BLOCK PROCESSING STRATEGIES FOR COMPUTATIONALLY EFFICIENT DYNAMIC RANGE CONTROLLERS

Germán Ramos,

ITACA Institue, Universitat Politècnica de València
Valencia, Spain
gramosp@eln.upv.es

ABSTRACT

This paper presents several strategies for designing Dynamic Range Controllers when using a block-based processing scheme instead of sample-by-sample processing scheme. The processes of energy measurement, gain calculus, and time constant selection are executed only once per each new incoming block of samples. Then, a simple and continuous gain update is computed and applied sample-by-sample between continuous sample blocks to achieve good sound quality and performance. This approach allows reducing the computational cost needs while maintaining the flexibility and behavior of sample-by-sample processing solutions. Several implementation optimizations are also presented for reducing the computational cost and achieving a flexible and better sounding dynamic curve using configurable soft knees or gain tables. The proposed approach has been tested and implemented in a modern DSP, achieving satisfactory results with a considerable computational costs saving.

1. INTRODUCTION

Dynamic Range Controllers (DRC) are often used in audio applications with the objective of mapping an incoming dynamic range to a different outcoming one. They are used in systems like compressors, expanders, noise-gates, limiters, or even all together in a general DRC [1].

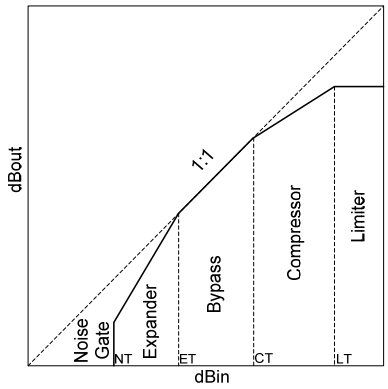

Figure 1: *Dynamic Range Controller (DRC) dB input to dB output relationship – Static Curve.*

Figure 1 shows the typical dB input to dB output relationship of a DRC that is defined by its Static Curve. It includes all the commented behaviours: noise gate, expander, compressor and limiter. The input levels NT, ET, CT, and LT are the threshold levels in which each behaviour is obtained respectively. The dB gain applied in each case is obtained as the dB difference from the bypass line (1:1) to the output dB level.

Actually most of the DRC implementations are carried out in the digital domain using Digital Signal Processors (DSP) or microprocessors. Several generic implementations have been proposed [1]-[5]. An interesting improvement that avoids the possibility of clipping the output signal in digital limiters is presented at [6]. The use on non-linearities in DRC with a power polynomial approximation is proposed at [7], [8] with the objective of simulating the non-linear behaviour of analog components like tube amplifiers [2]. [9] proposes the use of a time-varying loudness model in the level detection stage of DRC. Recently, a hardware implementation in a FPGA (Field Programmable Gate Array) of a DRC has been described at [10]. A different approach is [11] that proposes a multichannel DRC working in the frequency domain using frequency warping in order to achieve a closer behaviour to the auditory Bark scale.

This paper presents implementation strategies for DRC when working in block-sample processing schemes instead of classical sample-by-sample schemes. As the energy of the input signal has a considerable lower bandwidth that the signal itself, the processes of energy measurement, gain calculus, and time constant control, are executed only once per each new block of samples, instead of every new input sample. The gain applied is then interpolated between consecutive sample blocks to have a continuous update value and better sounding. By this way a great computational cost saving is obtained while maintaining the desired behaviour of the DRC. Several implementation optimizations are also detailed with the aim of reducing the demanded computational cost, together with a mathematical development of a configurable soft-knee characteristic for the Static Curve.

The paper is organized as follows. Section 2 makes an overview of a sample-by-sample DRC implementation. Section 3 explains the proposed modifications for a block-processing DRC. A mathematical development of a soft-knee DRC is described at Section 4. An implementation in a modern DSP is commented at Section 5. Finally Section 6 summarizes the conclusions.

2. SAMPLE PROCESSING OPERATION SCHEME

Figure 2 displays the scheme of a DRC that is executed for any new input sample *x[n]* to produce the processed output *y[n]*. An optional post-gain (not shown in the figure) could be applied after *y[n]* to move up or down the whole Static Curve of Figure 1. Following the implementation of [1], the level of the input *x[n]* is measured using a RMS detector or a peak detector, giving the input level value *xl[n]*. This level value is converted to dB and used as the input to the Static Curve to determine the output dB level and hence the needed dB gain that is converted to its linear value *g[n]*. This gain is smoothed to *gs[n]* with the Smooth Attack/Release block, which controls the dynamic behaviour of the

DRC with the proper selection of the attack and release time constants involved. See implementation details and time constant calculations and recommendation values at [1]. Finally, the smoothed gain *gs[n]* is applied to the input signal *x[n]* that could be delayed with the Look-ahead delay block in order to anticipate the behaviour of the DRC and avoiding big transients to pass without being controlled. All of these processes are executed for each new input sample, demanding computational cost.

