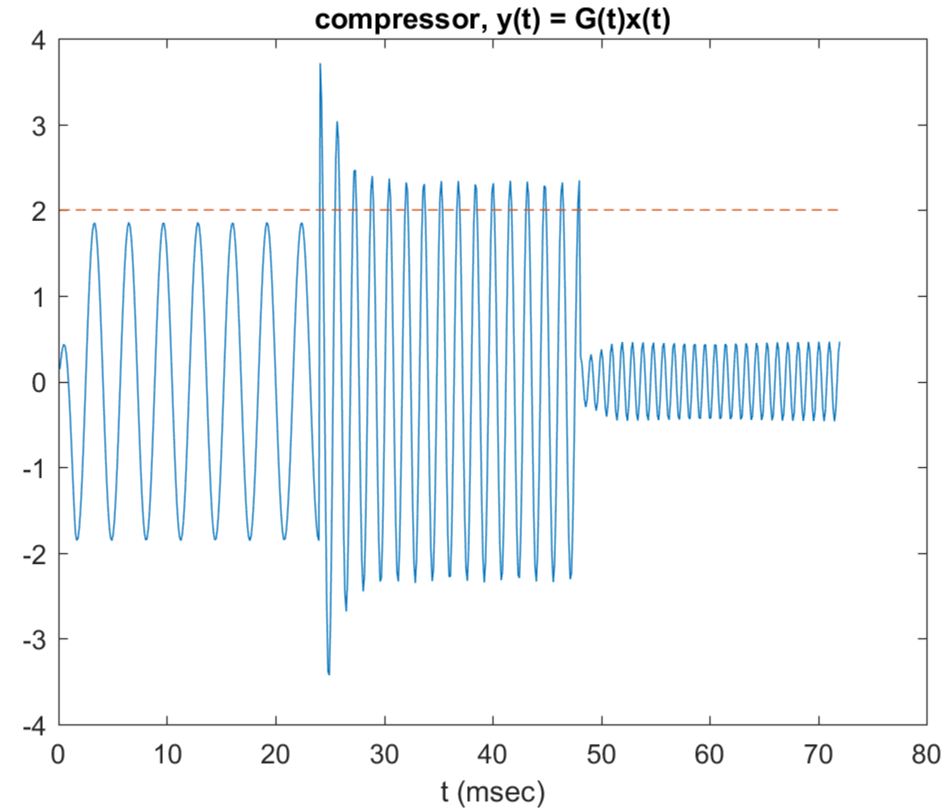

gain, g(t) = F(c(t))
1
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50
60
70
80
t (msec)

