Non-linear Processing

Non-linear Processors:

Characterised by the fact that they create (intentional or unintentional) harmonic and inharmonic frequency components **not present** in the original signal.

Three major categories of non-linear processing:

Dynamic Processing: control of signal envelope — aim to minimise harmonic distortion. Examples: **Compressors**, **Limiters**

Intentional non-linear harmonic processing: Aim to introduce strong harmonic distortion. Examples: Many electric guitar effects such as **distortion**

Exciters/Enhancers: add additional harmonics for subtle sound improvement.

Limiter

Limiter:

A device that **controls high peaks** in a signal but aims to change the dynamics of the main signal as little as possible:

- A limiter makes use of a peak level measurement and aims to react very quickly to **scale** the level if it is above some threshold.
- By lowering peaks the **overall signal** can be **boosted**.
- Limiting is used **not only** on single instrument but on **final** (multichannel) audio for CD mastering, radio broadcast *etc.*

MATLAB Limiter Example

limiter.m:

- The following code creates a modulated sine wave and then limits the amplitude when it exceeds some threshold.

```
% Create a sine wave with amplitude
% reduced for half its duration

anzahl=220;
for n=1:anzahl,
   x(n)=0.2*sin(n/5);
end;
for n=anzahl+1:2*anzahl;
   x(n)=sin(n/5);
end;
```

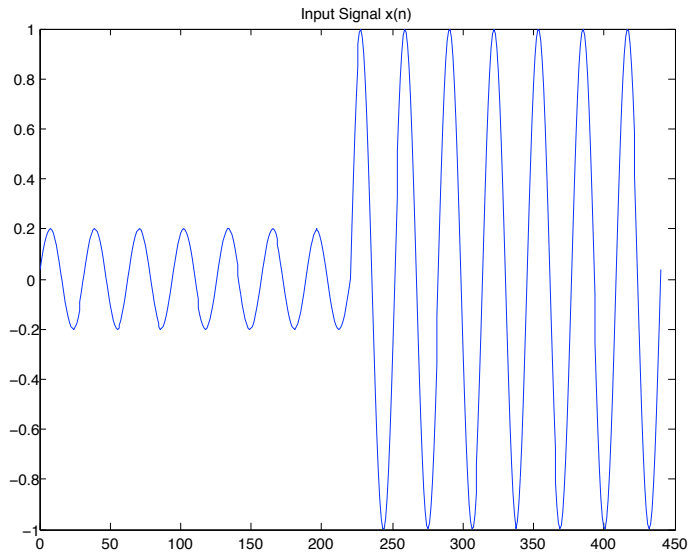
MATLAB Limiter Example (Cont.)

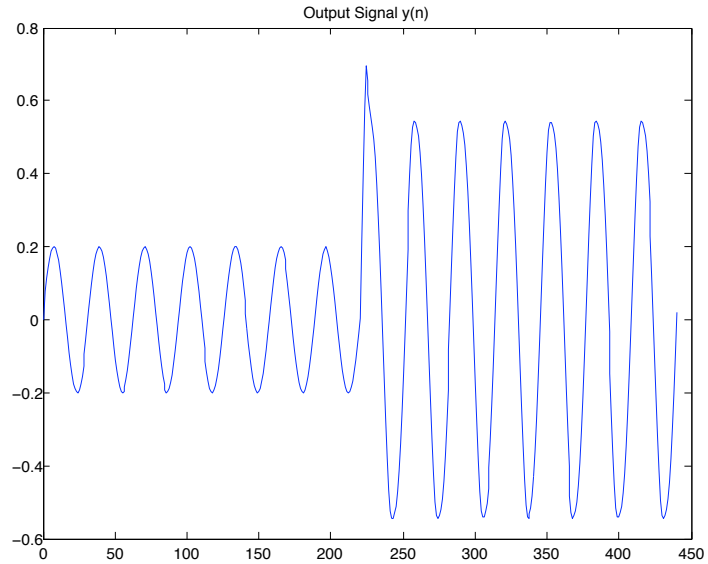
limiter.m (Cont.) :