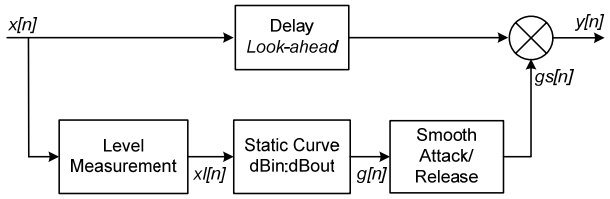

Figure 2: *Sample-by-sample DRC operation scheme.*

As commented, the RMS or peak level *xl[n]* has a considerable lower bandwidth than the incoming signal itself *x[n]*, and there is no reason in executing the dB conversion, the dB gain calculus and its conversion to linear, and the gain smoothing, for every new sample at the sampling frequency rate of the system *fs*. To save computational cost, Zölzer proposes at [1] the modified implementation scheme of Figure 3.

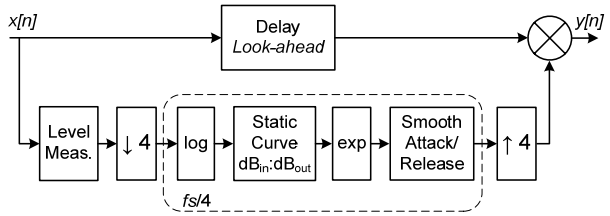

Figure 3: *Modified DRC operation scheme with down-sampling and up-sampling for the gain calculus.*

Once measured the level of the signal, a decimation by a factor 4 is applied for reducing the internal sampling frequency to *fs*/4 and making all the gain calculus processes. Then, a final interpolation by 4 is executed in order to apply the gain at *fs*. In this case, all the time constants must be calculated considering its internal *fs*/4 sampling frequency. For each new input sample *x[n]*, the level measurement, the down-sampling and up-sampling, and the gain multiplication are executed at a rate of *fs*. The four down-sampled processes (dB conversion, dB gain calculus, conversion to linear, smoother) are executed cyclically with a task scheduler, doing only once of them for each new *x[n]* cyclically. As a result, the computational cost is reduced because only one of the four processes is executed each time. This paper tries to go one step further increasing the practical decimation ratio as seen at the next section.

3. BLOCK PROCESSING PROPOSSED SCHEME

Most of the actual DSP and microprocessors used in audio, due to their internal hardware architecture and the possibility to use software pipeline techniques, are computationally more efficient when working in a frame basis in blocks of *N* samples, instead of working sample-by-sample. By this way they are also able to use the Direct Memory Access (DMA) engine to move all the audio in and out without the intervention of the CPU. Usual values for *N* in live audio applications are 16, 32 or 64. The introduced latency in samples is 2**N*. With *fs*=48kHz it is 0.67ms, 1.33ms, and 2.66ms respectively, and with *fs*=96kHz, 0.33ms, 0.67ms, and 1.33ms. This latency must be increased with the one introduced by the AD and DA converters that is usually bellow 1 ms. These latency values are considered acceptable for live use.

The proposed block-processing scheme is at Figure 4. The incoming data is the *N* samples block vector *x[n]* to *x[n-(N-1)]*. Now, only the optional look-ahead delay and the final gain interpolation and application are executed once per sample at *fs* rate. The rest of the process will be only executed once per block of *N* samples, with an effective rate of operation of *fs*/*N*.

First, the signal level of the block is measured. If an RMS detector is used, then the $x_{RMS}{}^2$ is computed as

$$x_{RMS^2} = \frac{1}{N}\sum_{i=0}^{N-1} x^2[n-i] \quad (1)$$

This $x_{RMS}{}^2$ is averaged with a first-order low-pass filter to have an energy value with an estimation time longer than *N* samples. The measured value xl_{block} is obtained with the difference equation (2) where TAV is averaging coefficient [3] and $xl_{block-1}$ is the value of xl_{block} in the previously processed block. For calculating TAV, the effective sampling frequency is now *fs*/*N*.

$$xl_{block} = (1-\mathrm{TAV})\cdot xl_{block-1} + \mathrm{TAV}\cdot x_{RMS^2} \quad (2)$$

If a peak detector is used (i.e. in a limiter case), then the maximum absolute value of the input vector is obtained and the peak detector proposed at [1] and [3] is used.

