REAL-TIME STATE-SPACE PARAMETERIZATION IN DIGITAL EQUALIZATION: A PRODUCTION-GRADE SVF IMPLEMENTATION

Gary Doman
FreeEQ8 / ProEQ8 Open-Source DSP Project
https://github.com/GareBear99/FreeEQ8

Abstract

We present a production-grade parametric equalizer implementing the Simper State Variable Filter (SVF) topology to eliminate high-frequency magnitude cramping without oversampling. The system achieves exact cutoff placement at 16 kHz (where traditional RBJ biquads exhibit 199% Q distortion) while consuming only 0.62% of a single-core CPU budget at 44.1 kHz. A lock-free SPSC triple-buffer isolates the audio thread from UI rendering, and a variable-cadence coefficient engine reduces Dynamic EQ overhead by 80%. The implementation is available as FreeEQ8 (8-band, GPL-3.0) and ProEQ8 (24-band, commercial) from a single codebase. Live demonstration accompanies this paper.

1 Introduction

Traditional parametric EQ implementations use Robert Bristow-Johnson's Audio EQ Cookbook biquad formulas [2] in Transposed Direct Form II (TDF-II). While mathematically correct, the bilinear transform introduces frequency cramping near Nyquist: a Bell filter at 16 kHz with $Q = 1.0$ at 44.1 kHz exhibits an effective bandwidth **199% narrower** than intended.

Industry solutions include brute-force oversampling (adding latency and CPU cost) or proprietary polynomial analog-matching curves. This paper documents a third path: the Simper SVF topology [1], which pre-warps the cutoff frequency via $g = \tan(\pi \cdot f_c/f_s)$, achieving exact cutoff placement at any frequency up to Nyquist without oversampling.

2 Filter Topology

2.1 RBJ TDF-II (FreeEQ8)

The transfer function follows the standard biquad form:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{1}$$

Measured Q distortion at 44.1 kHz sample rate is shown in Table 1.

Table 1: RBJ Q distortion (Bell, $Q = 1.0$, 44.1 kHz)

Frequency	Effective Q	Error
1 kHz	1.005	+0.5%
8 kHz	1.337	+33.7%
16 kHz	2.990	+199%

2.2 Simper SVF (ProEQ8)

The SVF uses pre-warped coefficients [1]:

$$g = \tan(\pi \cdot f_c/f_s) \tag{2}$$
$$k = 1/Q \tag{3}$$
$$a_1 = 1/(1 + g \cdot (g + k)) \tag{4}$$

Per-sample processing follows the optimized bounded form:

```
v3 = v0 - ic2eq;
v1 = a1*ic1eq + a2*v3;  // bandpass
v2 = ic2eq + a2*ic1eq + a3*v3; // lowpass
ic1eq = v1 + v1 - ic1eq;
ic2eq = v2 + v2 - ic2eq;
out = m0*v0 + m1*v1 + m2*v2;
```

All eight filter types (Bell, LowShelf, HighShelf, LP, HP, Bandpass, Notch, AllPass) emerge from a single two-integrator core with different output mix coefficients (m_0, m_1, m_2).

2.3 Measured Magnitude Response

To validate de-cramping, we swept a sine (20 Hz–20 kHz) through Bell filters (+6 dB, $Q = 1.0$) at $f_c = 16$ kHz. Results in Table 2.

Table 2: Magnitude at $f_c = 16$ kHz, 44.1 kHz sample rate

Topology	Magnitude	Error
RBJ @ 44.1 kHz	+0.73 dB	−5.27 dB
SVF @ 44.1 kHz	+6.00 dB	0.00 dB
RBJ @ 4× OS	+4.82 dB	−1.18 dB

The SVF achieves exact gain without oversampling. RBJ@4×OS reduces error but costs 4× CPU.

3 Real-Time Safety Architecture

3.1 SPSC Triple-Buffer

Three buffer slots indexed by {writeSlot, midSlot, readSlot}. Audio thread writes into writeSlot; atomically swaps with midSlot using memory_order_release. UI thread reads by swapping midSlot ↔ readSlot with memory_order_acquire. Zero mutex, zero blocking.

Stress testing on Intel Ivy Bridge (2012): **0 data tears across 239M samples**.

3.2 Variable-Cadence Dynamic EQ

Dynamic EQ recomputes coefficients per-sample. We observe that during sustained notes, envelope changes of < 0.1 dB per sample are inaudible within a 4-sample batch (0.09 ms at 44.1 kHz).

```
delta = |dynGainMod - lastDynGainMod|
if delta > 0.1 dB:
    update coefficients (per-sample)
else if intervalCounter++ >= 4:
    update coefficients (batched)
```

Measured savings: **80% reduction** in coefficient updates with no audible difference. Transients immediately restore per-sample accuracy.

4 Benchmarks

All benchmarks from standalone C++17, g++ -O3, 44.1 kHz, 512-sample blocks.

Table 3: Single-instance filter cost (8-band stereo)

Path	ns/sample	CPU%	Headroom
RBJ 8-band	41.0	0.36%	277×
SVF 8-band	72.7	0.63%	161×
SVF DynEQ	68.8	0.61%	165×

Instance scaling (1→128 instances): per-instance cost rises only 5%. At 128 SVF instances, total CPU = 85.8% of one core. A modern 8-core CPU can host ∼900 instances.

5 Product Architecture

The codebase produces two plugins via #if PROEQ8:

FreeEQ8 (GPL-3.0): 8-band RBJ biquad. Zero real-time restrictions. Offline export limited to 4:30.

ProEQ8 ($20): 24-band SVF with de-cramped response. Adds saturation modes, A/B comparison, auto-gain, collision detection. No export limit.

Both validated via pluginval at strictness-level-10 on macOS/Linux/Windows CI.

6 Demonstration

The live demonstration showcases:

1. Side-by-side RBJ vs SVF response at 16 kHz
2. Real-time spectrum analyzer with lock-free rendering
3. Dynamic EQ envelope tracking on transient material
4. 24-band surgical EQ workflow in ProEQ8

Source code, benchmarks, and builds available at: https://github.com/GareBear99/FreeEQ8

7 Conclusion

We have demonstrated that the Simper SVF topology eliminates high-frequency cramping while maintaining sub-1% CPU overhead. The lock-free architecture ensures glitch-free operation across all tested DAWs, and variable-cadence Dynamic EQ reduces computational cost by 80% without perceptual degradation.

References

[1] A. Simper, "Solving the continuous SVF equations using trapezoidal integration and equivalent currents," Cytomic, 2013. https://cytomic.com/files/dsp/SvfLinearTrapOptimised2.pdf

[2] R. Bristow-Johnson, "Audio EQ Cookbook," musicdsp.org, 1994. https://www.musicdsp.org/files/Audio-EQ-Cookbook.txt

[3] W. Pirkle, *Designing Audio Effect Plugins in C++*, Focal Press, 2019.

[4] J. Reiss and A. McPherson, *Audio Effects: Theory, Implementation and Application*, CRC Press, 2014.