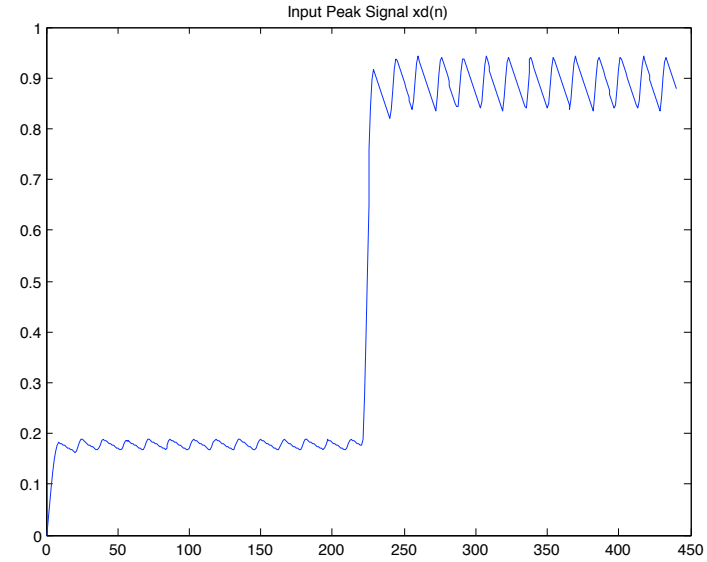
```
% do Limiter

slope=1;
tresh=0.5;
rt=0.01;
at=0.4;

xd(1)=0; % Records Peaks in x
for n=2:2*anzahl;
  a=abs(x(n))-xd(n-1);
  if a<0, a=0; end;
    xd(n)=xd(n-1)*(1-rt)+at*a;
    if xd(n)>tresh,
      f(n)=10^(-slope*(log10(xd(n))-log10(tresh)));
      % linear calculation of f=10^(-LS*(X-LT))
    else f(n)=1;
  end;
  y(n)=x(n)*f(n);
end;
```

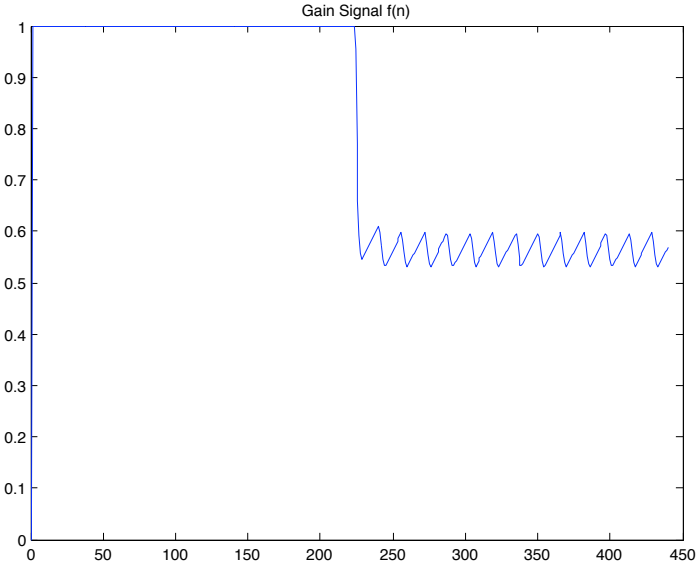
MATLAB Limiter Example Output

Display of the signals from the above limiter example:









Compressors/Expanders

Compressors:

Devices used to reduce the dynamics of the input signal:

- Quiet parts are **modified**.
- Loud parts are reduced according to some static curve.
- A bit like a limiter and used again to boost overall signals in mastering or other applications.
- Often, used on vocals and guitar effects.

Expanders:

Devices that operate on **low signal levels** and **boost** the dynamics in these signals.

- Used to create a more **lively** sound characteristic.