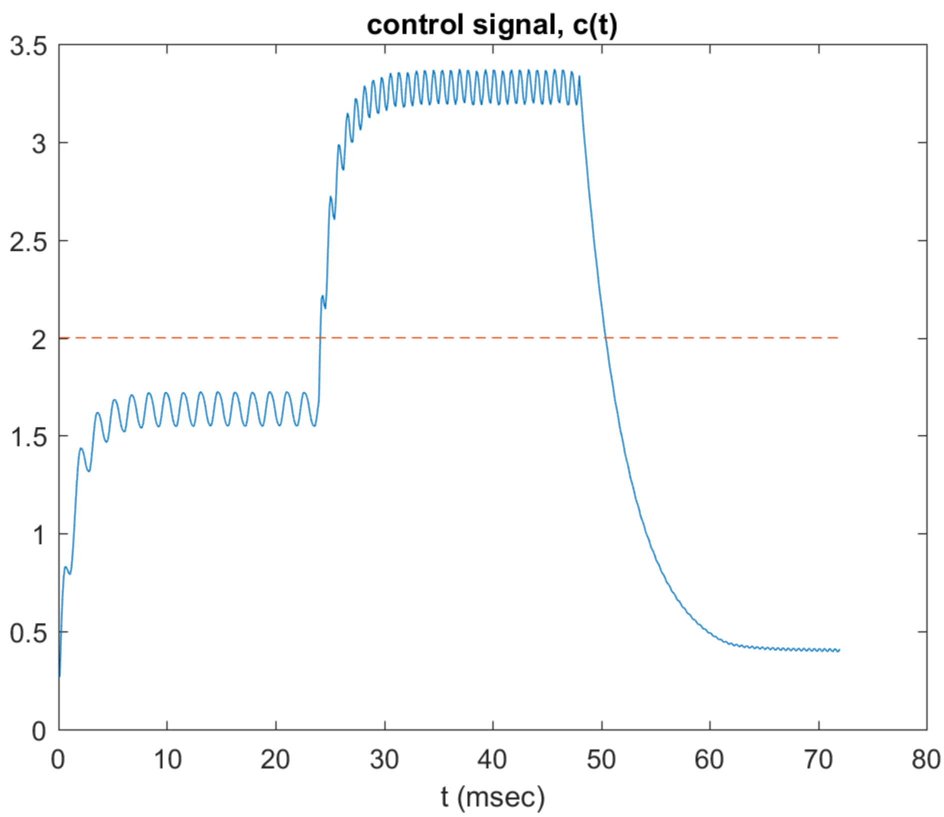


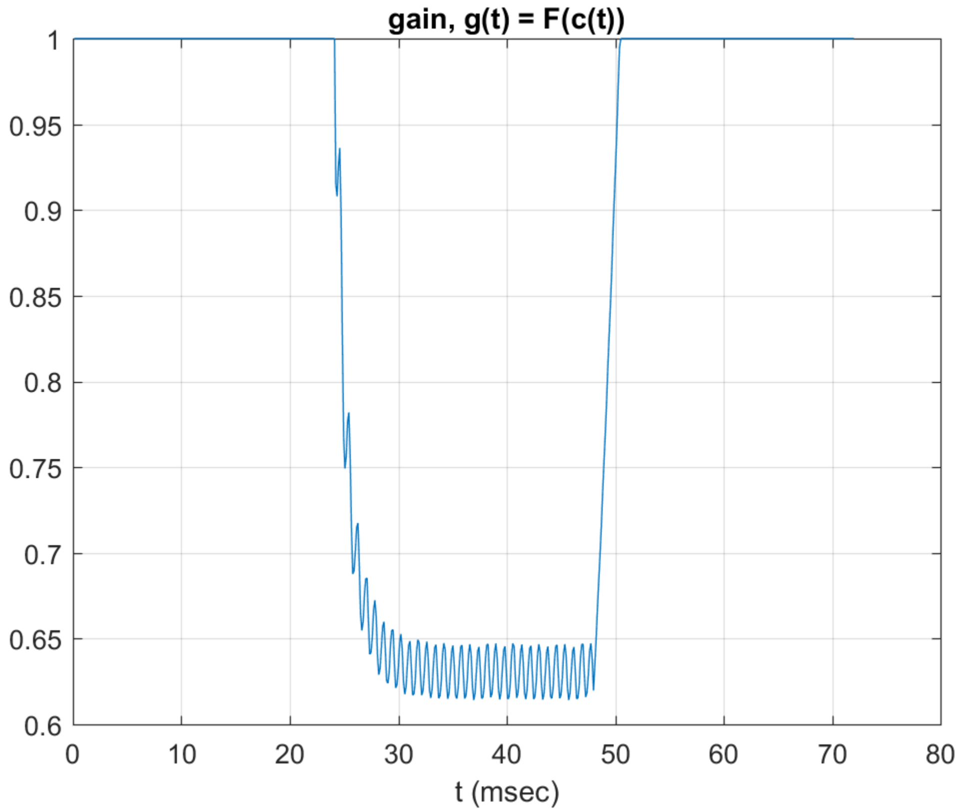

# Part E Limiter

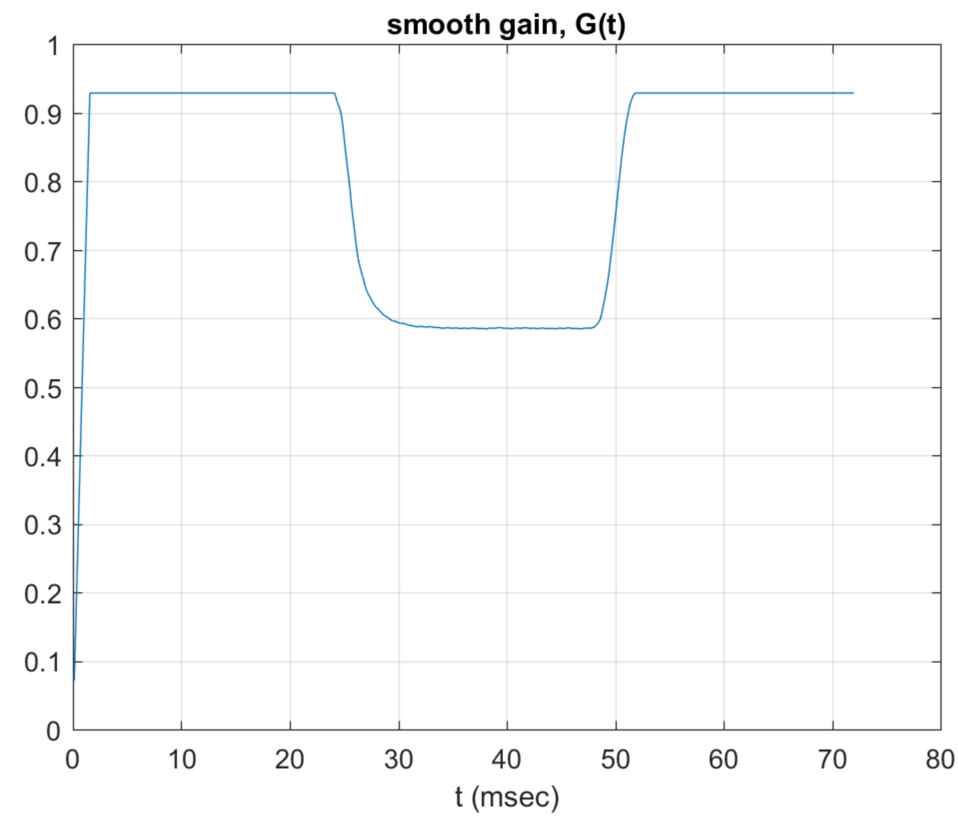
```
% For a limiter p should be much less than 1 arond the area of 1/10

p = 1/15;
c0 = 2;


MvAvgBuf = zeros(1,L);
pt = @(i) mod(i,L-1) + 1;
MvAvgSum = 0;
MvAvgOld = 0;
cPrev = 0;
c = zeros(1,length(x));
g = zeros(1,length(x));

% Level detector

cN = @(xn,cPrev) (LbAtck* cPrev + (1-LbAtck)*abs(xn)).*(abs(xn) >=
 cPrev) ...
    + (LbRels*cPrev + (1 - LbRels).*abs(xn)).*(abs(xn) < cPrev);

% Gain Processor functions both Compressor and Expander

gCom = @(c) (c/c0)^(p-1)*(c>= c0) + 1*(c <=c0);
```

```
gExp = @(c) (c/c0)^(p-1)*(c <=c0) + 1*(c>= c0);

% Since we are compressing the signal I will use gCom
for i = 1:length(x)

    xn = x(i);
    c(i) = cN(xn,cPrev);
    cPrev = c(i);
    g(i) = gCom(c(i));
    MvAvgOld = MvAvgBuf(pt(i-1));
    MvAvgBuf(pt(i-1)) = g(i);
    MvAvgSum = MvAvgSum + g(i) - MvAvgOld;
    G(i) = MvAvgSum/L;
    y(i) = G(i)*xn;

end

figure;
plot(n*3/25,y, n*3/25, ones(1,length(n))*c0,'--')
title('compressor, y(t) = G(t)x(t)');
xlabel('t (msec)');

figure;
plot(n*3/25,c, n*3/25, ones(1,length(n))*c0,'--')
title('control signal, c(t)');
xlabel('t (msec)');

figure;
plot(n*3/25,g)
title('gain, g(t) = F(c(t))');
xlabel('t (msec)');
grid on;

figure;
plot(n*3/25,G)
title('smooth gain, G(t)');
xlabel('t (msec)');
grid on;
```


compressor, y(t) = G(t)x(t)
4
3
2
1
0
-1
-2
-3
-4
0
10
20
30
40
50
60
70
80
t (msec)


control signal, c(t)
3.5
3
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
t (msec)

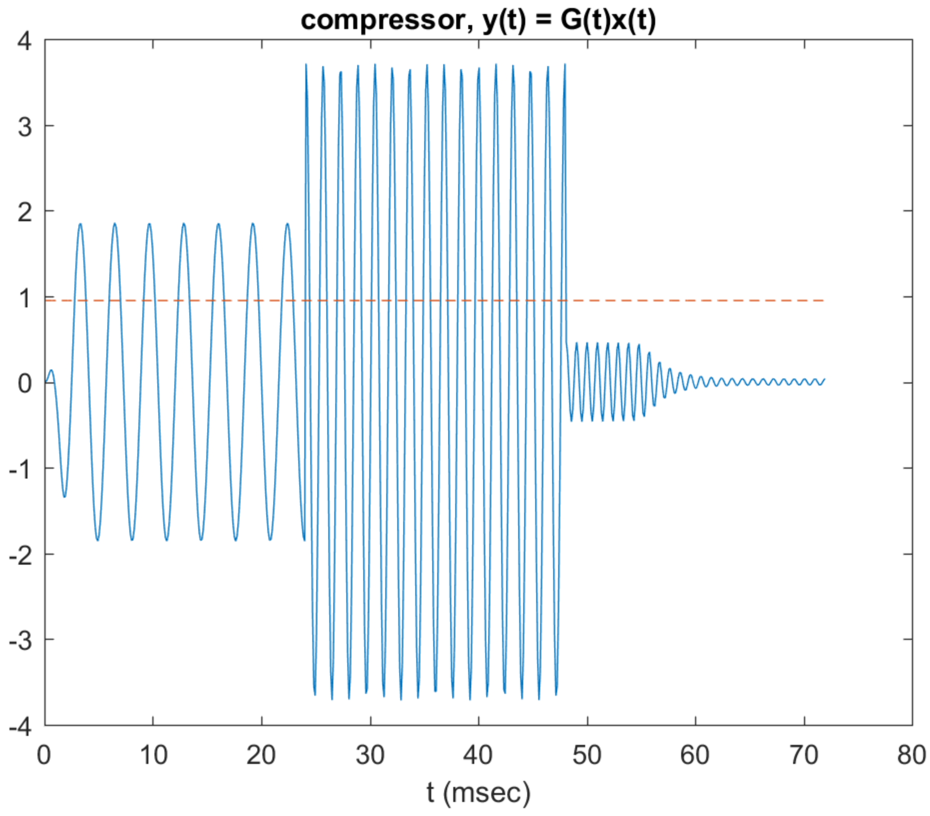

gain, g(t) = F(c(t))
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0
10
20
30
40
50
60
70
80
t (msec)

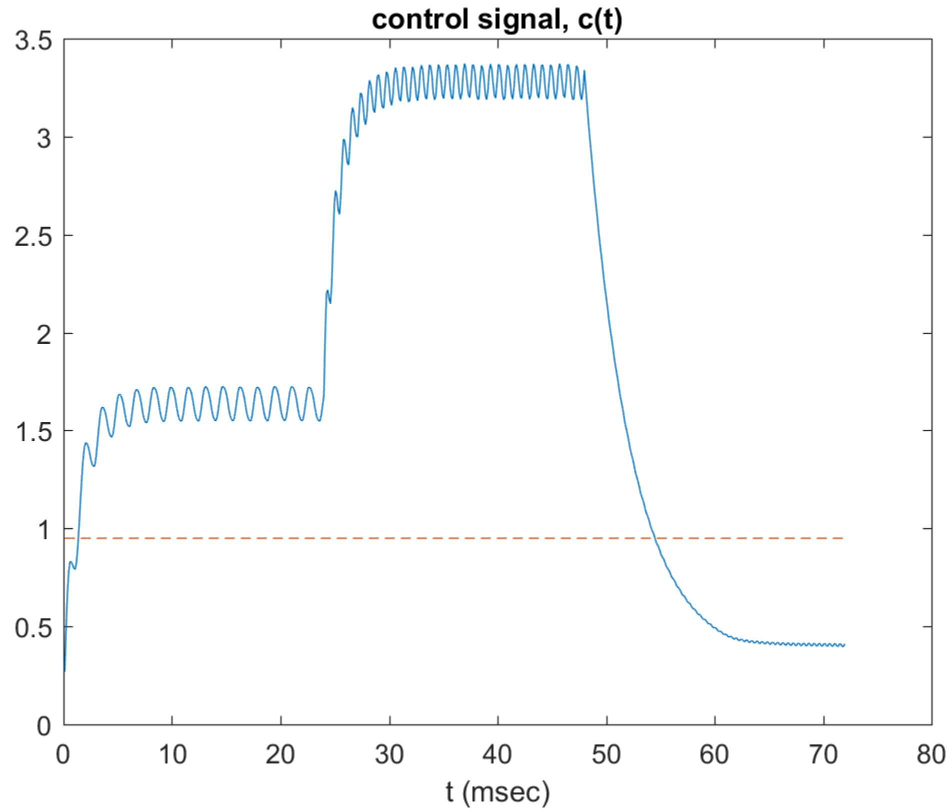

smooth gain, G(t)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
t (msec)