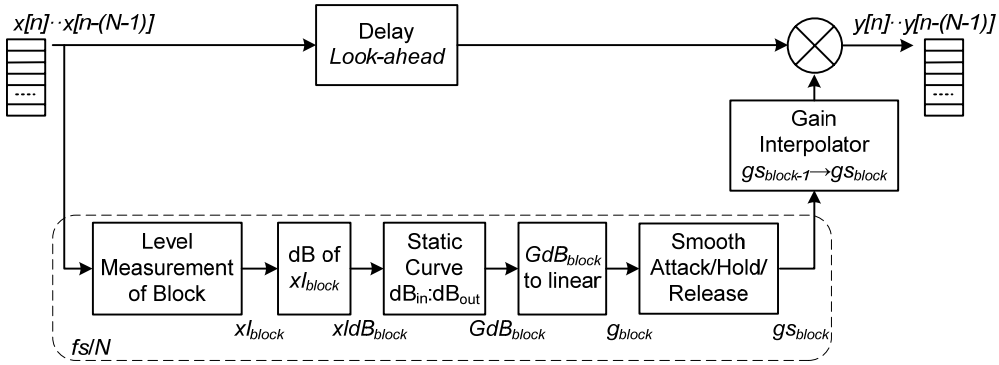

Figure 4: *Proposed block-sample DRC operation scheme.*

The measured level xl_{block} is now converted to $xldB_{block}$ in dB, using a fast approximation by series expansion [1] or a log table in memory. In case of RMS detector, xl_{block} is the squared RMS measurement and the obtained $xldB_{block}$ must be multiplied by 0.5 to calculate the square-root and achieve the RMS value.

The dB value of the input signal $xldB_{block}$ is used as input to the Static Curve (i.e. like Fig. 1) to determine de output dB level and hence the dB gain to be applied GdB_{block} as the difference between the output dB value and the input dB value. One possibility to implement the Static Curve is using the simple linear equations of the lines of Fig. 1 in each region of operation (noise-gate, expander, bypass, compressor, limiter) using the threshold and ratio values of each region [1]. A more flexible choice is using a dB table in memory, mapping the incoming dB level directly to the applied dB gain. This allows the creation of any kind of Static Curve like the one seen at Figure 5. In this example, the dB gain table is defined every 3 dB, obtaining the dB gains between the table values by interpolation.

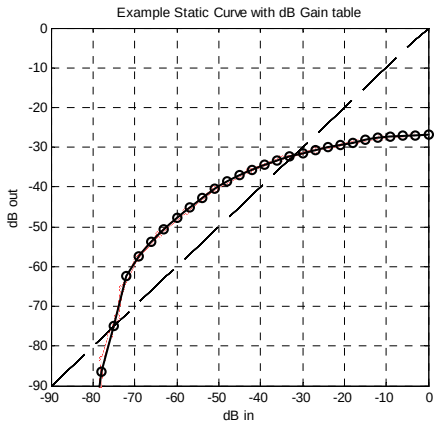

Figure 5: *Example of Static Curve implemented with a dB gain table defined with 3dB steps.*

The GdB_{block} value is converted to its lineal value g_{block} using again a series expansion or an antilog table. Finally this gain is smoothed to gs_{block} with the Attack/Hold/Release block with equation (3) where $gs_{block-1}$ is the gain of the previously processed block of samples. It controls the dynamic behavior of the DRC with the selection of the time constants *AT* (Attack Time, signal level increases) and *RT* (Release Time, signal level decreases).

$$\begin{aligned} gs_{block} &= (1-k)\cdot gs_{block-1} + k\cdot g_{block} \\ g_{step} &= gs_{block} - gs_{block-1} \\ k &= \begin{cases} AT \text{ if } g_{step} < 0 \\ RT \text{ if } g_{step} \geq 0 \end{cases} \end{aligned} \tag{3}$$

A Hold Time *HT* is included for controlling the time that must stay in the release state before the gain starts to recover, maintaining its gain value. This avoids continuous gain changes and decreases the distortion introduced by the DRC. A good option is using a counter for the *HT*, being each count value the time of a block of samples of *N/fs* seconds.

Once calculated the gain gs_{block} for each new input *N* samples block, it must be applied to the input vector *x[n:n-(N-1)]* to give the output vector *y[n:n-(N-1)]*. This gain, calculated at a rate of *fs/N*, is up-sampled (smoothed) to *fs* with the Gain Interpolator block. It performs a linear interpolation between $gs_{block-1}$ to gs_{block} just adding to the applied gain the gain step g_{step}/N for each new output sample. A simulation of a limiter with this interpolation is at Figure 6 where the calculated gs_{block} every *N* samples is displayed (the stepped gain) together with the linearly interpolated smoothed gain. More complex interpolation methods like splines could be used at expense on increasing the computational cost.

