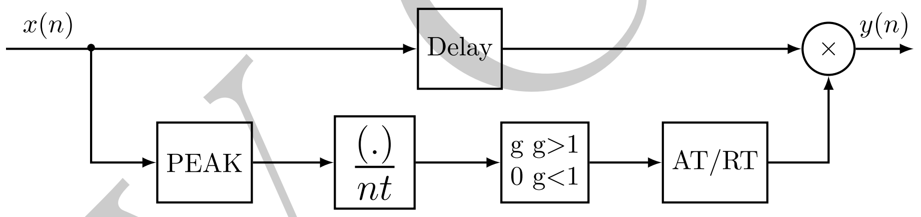
Noise Gate

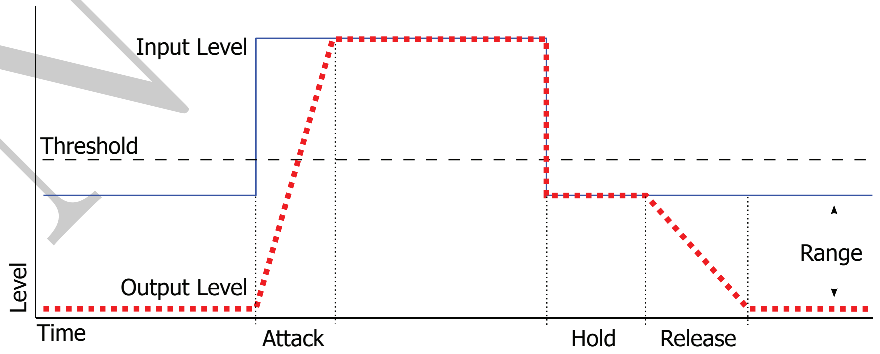
Yangang Cao

February 27, 2019

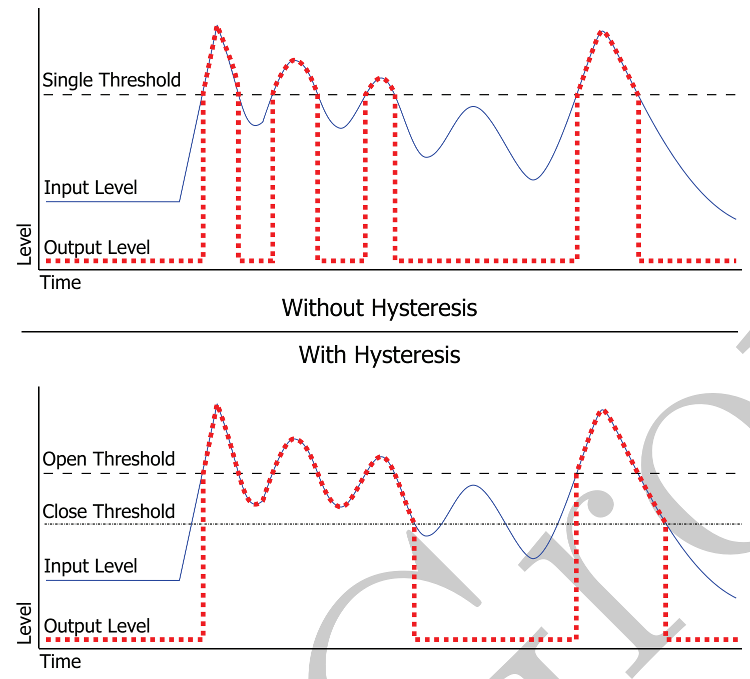
A noise gate can be considered as an extreme expander with a slope of $-\infty$. This results in the complete muting of signals below the chosen threshold NT. As the name implies, the noise gate is typically used to gate out noise by setting the threshold just above the level of the background noise, 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. The functional units of a noise gate are shown in following block diagram.



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 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(audio, para)

% noise gate with hysteresis
% para(1): time in seconds the sound level has to be below the
%           threshhold value before the gate is activated
% para(2): threshold value for activating the gate
% para(3): threshold value for deactivating the gate > para(2)
% para(4): time in seconds before the sound level reaches zero
% para(5): time in seconds before the output sound level is the
%         same as the input level after deactivating the gate
% para(6): pole placement of the envelope detecting filter < 1
% para(7): sampling frequency

rel = round(para(4)*para(7));  %number of samples for fade
att = round(para(5)*para(7));   %number of samples for fade
g = zeros(size(audio));
lthcnt = 0;
uthcnt = 0;
ht = round(para(1)*para(7));
h = filter([(1-para(6))^2], [1.0000 -2*para(6) para(6)^2], ...
    abs(audio)); %envelope detection
h = h / max(h);
for i = 1: length(h)
    if (h(i) ≤ para(2)) | ((h(i) < para(3)) & (lthcnt > 0)) % ...
        Value below the lower threshold?
        lthcnt = lthcnt + 1;
        uthcnt = 0;
        if lthcnt > ht % Time below the lower threshold longer ...
            than the hold time?
            if lthcnt > (rel + ht)
                g(i) = 0;
            else
```

```
                g(i) = 1 - (lthcnt-ht) / rel;       % fades the ...
                    signal to zero
            end
        elseif ((i < ht) & (lthcnt == i))
            g(i) = 0;
        else
            g(i) = 1;
        end
    elseif (h(i) ≥ para(3)) | ((h(i) > para(2)) & (uthcnt > 0))
    % Value above the upper threshold or is the signal being ...
        faded in?
        uthcnt = uthcnt + 1;
        if (g(i-1) < 1) % Has the gate been activated or isn't ...
            the signal faded in yet?
            g(i) = max(uthcnt/att, g(i-1));
        else
            g(i) = 1;
        end;
        lthcnt = 0;
    else
        g(i) = g(i-1);
        lthcnt = 0;
        uthcnt = 0;
    end
end
y = audio .* g;
y = y * max(abs(audio)) / max(abs(y));
```