# Part F Expander

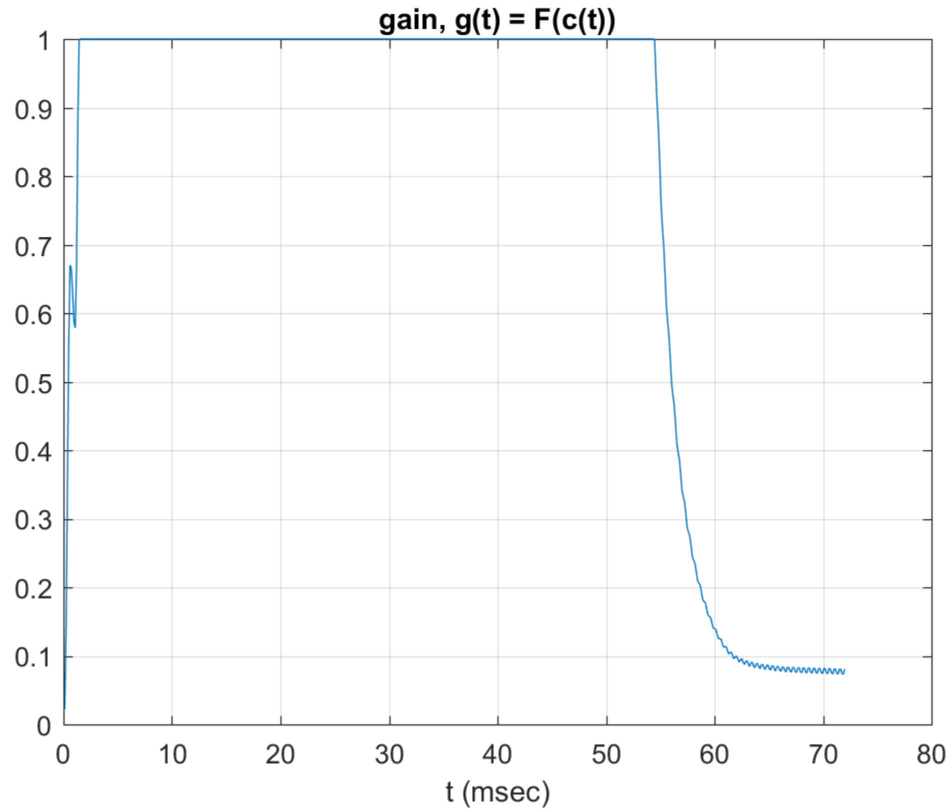
```
% For an expander p should be around 2-4

p = 4;
c0 = 0.95;


MvAvgBuf = zeros(1,L);
pt = @(i) mod(i,L-1) + 1;
MvAvgSum = 0;
MvAvgOld = 0;
cPrev = 0;
c = zeros(1,length(x));
g = zeros(1,length(x));

% Level detector

cN = @(xn,cPrev) (LbAtck* cPrev + (1-LbAtck)*abs(xn)).*(abs(xn) >=
 cPrev) ...
    + (LbRels*cPrev + (1 - LbRels).*abs(xn)).*(abs(xn) < cPrev);

% Gain Processor functions both Compressor and Expander

gCom = @(c) (c/c0)^(p-1)*(c>= c0) + 1*(c <=c0);

gExp = @(c) (c/c0)^(p-1)*(c <=c0) + 1*(c>= c0);

% Since we are compressing the signal I will use gCom
for i = 1:length(x)

    xn = x(i);
    c(i) = cN(xn,cPrev);
    cPrev = c(i);
    g(i) = gExp(c(i));
    MvAvgOld = MvAvgBuf(pt(i-1));
    MvAvgBuf(pt(i-1)) = g(i);
    MvAvgSum = MvAvgSum + g(i) - MvAvgOld;
    G(i) = MvAvgSum/L;
    y(i) = G(i)*xn;

end

figure;
plot(n*3/25,y, n*3/25, ones(1,length(n))*c0,'--')
title('compressor, y(t) = G(t)x(t)');
xlabel('t (msec)');

figure;
plot(n*3/25,c, n*3/25, ones(1,length(n))*c0,'--')
title('control signal, c(t)');
xlabel('t (msec)');
```

```
figure;
plot(n*3/25,g)
title('gain, g(t) = F(c(t))');
xlabel('t (msec)');
grid on;

figure;
plot(n*3/25,G)
title('smooth gain, G(t)');
xlabel('t (msec)');
grid on;
```

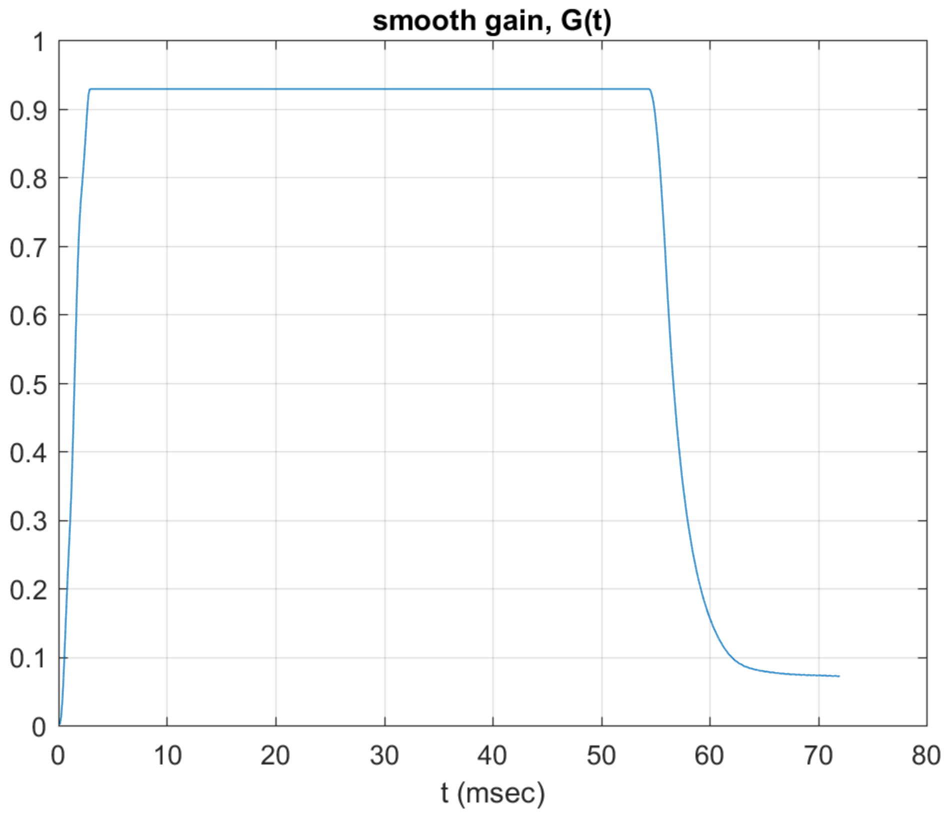




control signal, c(t)
3.5
3
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
t (msec)

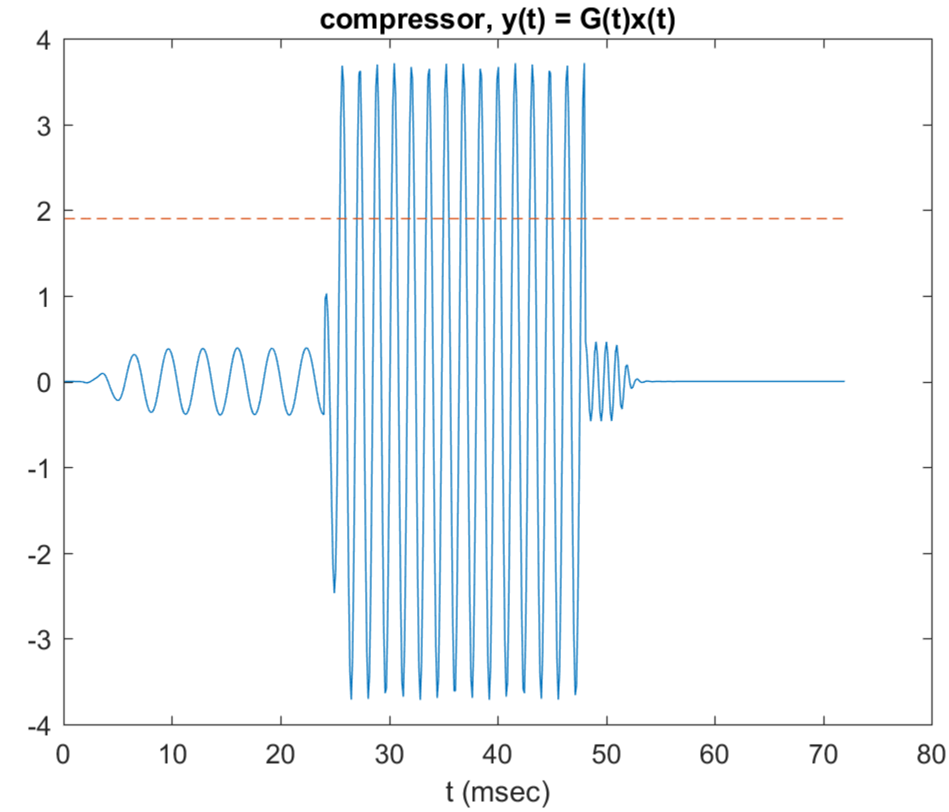
gain, g(t) = F(c(t))
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
t (msec)

