Nonlinear Processing

Yangang Cao

February 25, 2019

1 Introduction

The terms nonlinear processing or nonlinear processors are used for all signal processing algorithms or signal processing devices in the analog or digital domains which do not satisfy the condition of linearity. A system with input $x(n)$ and output $y(n)$ is called linear if the property

$$x(n) = Ax_1(n) + Bx_2(n) \rightarrow y(n) = Ay_1(n) + By_2(n)$$

is fullfilled. In all other cases it is called nonlinear. Nonlinear processing create intentional or untentional harmonic or inharmonic frenquency components which are not present in the input signal, this is what we called harmonic distorstion.

A measurement of the total harmonic distortion(THD) gives an indication of the nonliearnity of the system. Total harmonic distortion is defined by

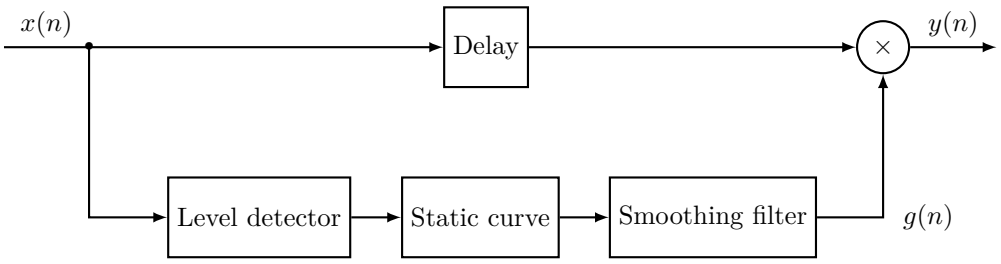
$$\text{THD} = \sqrt{\frac{A_2^2 + A_3^2 + \cdots + A_N^2}{A_1^2 + A_2^2 + \cdots + A_N^2}},$$

which is the square root of the radio of the sum of powers of all harmonic frenquency above the fundamental frequency to the power of all harmonics frequencies, including the fundmental frequency.

2 Dynamic Range Control

Dynamics processing is performed by amplifying devices where the gain is automatically controlled by the level of the input signal. We will discuss limiters, compressors, expanders and noise gates.

Dynamics processing is based on an amplitude/level detection scheme sometimes called an envelope follower, a static curve to derive a gain factor from the result of the envelope follower, a smoothing filter to prevent too abrupt gain changes and a multiplier to weight the input signal. Optionally, the input signal is delayed to compensate for any delay in the side chain. Normally, the gain factor is derived from the input signal, but the side chain path can also be connected to another signal for controlling the gain factor of the input signal.



Figure

The level detector follows one of the two approaches :

- A full-wave rectifier in combination with an AR-averager with very short attack time may be used to track the peak value of the signal.
- A square averaged with only a single time-constant provides the RMS value, measuring the signal's power.

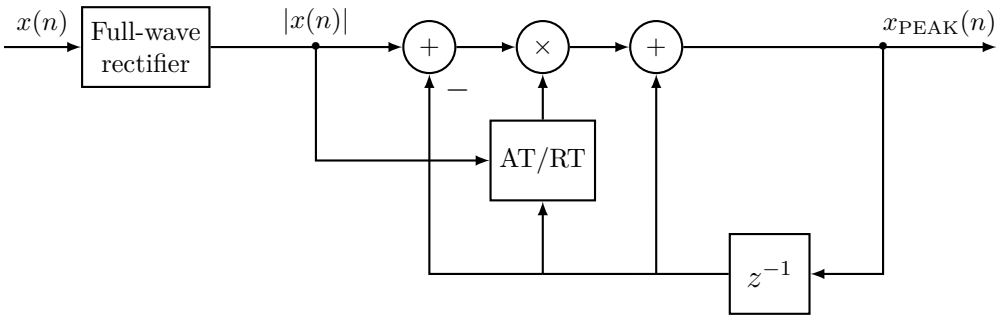


Figure Peak measurement for a dynamic range controller.

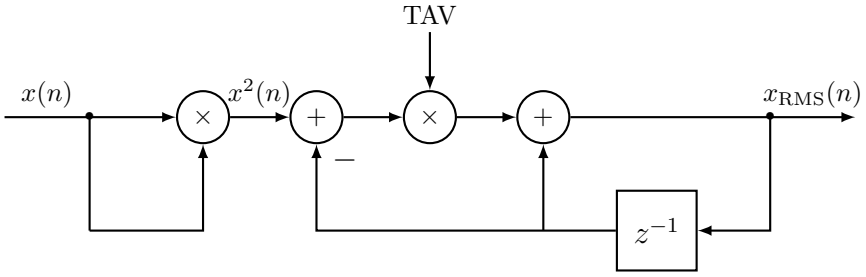


Figure RMS measurement for a dynamic range controller.