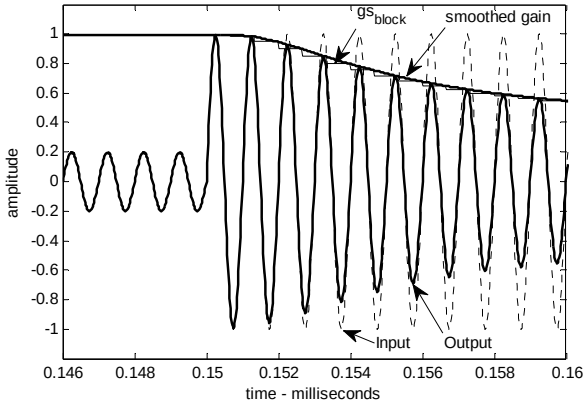

Figure 6: *Example of a Limiter with the proposed gain interpolation between consecutive N samples blocks.*

With conventional audio content, the proposed block-processing DRC implementation with *N* values lower than 1ms (i.e. *N*=32 with *fs*=48kHz is 0.66ms.), achieves quasi-identical sounding results than a sample-by-sample DRC implementation, with a considerable reduction in computational cost. The effect of the block-sample processing is a slight increase in the effective time constants that are low-limited by the *N* value. Usually a value greater than 1ms is used for the attack-time. For greater *N* values, the input vector must be split in smaller blocks and repeat the process once per each smaller block. Other ways the time constants are excessively smeared and it will not be possible to work with short attack times below *N/fs* that are needed for example in limiters or when working with high frequency signals.

4. SOFT-KNEE STATIC-CURVE DEVELOPMENT

Figure 1 shows a typical Static Curve made of straight lines on each section with sharp transitions (sharp gain changes) between regions (i.e. bypass to compressor). These transitions are referenced as hard-knee. Some DRC use what is called soft-knee transitions that vary progressively from the slope of one region to the slope of the next one. This section describes a procedure to design configurable soft-knee links between consecutive regions of a Static Curve using a parabola as the link function.

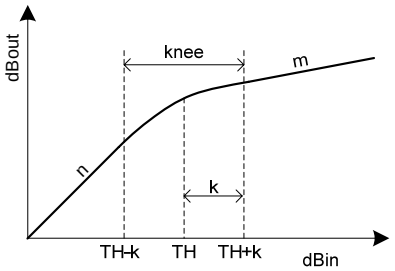

Figure 7: *Soft-knee Static Curve design.*

Figure 7 shows a soft-knee link of width *knee=2*k* between two regions of slopes *n* and *m* (the ratio is the inverse of the slope). The threshold dB input is *TH*, and the soft-knee link is carried out within the input region *TH-k* to *TH+k* dB. The parabola (4) is used as the link function and its coefficients are calculated solving the equation system that equals the derivative of the parabola to *n* at *TH-k*, to *m* at *TH+k*, and forces the value of the parabola at *TH-k* or *TH+k*. Once defined the soft-knee Static Curve, the gain in dB is obtained again as the difference between the output and input dB levels. An example of a soft-knee limiter (*n*=1, *m*=0, *TH*=-12dB) with different *k* values (from 0 to 10 dB) is displayed at Figure 8. This soft-knee limiter configuration allows arriving to the limit value gradually, not instantly as happens with hard-knee limiters. It works like a compressor that increments continuously its ratio from 1:1 to ∞:1 in 2·*k* dB input range, and it is judged to have a better sounding behaviour.

$$\begin{aligned} dBout &= a + b \cdot dBin + c \cdot dBin^2 \\ a &= -1/(4 \cdot k) \cdot \left(2 \cdot TH \cdot k \cdot (n+m-2) + (TH^2 + k^2) \cdot (n-m)\right) \\ b &= 1/(2 \cdot k) \cdot \left(TH \cdot (n-m) + k \cdot (n+m)\right) \\ c &= -1/(4 \cdot k) \cdot (n-m) \end{aligned} \quad (4)$$

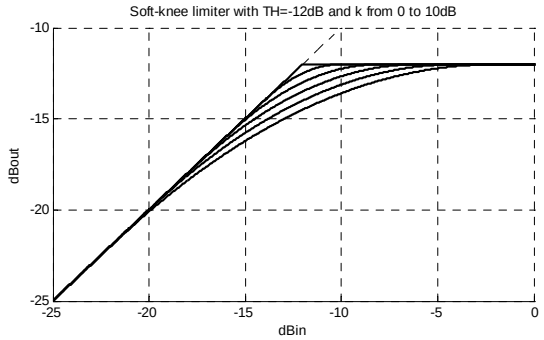

Figure 8: *Example of a Limiter with Soft-knee.*

5. DSP IMPLEMENTATION

An efficient implementation of the proposed block-based DRC with *N*=32 and *fs*=48kHz has been carried out in a SHARC ADSP21489 DSP [12] taking profit of its SIMD (Single Instruction Multiple Data) architecture, and its DMA engine.