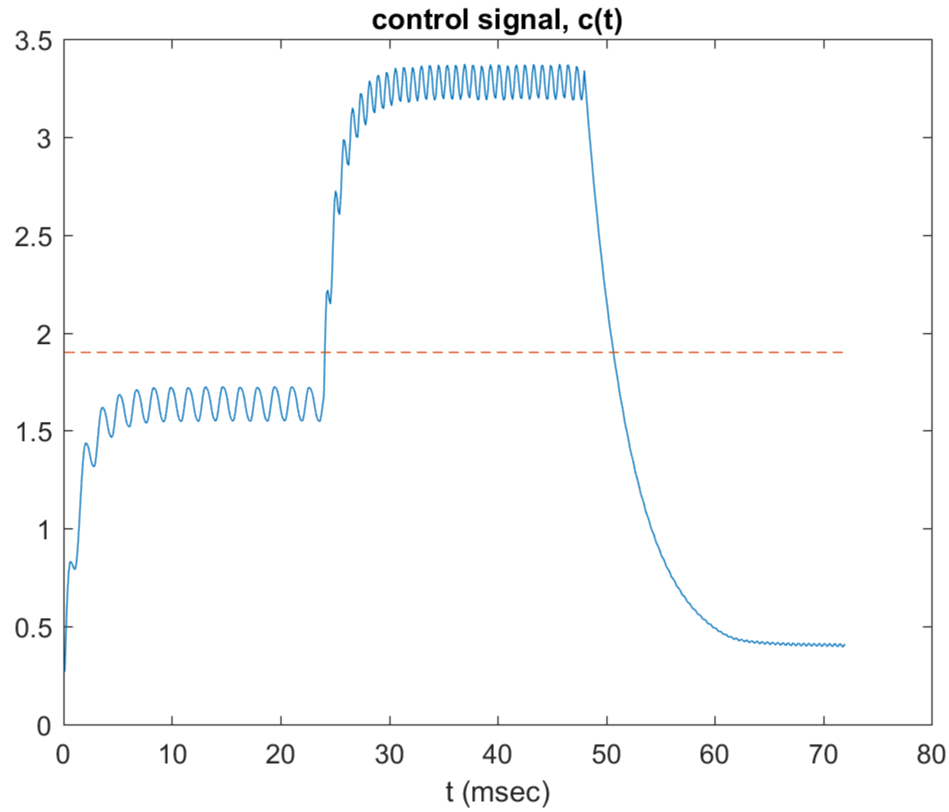


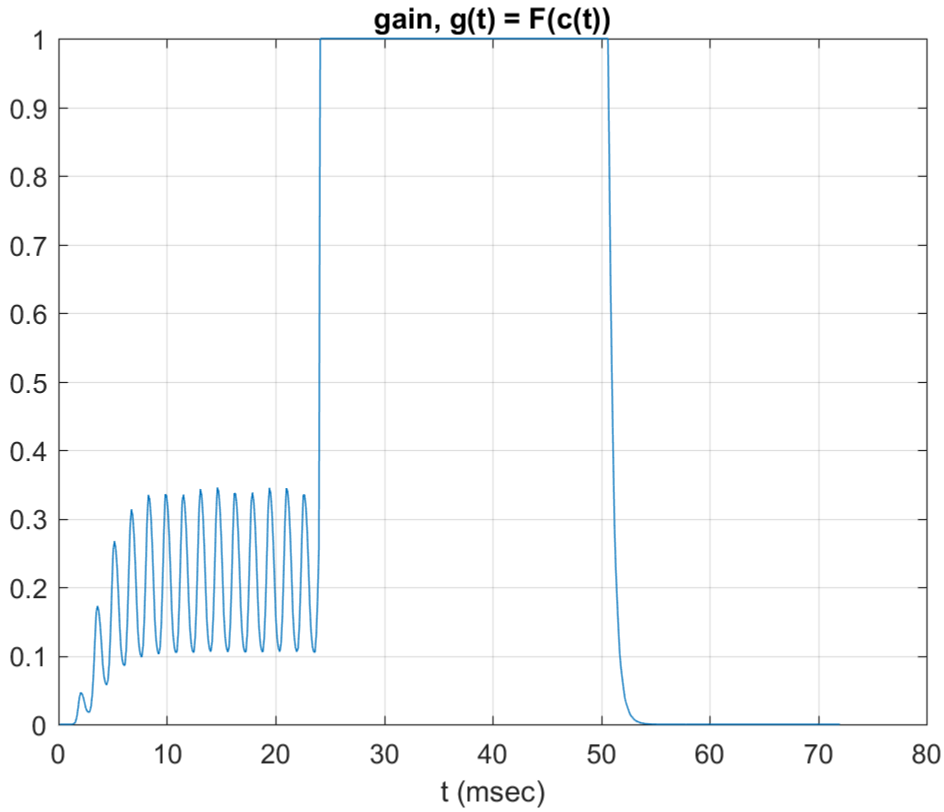

# Part G Noise Gate: f1 & f3

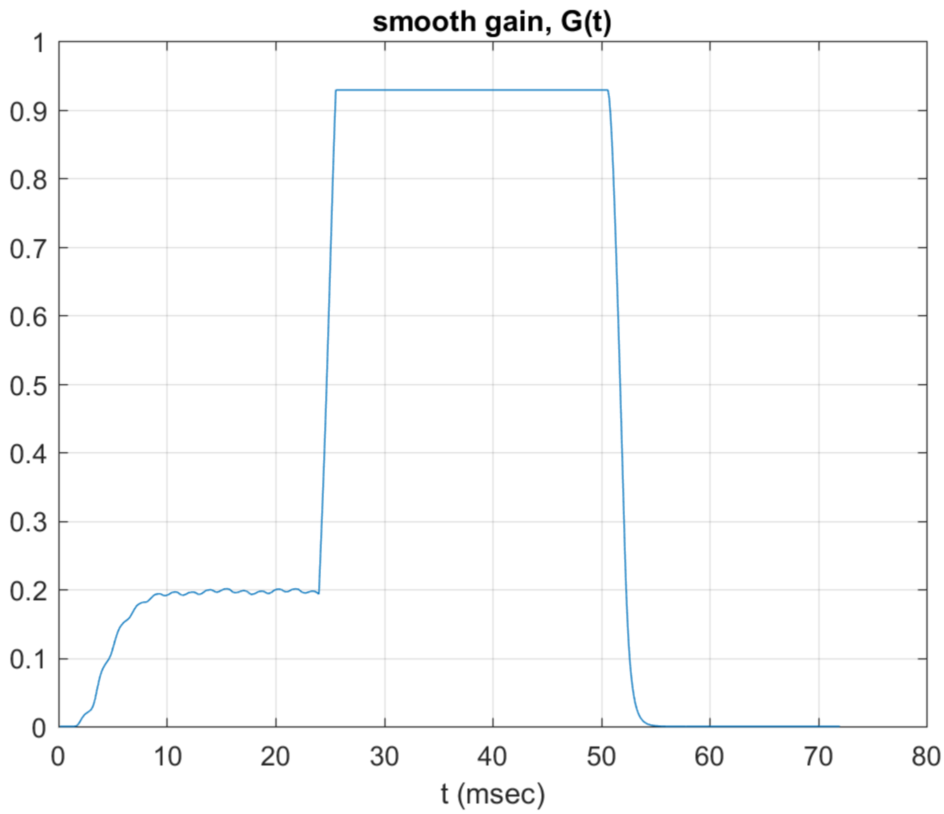
```
% For a noise gate p should be much greater than 1, around 10+

p = 12;
c0 = 1.9;


MvAvgBuf = zeros(1,L);
pt = @(i) mod(i,L-1) + 1;
MvAvgSum = 0;
MvAvgOld = 0;
cPrev = 0;
c = zeros(1,length(x));
g = zeros(1,length(x));

% Level detector

cN = @(xn,cPrev) (LbAtck* cPrev + (1-LbAtck)*abs(xn)).*(abs(xn) >=
 cPrev) ...
    + (LbRels*cPrev + (1 - LbRels).*abs(xn)).*(abs(xn) < cPrev);

% Gain Processor functions both Compressor and Expander

gCom = @(c) (c/c0)^(p-1)*(c>= c0) + 1*(c <=c0);
```

```
gExp = @(c) (c/c0)^(p-1)*(c <=c0) + 1*(c>= c0);

% Since we are compressing the signal I will use gCom
for i = 1:length(x)

    xn = x(i);
    c(i) = cN(xn,cPrev);
    cPrev = c(i);
    g(i) = gExp(c(i));
    MvAvgOld = MvAvgBuf(pt(i-1));
    MvAvgBuf(pt(i-1)) = g(i);
    MvAvgSum = MvAvgSum + g(i) - MvAvgOld;
    G(i) = MvAvgSum/L;
    y(i) = G(i)*xn;

end

figure;
plot(n*3/25,y, n*3/25, ones(1,length(n))*c0,'--')
title('compressor, y(t) = G(t)x(t)');
xlabel('t (msec)');

figure;
plot(n*3/25,c, n*3/25, ones(1,length(n))*c0,'--')
title('control signal, c(t)');
xlabel('t (msec)');

figure;
plot(n*3/25,g)
title('gain, g(t) = F(c(t))');
xlabel('t (msec)');
grid on;

figure;
plot(n*3/25,G)
title('smooth gain, G(t)');
xlabel('t (msec)');
grid on;
```


compressor, y(t) = G(t)x(t)
4
3
2
1
0
-1
-2
-3
-4
0
10
20
30
40
50
60
70
80
t (msec)


control signal, c(t)
3.5
3
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
t (msec)