The static curve decides which function(limiter, compressor, expander,…) the dynamic range controller emploies. It is best described on a dB scale, so we introduce X, G and Y for the levels of the respective signals in dB. The multiplication at the dynamic range controller's output then becomes $Y = X + G$.

The dynamic behavior of a dynamic range controller is influenced by the level measurement approach (with attack AT and release time RT for peak measurement and averaging time TAV for RMS measurement) and further adjusted with a smoothing filter.

3 Limiter

The limiter is used to control the highest peaks in the signal, but to otherwise change the dynamics of the signal as little as possible.

$$G = \begin{cases} 0 \text{ dB} & \text{if } X < LT \\ LT - X & \text{else} \end{cases}$$

$$g(n) = \min\left(1, \frac{lt}{x_{\text{PEAK}}(n)}\right)$$

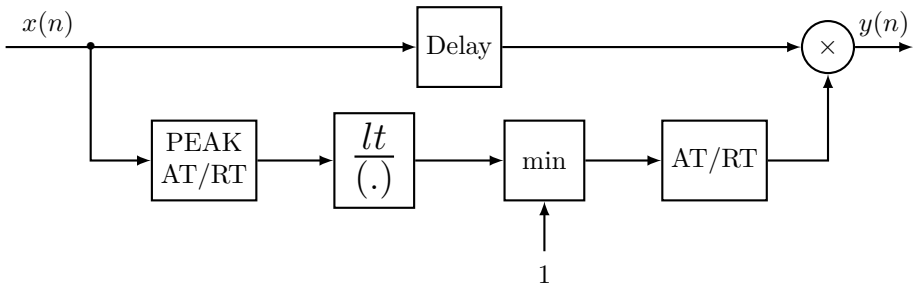


Figure Block diagram of a limiter

```
function y = limiter(x, lt)
% function y = limiter(x, lt)
% Author: Yangang Cao

at = 0.3; % Attack time
rt = 0.01; % Release time
delay = 5;

xpeak = 0; % intialization
g = 1; % gain
buffer = zeros(1,delay);

for n = 1:length(x)
    a = abs(x(n));
    if a > xpeak
        coeff = at;
    else
        coeff = rt;
    end
    xpeak = (1-coeff) * xpeak + coeff * a; % Level detector (A ...
        full-wave rectifier in combination with an AR-averager)
    f = min(1, lt/xpeak); %Static curve
    if f < g
        coeff = at;
    else
        coeff = rt;
    end
    g = (1-coeff) * g + coeff * f; % Smoothing filter
    y(n) = g * buffer(end);
    buffer = [x(n) buffer(1:end-1)];
end
```

4 Compressor and Expander

$$G = \begin{cases} CS \cdot (CT - X) & \text{if } X > CT \\ 0 \text{ dB} & \text{if } ET \leqslant X \leqslant CT \\ ES \cdot (ET - X) & \text{if } X < ET \end{cases}$$
$$= min(0,\ CS \cdot (CT - X),\ ES \cdot (ET - X)).$$