MATLAB Compressor/Expander

compexp.m:

```
function y=compexp(x,comp,release,attack,a,Fs)
% Compressor/expander
% comp          - compression: 0>comp>-1, expansion: 0<comp<1
% a                       - filter parameter <1
h=filter([(1-a)^2],[1.0000 -2*a a^2],abs(x));
h=h/max(h);
h=h.^comp;
y=x.*h;
y=y*max(abs(x))/max(abs(y));
```

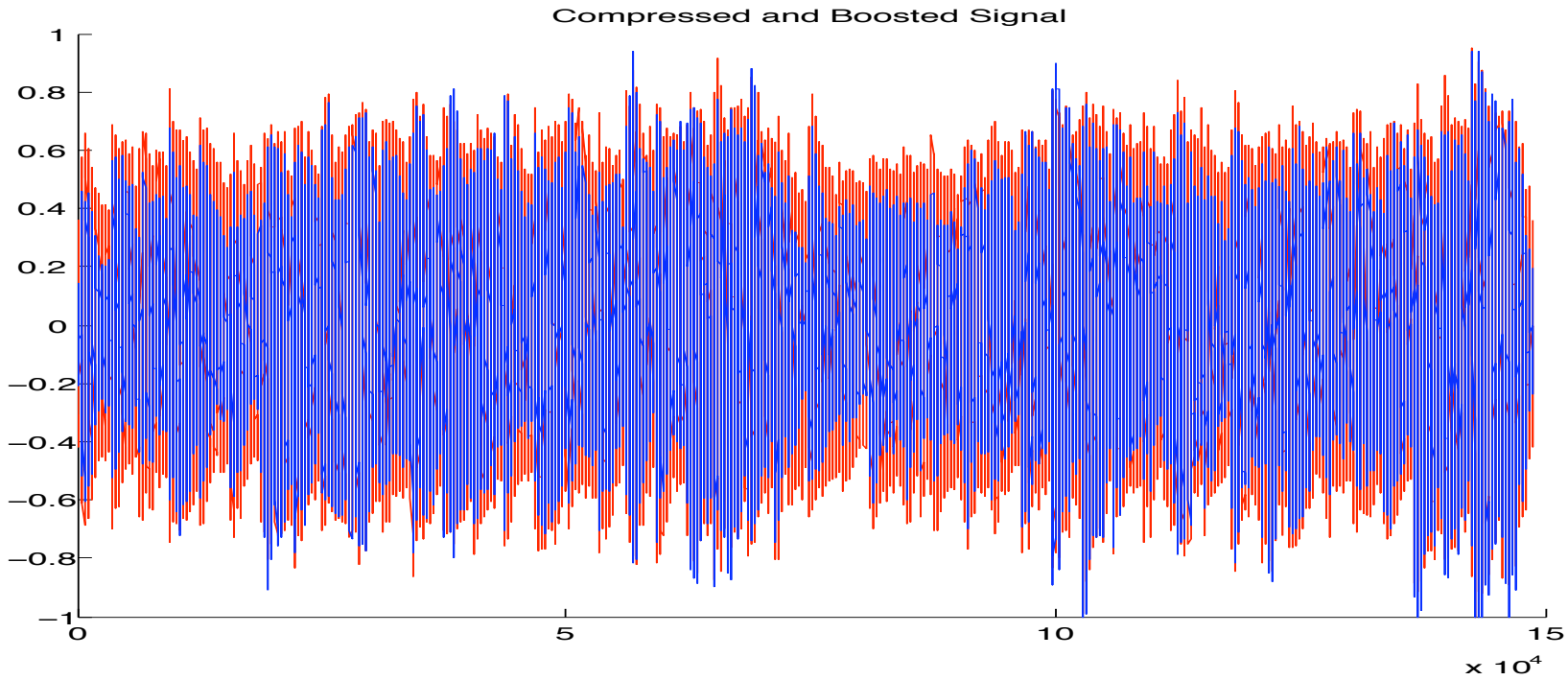
MATLAB Compressor/Expander (Cont.)