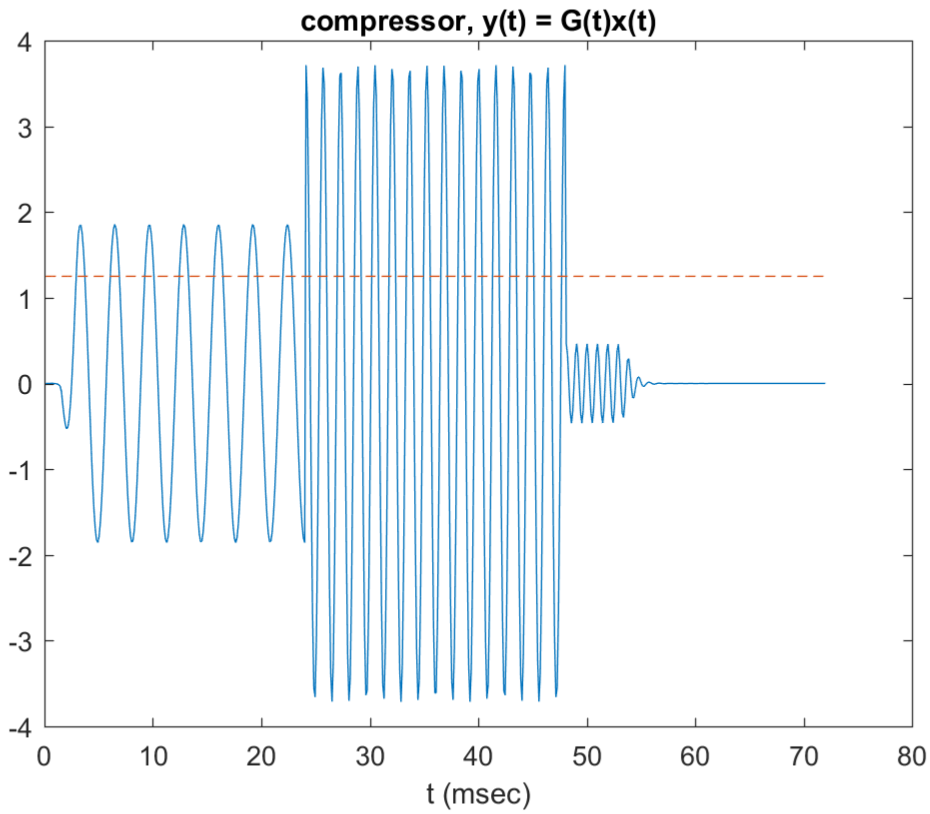

gain, g(t) = F(c(t))
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
t (msec)

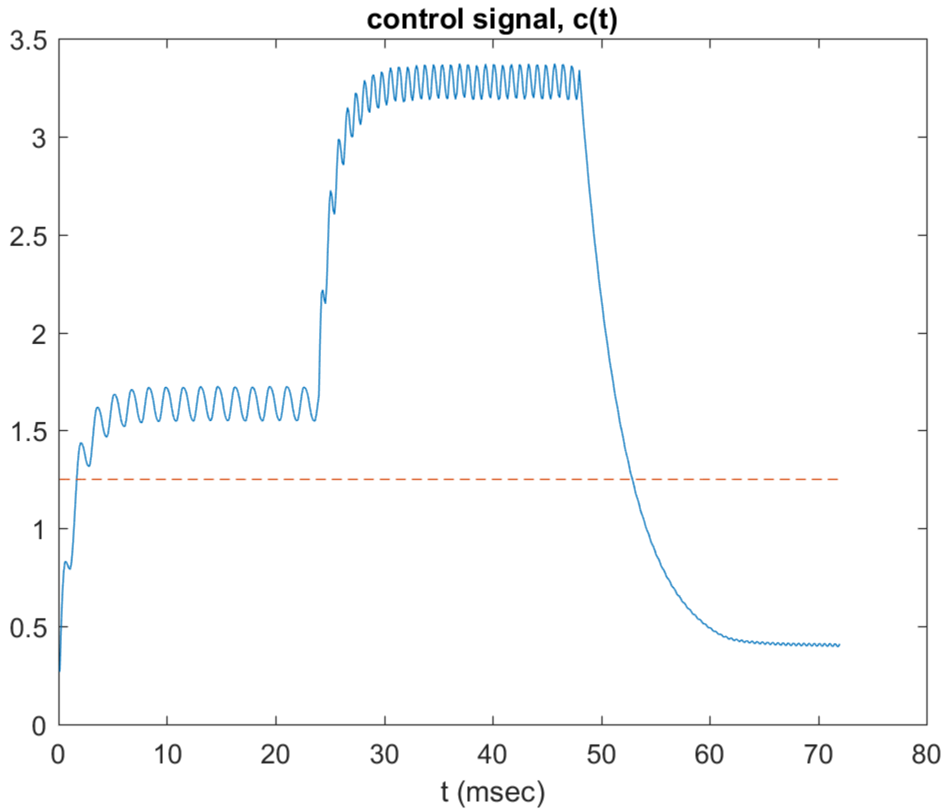

smooth gain, G(t)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
t (msec)