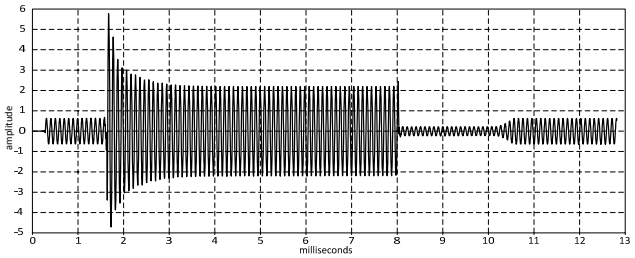

Figure 9: *Behavior of the proposed DRC on a DSP with a sinusoidal burst as input*

The implementation was able to work with Static Curves defined by lines as Figure 1 with or without soft-knee, and with a gain table defined by the user. The percentage of CPU used by the DRC was of only 0.25%, meanwhile a sample-by-sample version of the DRC took 2.1%. The behavior of the DSP implementation with a sinusoidal burst input signal is shown at Figure 9. It is easy to observe the level control of the DRC and the effect of the attack, release and hold times.

6. CONCLUSION

An efficient implementation of a DRC is proposed in this paper that exploits the benefits of working in blocks of *N* samples instead of processing sample-by-sample. Most of the processes involved in a DRC are modified and executed only once per new block of samples, and only the final gain smoothing and the application of the gain is executed once per sample. This allows saving computational cost while maintaining the DRC behaviour and sound properties. A generic soft-knee link parabola is also presented. Finally, the proposed DRC has been implemented and tested in an actual DSP verifying the computational cost saving.

7. ACKNOWLEDGEMENTS

This paper has been supported in part by the project PAID-05-10 of the Universitat Politècnica de Valencia, Spain. The author wishes to acknowledge Analog Devices Inc. for the tools and support in the DSP implementation, and to the reviewers for their valuable comments.

8. REFERENCES

[1] U. Zölzer, *Digital Audio Signal Processing*, John Wiley and Sons, New York, second edition, 2008

[2] U. Zölzer, *DAFX:Digital Audio Effects,* John Wiley and Sons, New York, 2002

[3] G. W. McNally, "Dynamic Range Control of Digital Audio Signals," *J. Audio Eng. Soc*, vol. 32, no. 5, pp. 316–327, May 1984.

[4] E. F. Stikvoort, "Digital Dynamic Range Compressor for Audio", *J. Audio Eng. Soc*, vol. 34, no. 1/2, pp. 3–9, Jan./Feb. 1986.

[5] S. J. Orfanidis, *Introduction to Signal Processing*, Prentice Hall, New York, 1996.

[6] P. Hämäläinen, "Smoothing of the Control Signal without Clipped Output in Digital Peak Limiters", in *Proc. Digital Audio Effects (DAFx'02),* Hamburg, Germany, Sep. 2002.

[7] J. Schimmel, "Using Nonlinear Amplifier Simulation in Dynamic Range Controllers", in *Proc. Digital Audio Effects (DAFx'03),* London, UK, Sep. 2003.

[8] J. Schimmel, "Non-linear Dynamics Processing", in *Proc. 114th Convention of the Audio Engineering Society*, 2003

[9] R. J. Cassidy, "Dynamic range compression of audio signals consistent with recent time-varying loudness models," in *Proc. Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on* , vol.4, no., pp. iv-213- iv-216 vol.4, 17-21 May 2004

[10] A. De Stephanis, M. Conti, M. Caldari, F. and Ripa, "Design refinement for the development of an audio dynamic range controller," in *Proc. Intelligent Solutions in Embedded Systems (WISES), 2010 8th Workshop on* , vol., no., pp.85-90, 8-9 July 2010.

[11] J. M. Kates, K. H. Arehart, "Multichannel Dynamic-Range Compresion Using Digital Frequency Warping", EURASIP *Journal on Applied Signal Processing*, vol. 2005, pp. 3003-3014, 2005

[12] http://www.analog.com/en/embedded-processing-dsp/SHARC/ADSP-21489/processors/product.html