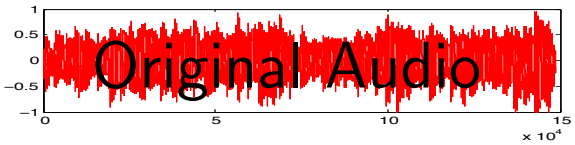
Example call: compression_eg.m:

```
% read the sample waveform
filename='acoustic.wav';
[x,Fs] = audioread(filename);

comp = -0.5; %set compressor
a = 0.5;
y = compexp(x,comp,a,Fs);

% write output
audiowrite('out_compression.wav', y,Fs,bits);

figure(1);
hold on
plot(y,'r');
plot(x,'b');
title('Compressed and Boosted Signal');
```

MATLAB Compressor Output

A **compressed signal** looks and sounds like this:





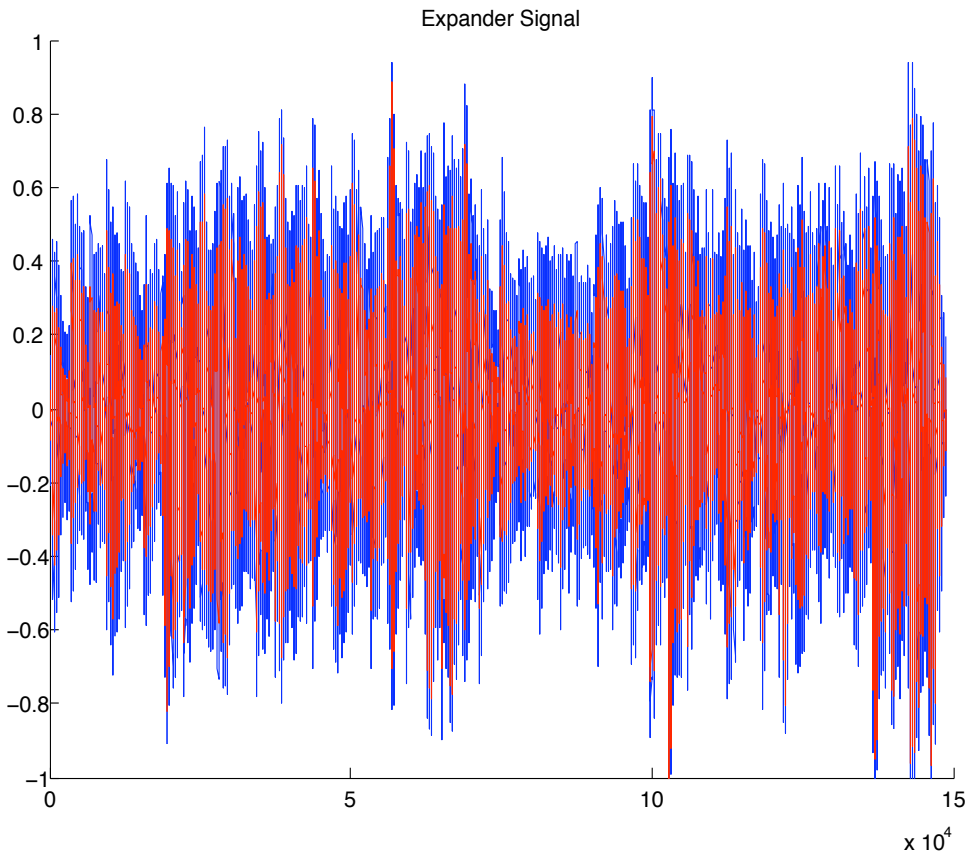
Click Image or here to hear: original audio, compressed audio.

MATLAB Expander Output

An **expanded signal** looks like this:

expander_eg.m:

```
% read the sample waveform
filename='acoustic.wav';
[x,Fs] = audioread(filename);

comp = 0.5; %set expander
a = 0.5;
y = compexp(x,comp,a,Fs);

% write output
audiowrite('out_expander.wav', y,Fs);

figure(1);
hold on
plot(y,'r');
plot(x,'b');
title('Expander Signal');
```



Click image or here to hear: original audio, expander audio.