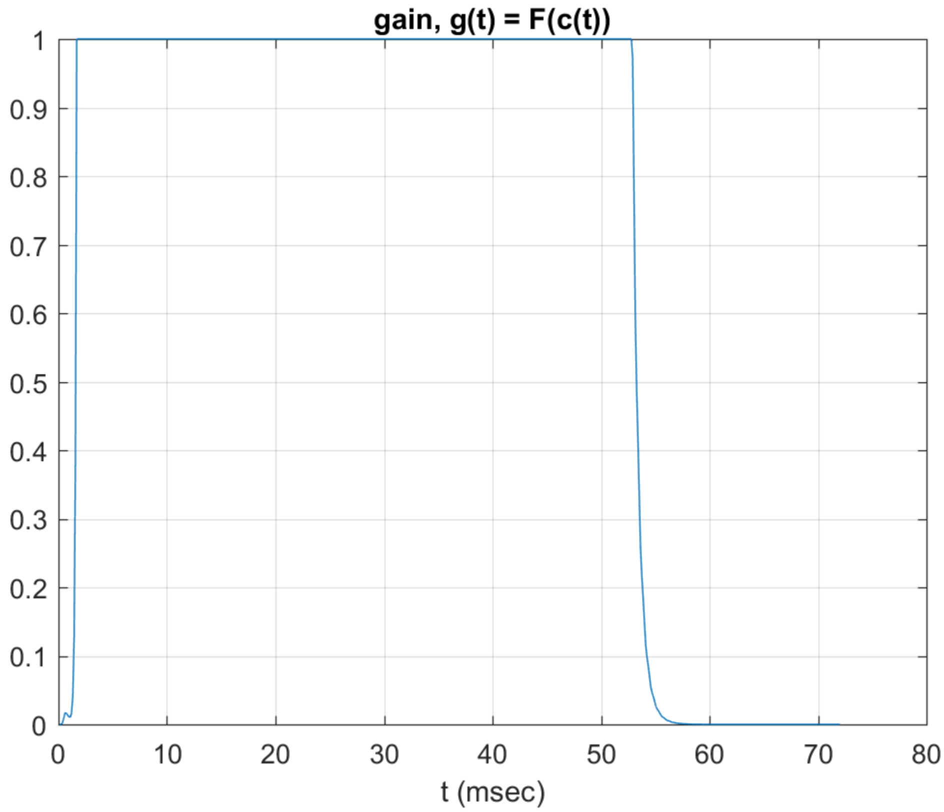
# Part G Noise Gate: f3

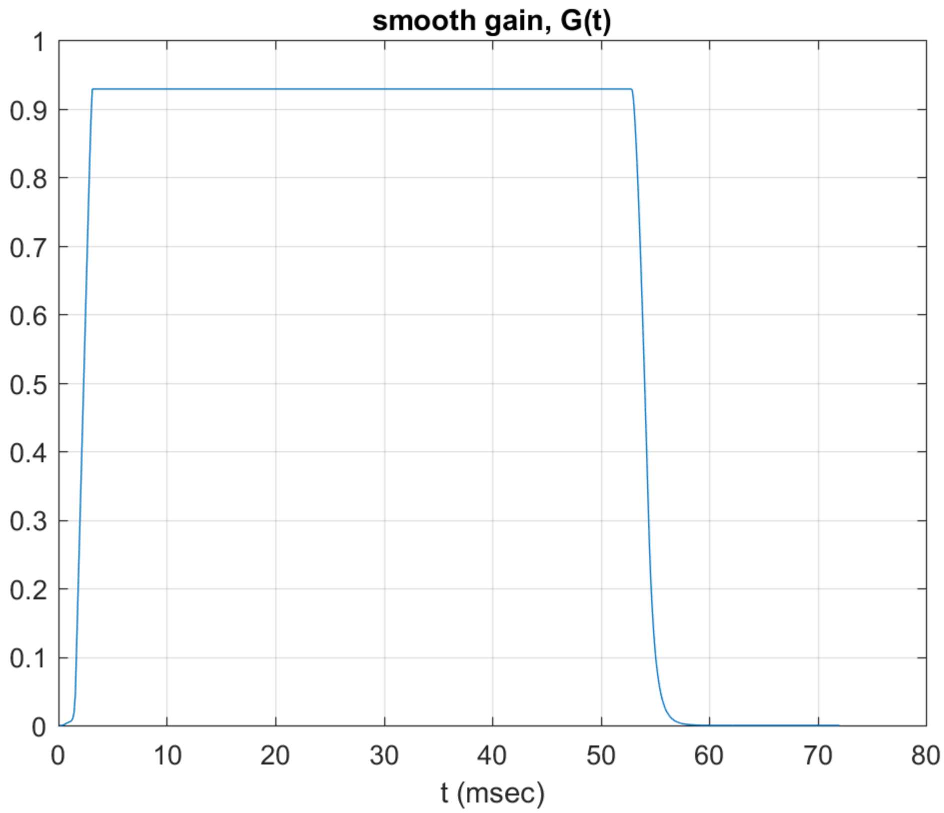
```
% For a noise gate p should be much greater than 1, around 10+

p = 11;
c0 = 1.25;


MvAvgBuf = zeros(1,L);
pt = @(i) mod(i,L-1) + 1;
MvAvgSum = 0;
MvAvgOld = 0;
cPrev = 0;
c = zeros(1,length(x));
g = zeros(1,length(x));

% Level detector

cN = @(xn,cPrev) (LbAtck* cPrev + (1-LbAtck)*abs(xn)).*(abs(xn) >=
 cPrev) ...
    + (LbRels*cPrev + (1 - LbRels).*abs(xn)).*(abs(xn) < cPrev);

% Gain Processor functions both Compressor and Expander

gCom = @(c) (c/c0)^(p-1)*(c>= c0) + 1*(c <=c0);

gExp = @(c) (c/c0)^(p-1)*(c <=c0) + 1*(c>= c0);

% Since we are compressing the signal I will use gCom
for i = 1:length(x)

    xn = x(i);
    c(i) = cN(xn,cPrev);
    cPrev = c(i);
    g(i) = gExp(c(i));
    MvAvgOld = MvAvgBuf(pt(i-1));
    MvAvgBuf(pt(i-1)) = g(i);
    MvAvgSum = MvAvgSum + g(i) - MvAvgOld;
    G(i) = MvAvgSum/L;
    y(i) = G(i)*xn;

end

figure;
plot(n*3/25,y, n*3/25, ones(1,length(n))*c0,'--')
title('compressor, y(t) = G(t)x(t)');
xlabel('t (msec)');

figure;
plot(n*3/25,c, n*3/25, ones(1,length(n))*c0,'--')
title('control signal, c(t)');
xlabel('t (msec)');
```

```
figure;
plot(n*3/25,g)
title('gain, g(t) = F(c(t))');
xlabel('t (msec)');
grid on;

figure;
plot(n*3/25,G)
title('smooth gain, G(t)');
xlabel('t (msec)');
grid on;
```




control signal, c(t)
3.5
3
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
t (msec)


gain, g(t) = F(c(t))
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
t (msec)