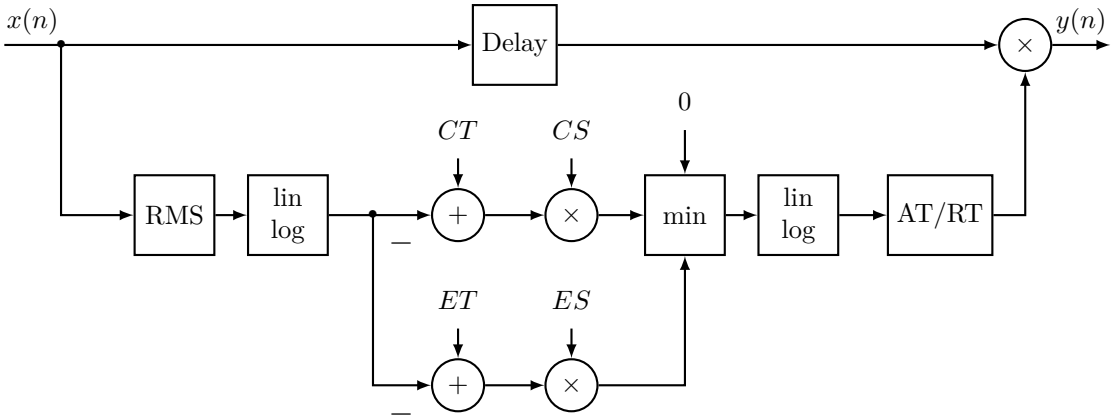


Figure Block diagram of a compressor/expander

$$g(n) = \min\left(1, \left(\frac{x_{\text{RMS}}(n)}{ct^2}\right)^{-CS/2}, \left(\frac{x_{\text{RMS}}(n)}{et^2}\right)^{-ES/2}\right),$$

```
function y = compexp(x, CT, CS, ET, ES)

tav = 0.01;
at = 0.03;
rt = 0.003;
delay = 150;

xrms = 0;
g = 1;
buffer = zeros(1,delay);

for n = 1:length(x)
    xrms = (1-tav) * xrms + tav * x(n)^2;
    X = 10*log10(xrms);
    G = min([0, CS * (CT - X), ES * (ET - X)]);
    f = 10^(G/20);
    if f < g
        coeff = at;
    else
        coeff = rt;
    end
    g = (1-coeff) * g + coeff * f;
    y(n) = g * buffer(end);
    buffer = [x(n) buffer(1:end-1)];
end
```

5 Noise Gate

A noise gate can be considered as an extreme expander with a slope of $-\infty$. This results in the complete muting of signals below the chosen threshold NT. As the name implies, the noise gate is typically used to gate out noise by setting the threshold just above the level of the background noise, such that the gate only opens when a desired signal with a level above the threshold is present. A particular application is found when recording a drum set. Each element of the drum set has a different decay time. When they are not manually damped, their sounds mix together and the result is no longer distinguishable. When each element is processed by a noise gate, every sound can automatically be faded out after the attack part of the sound. This results in an overall cleaner sound.

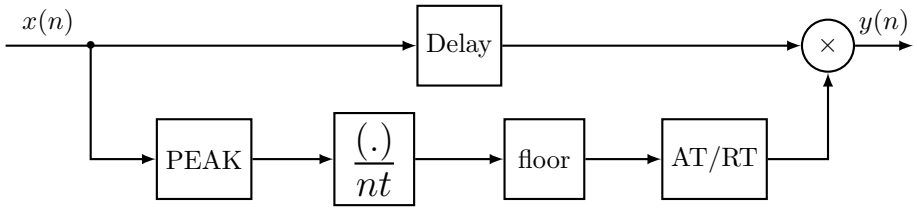


Figure Block diagram of a noise gate

The functional units of a noise gate are shown in Figure 4.16. The decision to activate the gate is typically based on a peak measurement which leads to a fade in/fade out of the gain factor g(n) with appropriate attack and release times. Further possible refinements include the use of two thresholds to realize a hysteresis and a hold time to avoid unpleasant effects when the input level fluctuates around the threshold. These are demonstrated in the implementation given in following code.

```
function y=noisegt(x,holdtime,ltrhold,utrhold,release,attack,a,Fs)

% noise gate with hysteresis
% holdtime: time in seconds the sound level has to be below the
%           threshhold value before the gate is activated
% ltrhold: threshold value for activating the gate
% utrhold: threshold value for deactivating the gate > ltrhold
% release: time in seconds before the sound level reaches zero
% attack: time in seconds before the output sound level is the
%         same as the input level after deactivating the gate
% a: pole placement of the envelope detecting filter < 1
% Fs: sampling frequency

rel = round(release*Fs);  %number of samples for fade
att = round(attack*Fs);   %number of samples for fade
g = zeros(size(x));
lthcnt = 0;
uthcnt = 0;
ht = round(holdtime*Fs);
h = filter([(1-a)^2], [1.0000 -2*a a^2], abs(x)); %envelope ...
    detection
h = h / max(h);
for i = 1: length(h)
```

```
23      if (h(i) ≤ ltrhold) | ((h(i) < utrhold) & (lthcnt > 0)) % ...
            Value below the lower threshold?
24          lthcnt = lthcnt + 1;
25          uthcnt = 0;
26          if lthcnt > ht % Time below the lower threshold longer ...
                than the hold time?
27              if lthcnt > (rel + ht)
28                  g(i) = 0;
29              else
30                  g(i) = 1 - (lthcnt-ht) / rel;       % fades the ...
                        signal to zero
31              end;
32          elseif ((i < ht) & (lthcnt == i))
33              g(i) = 0;
34          else
35              g(i) = 1;
36          end;
37      elseif (h(i) ≥ utrhold) | ((h(i) > ltrhold) & (uthcnt > 0))
38      % Value above the upper threshold or is the signal being ...
            faded in?
39          uthcnt = uthcnt + 1;
40          if (g(i-1) < 1) % Has the gate been activated or isn't ...
                the signal faded in yet?
41              g(i) = max(uthcnt/att, g(i-1));
42          else
43              g(i) = 1;
44          end;
45          lthcnt = 0;
46      else
47          g(i) = g(i-1);
48          lthcnt = 0;
49          uthcnt = 0;
50      end;
51  end;
52  y = x .* g;
53  y = y * max(abs(x)) / max(abs(y));
```