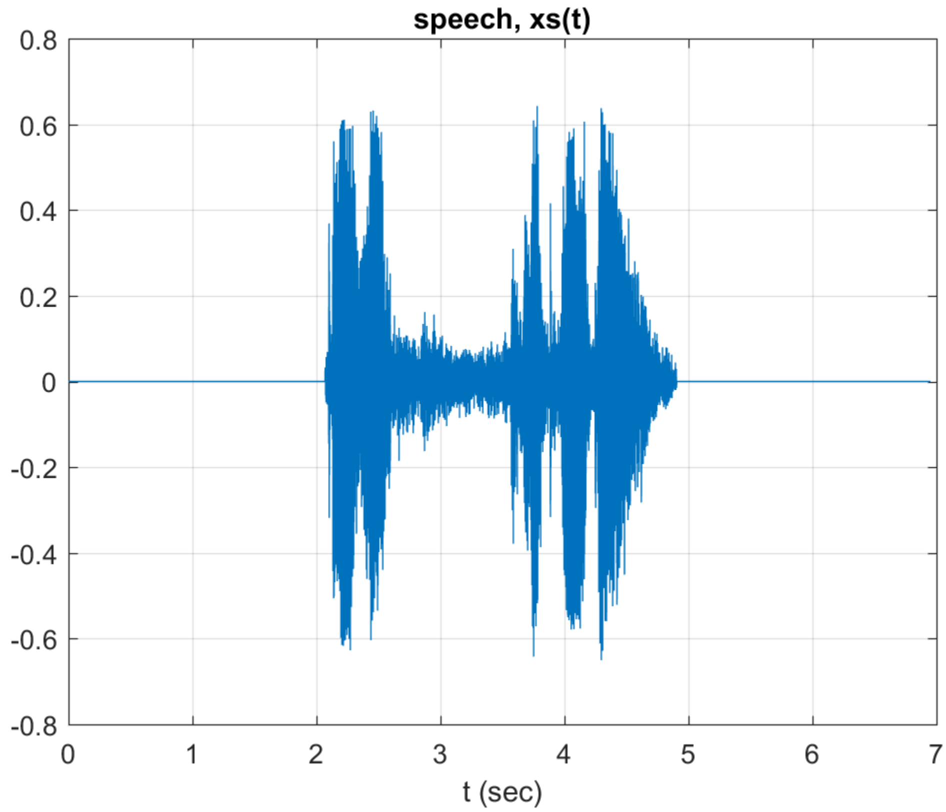


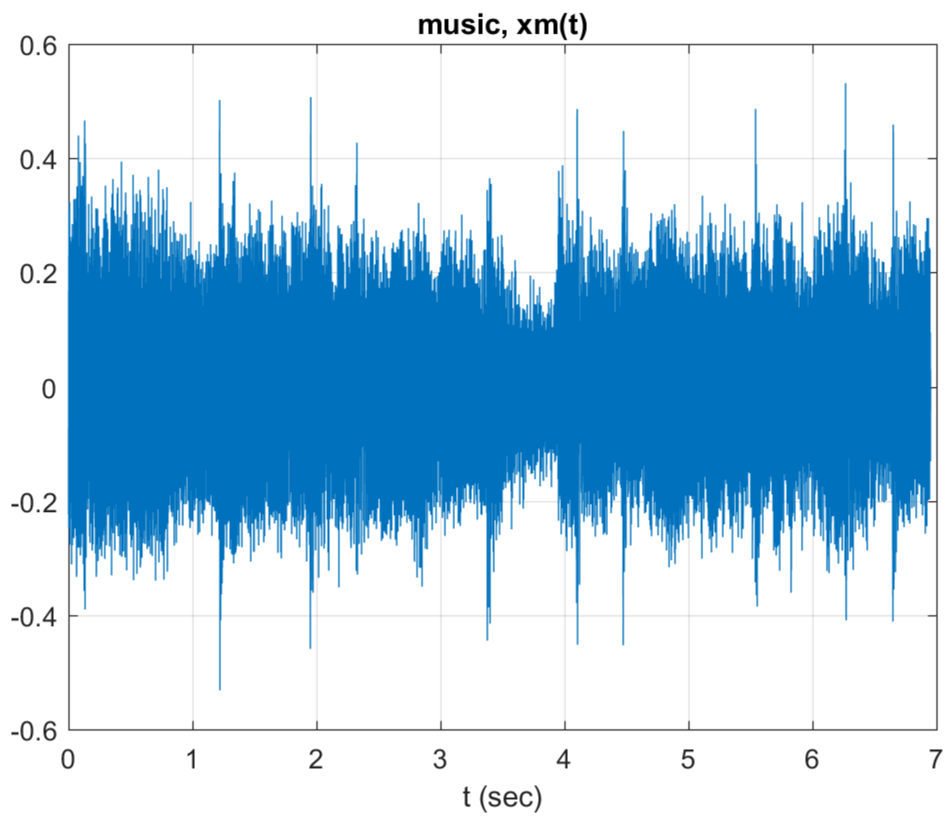

# Part H

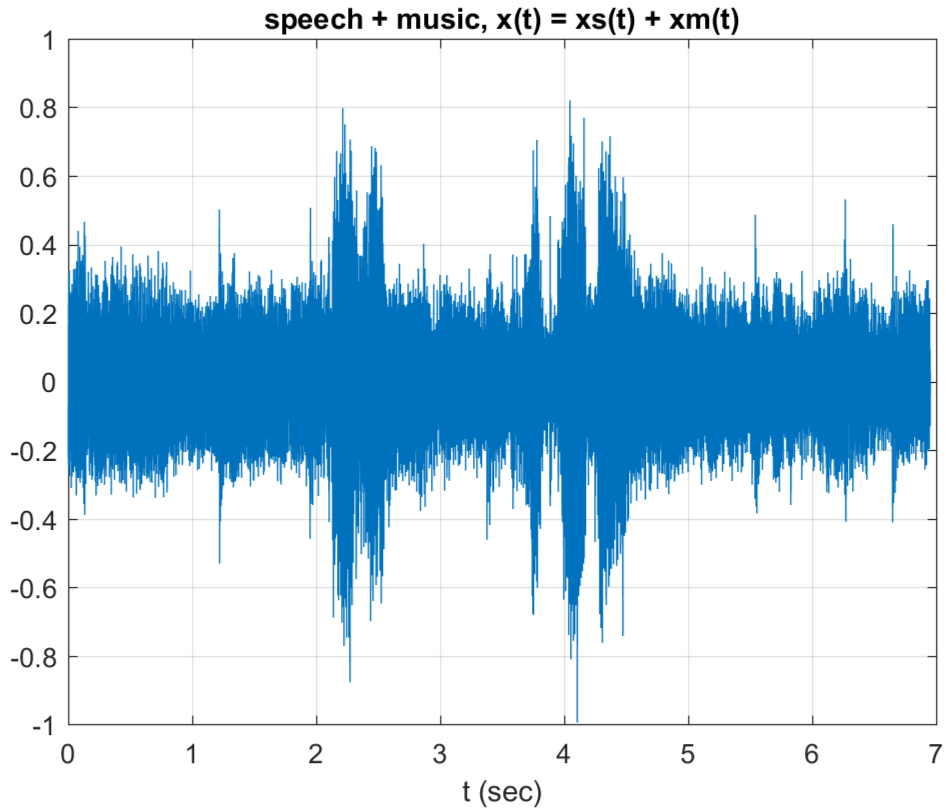
```
% Reading input files
[xs, fs] = audioread('speech.wav');     % Speech signal
[xm, fs] = audioread('music.wav');      % Music signal

% Plot of xs(t), xm(t) & xs(t) + xm(t)
n = 1:length(xs);
figure;
plot(n/44100, xs);
title('speech, xs(t)');
xlabel('t (sec)');
grid on;

figure;
plot(n/44100, xm);
title('music, xm(t)');
xlabel('t (sec)');
grid on;

figure;
plot(n/44100,xm +xs);
title('speech + music, x(t) = xs(t) + xm(t)');
xlabel('t (sec)');
grid on;
```

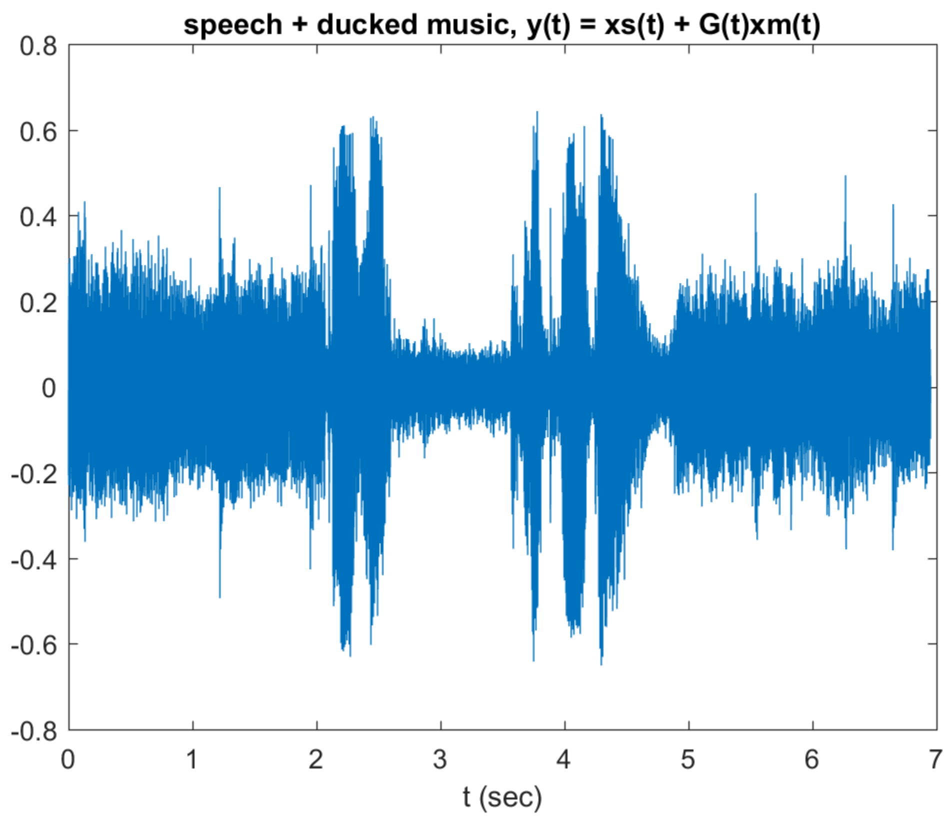
```
% Experimental values of p & c0
p = 1/10;
c0 = 0.005;


MvAvgBuf = zeros(1,L);
pt = @(i) mod(i,L-1) + 1;
MvAvgSum = 0;
MvAvgOld = 0;
cPrev = 0;
c = zeros(1,length(xs));
g = zeros(1,length(xs));
G = zeros(1,length(xs));
ym = zeros(1,length(xs));
y = zeros(1,length(xs));


% Level detector

cN = @(xn,cPrev) (LbAtck* cPrev + (1-LbAtck)*abs(xn)).*(abs(xn) >=
 cPrev) ...
    + (LbRels*cPrev + (1 - LbRels).*abs(xn)).*(abs(xn) < cPrev);

% Gain Processor, since we are compressing I've opted for the
 compressor
% function

gCom = @(c) (c/c0)^(p-1)*(c> c0) + 1*(c <=c0);

for i = 1:length(xs)

    x = xs(i);
    m = xm(i);
    c(i) = cN(x,cPrev);
    cPrev = c(i);
    temp = gCom(c(i));
    % When gCom is inputted 0 it produces NaN istead of outputting
 unity
    % gain so an if statment takes care of that situation
    if(isnan(temp))
        g(i) = 1;
    else
        g(i) = temp;
    end
    MvAvgOld = MvAvgBuf(pt(i-1));
    MvAvgBuf(pt(i-1)) = g(i);
    MvAvgSum = MvAvgSum + g(i) - MvAvgOld;
    gT = MvAvgSum/L;
    G(i) = gT;
    f = m*gT;
    ym(i) = f;
    y(i) = f +x;
```

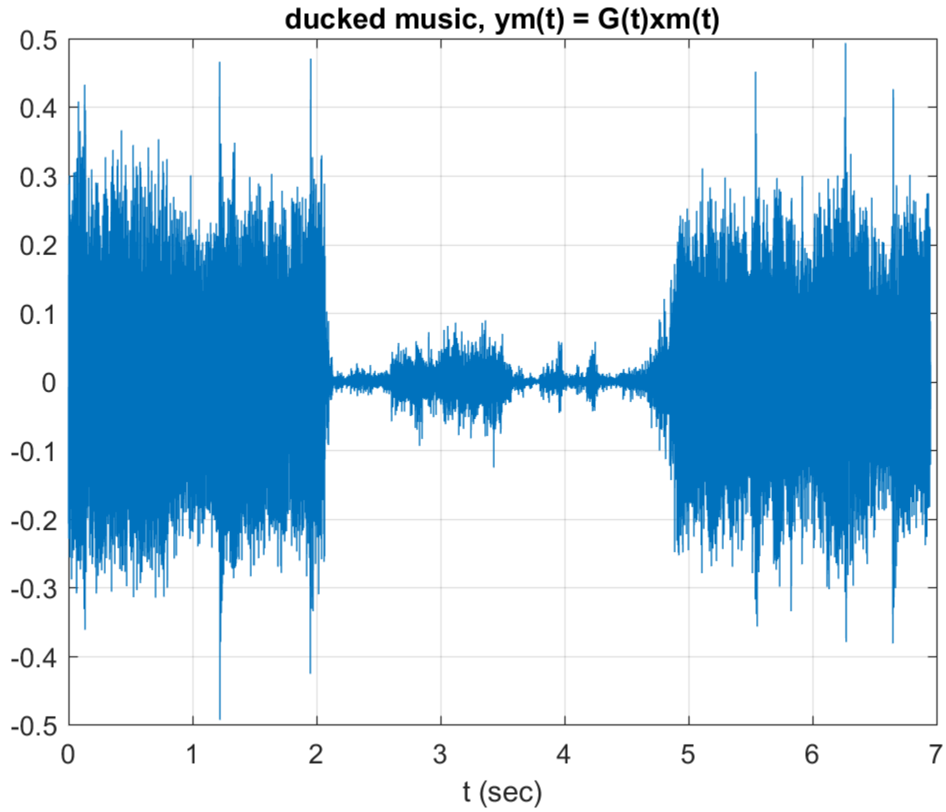
```
end


figure;
plot(n/44100,y)
title('speech + ducked music, y(t) = xs(t) + G(t)xm(t)');
xlabel('t (sec)');

figure;
plot(n/44100,ym)
title('ducked music, ym(t) = G(t)xm(t)');
xlabel('t (sec)');
grid on;

figure;
plot(n/44100,c)
title('control signal, c(t)');
xlabel('t (sec)');
grid on;

figure;
plot(n/44100,g)
title('ducking gain, g(t)');
xlabel('t (sec)');
grid on;

figure;
plot(n/44100,G)
title('smoothed gain, G(t)');
xlabel('t (sec)');
grid on;

audiowrite('duckedSpeech.wav',y,fs);
```

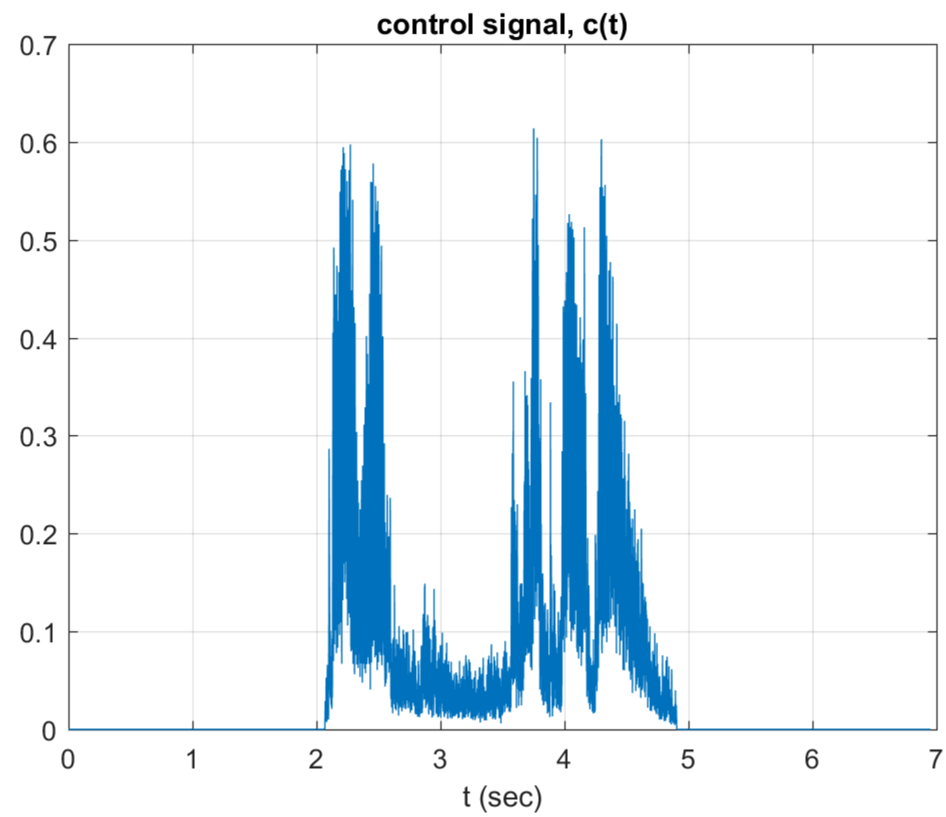

speech, xs(t)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1
2
3
4
5
6
7
t (sec)

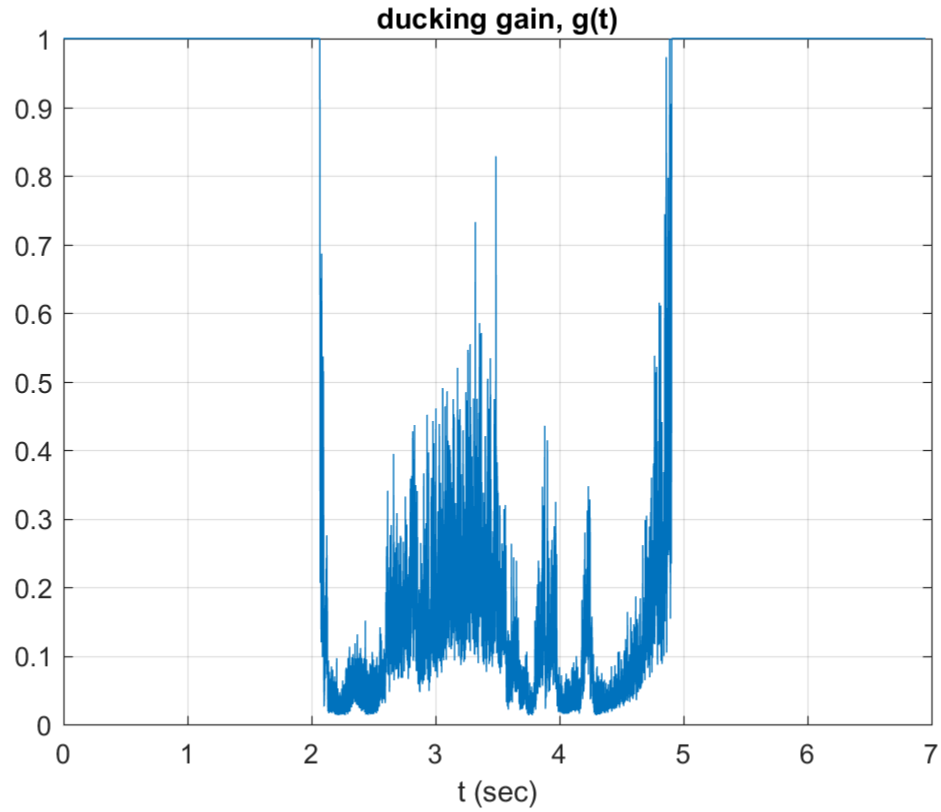

music, xm(t)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
1
2
3
4
5
6
7
t (sec)

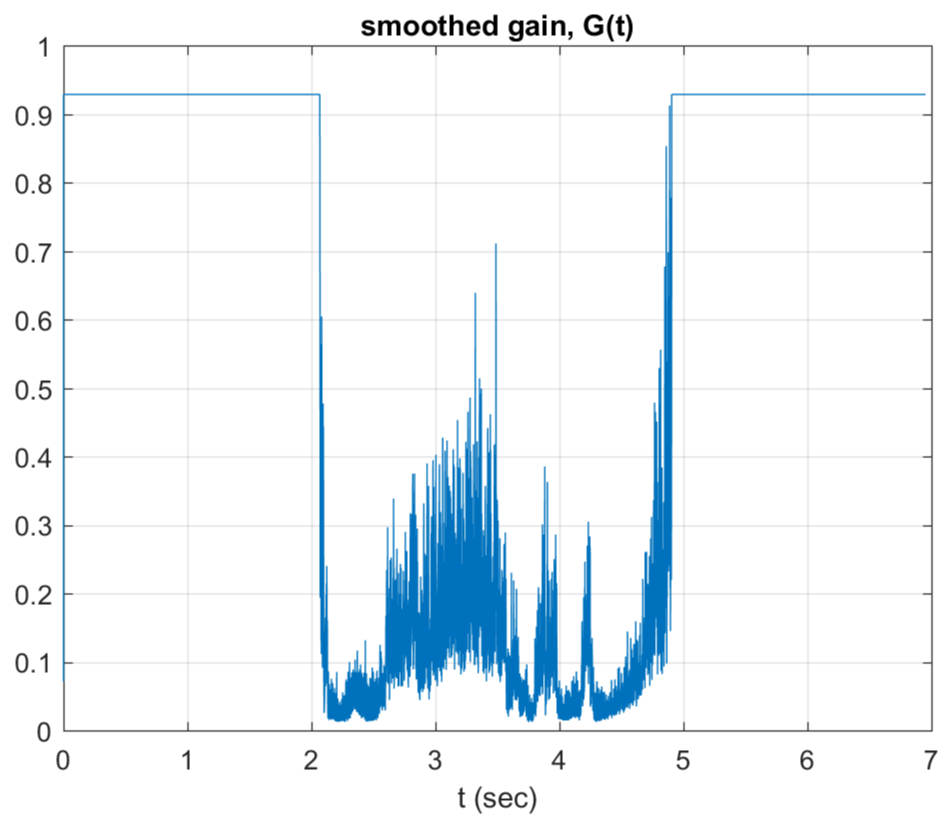

speech + music, x(t) = xs(t) + xm(t)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
t (sec)


speech + ducked music, y(t) = xs(t) + G(t)xm(t)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1
2
3
4
5
6
7
t (sec)


ducked music, ym(t) = G(t)xm(t)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
1
2
3
4
5
6
7
t (sec)


control signal, c(t)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
7
t (sec)


ducking gain, g(t)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7
t (sec)


smoothed gain, G(t)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7
t (sec)

*Published with MATLAB® R2016b*