Advanced analog filter modeling

SVF implementation with non-linear characteristics

Technical paper explaining various methods used in Filterbank fx plugin

by Mathijs Maas (2025)

Abstract

This paper presents an advanced digital implementation of analog filter characteristics, specifically designed to emulate the distinctive sound qualities of legendary analog hardware filters. Our approach combines a State Variable Filter (SVF) architecture with several non-linear processing techniques including diode pair simulation, resonance feedback modeling, and capacitor charge/discharge behavior. The result is a filter that produces the warmth, distortion characteristics, and self-oscillation behavior of analog hardware while maintaining the flexibility and reliability of digital processing. We demonstrate how these techniques combine to create a more authentically "analog" sound compared to traditional biquad filter implementations, and provide empirical evidence of the sonic differences.

1. Introduction

Digital filters have long attempted to replicate the distinctive characteristics of analog hardware filters. While digital filters offer precision and stability, they often lack the character that comes from the non-linearities and component interactions present in analog circuits. This paper describes our implementation of a digital filter that incorporates these analog characteristics through several techniques:

1. State Variable Filter (SVF) topology for accurate analog filter response
2. Non-linear feedback path modeling with diode-pair simulation
3. Dynamic resonance feedback for authentic self-oscillation
4. Capacitor charge/discharge simulation
5. Asymmetric distortion for harmonic enhancement
6. Thermal drift and noise modeling for realistic behavior
7. Analog-inspired gain staging and compensation

Our implementation builds upon the theoretical foundation established by previous work in this area, particularly Zavalishin's "The Art of VA Filter Design" [1] and Stilson & Smith's "Analyzing the Moog VCF" [2], while incorporating novel approaches to component-level behavior simulation.

2. SVF implementation methodology

2.1 Basic SVF architecture

The State Variable Filter (SVF) is an ideal starting point for analog modeling due to its topology closely matching analog circuits with multiple filter outputs (lowpass, bandpass, highpass) available simultaneously. Unlike direct-form biquad filters, the SVF maintains separate state variables that correspond to physical elements in analog circuits.

Our implementation uses a modified Chamberlin SVF architecture, enhanced with digital optimizations based on modern SVF techniques [3]. The filter's core state variables represent the integrator capacitor values in the equivalent circuit. This approach allows us to model the signal flow through integrators in a way that closely resembles analog counterparts, particularly important for accurate resonance behavior.

2.2 Coefficient calculation

The filter coefficients are calculated based on cutoff frequency and resonance parameters, with bilinear transform prewarping to ensure accurate frequency response. Filter output mixing is controlled by coefficients that determine the filter mode (lowpass, bandpass, highpass), allowing flexible configuration while maintaining the characteristic response of the underlying SVF architecture.

3. Modeling analog diode behavior

3.1 Diode pair simulation

A key element in analog filters is the non-linear behavior of components, particularly in the feedback path. We simulate this using a diode pair model that approximates the voltage/current relationship of silicon diodes. This function models the characteristic that diodes conduct current differently depending on whether the voltage is above or below a threshold, creating asymmetric distortion that enriches the harmonic content of the signal.

A simplified implementation of this diode simulation might look like this:

```
// Simplified diode pair simulation
double diodePair(double x, double threshold, double curve) {
    if (fabs(x) < threshold)
        return x;  // Linear region below threshold

    double sign = (x > 0) ? 1.0 : -1.0;
    double over = fabs(x) - threshold;
    // Logarithmic response above threshold
    return sign * (threshold + curve * log(1.0 + over / curve));
}
```

This creates the characteristic non-linear response curve of analog diodes, which is essential for authentic resonance behavior.

3.2 Integration in the feedback path

The diode simulation is applied to the feedback path of the SVF, specifically at the point where the output signal is fed back to the input. This closely resembles how analog filter circuits use diodes to control the resonance behavior. A non-linearity factor parameter controls the degree of non-linearity, allowing a smooth transition between a clean digital response and a more analog-like behavior. This is particularly important at high resonance settings, where analog filters exhibit their most characteristic behaviors.

4. Resonance feedback system

4.1 Dynamic resonance modulation

Real analog filters exhibit complex interactions between resonance and other filter parameters. We model this by implementing a dynamic resonance modulation system that adjusts the resonance based on modulation sources (LFO, envelope) while also applying a non-linear dampening that prevents excessive gain at high resonance settings, mimicking the saturation behavior of analog circuits.

4.2 Moog-style resonance feedback

Our implementation incorporates advanced resonance feedback techniques inspired by the Moog ladder filter design. We implement a recursive resonance system that closely emulates the behavior of analog Moog-style filters. This sophisticated approach uses a non-linear gain function inspired by the Moog ladder filter's transfer function:

```
// Moog-style non-linear gain function
double moogGain(double x) {
    return x * (((x * x + 105) * x * x + 945) /
                ((15 * x * x + 420) * x * x + 945));
}
```

This gain function has a characteristic curve that produces the distinctive "Moog sound" by shaping the resonance response. The recursive nature of the feedback calculation simulates how analog circuits build up resonance over time, creating a more organic self-oscillation behavior.

4.3 Resonance delay modeling

To further emulate analog behavior, we implement a slight delay in the resonance response, creating a more organic feeling when parameters are modulated. This simple but effective technique creates a more natural response to rapid parameter changes, avoiding the clinical immediacy of typical digital filters. A small buffer allows resonance values to lag slightly behind parameter changes, simulating the subtle component-level delays present in analog circuits.

5. Capacitor charge/discharge simulation

5.1 RC circuit model

The distinctive response of analog filters is partly due to the capacitor charge and discharge characteristics in their circuits. We model this behavior using a simplified RC circuit simulation with different time constants for charging and discharging phases:

```
// Simplified capacitor simulation
double processCapacitor(double input, double voltage,
                        double chargeRate, double dischargeRate) {
    if (fabs(input) > fabs(voltage)) {
        // Charging - asymptotic approach to input value
        return voltage + chargeRate * (input - voltage);
    } else {
        // Discharging - exponential decay
        return voltage * dischargeRate;
    }
}
```

This asymmetry in the charging and discharging rates creates the characteristic non-linear response curves associated with analog filters.

5.2 Enhanced switched capacitor behavior

To more accurately model the behavior of capacitors in analog circuits, we implement a switched capacitor model that simulates the charging and discharging phases with different time constants. This model simulates how analog circuits naturally switch between charging and discharging states. The capacitor model even includes sign-dependent reset behavior, accurately capturing how real capacitors respond to signal polarity inversions.

5.3 Dynamic parameter adjustment

The capacitor behavior parameters are dynamically adjusted based on filter settings, particularly resonance and cutoff frequency. These dynamic adjustments create subtle variations in response that contribute to the organic, analog sound quality of the filter. Higher resonance settings, for example, trigger more pronounced capacitor effects, creating the characteristic "squeeze" heard in analog filters.

6. Non-linear saturation and overdrive

6.1 Signal path saturation

To emulate the subtle saturation that occurs in analog circuits, we implement a soft saturation function based on the hyperbolic tangent curve. This function is applied at various points in the signal path, particularly at high signal levels, creating a natural compression effect similar to analog circuits.

6.2 Harmonic enhancement via asymmetric distortion

A characteristic feature of analog circuits is their harmonic enhancement through asymmetric distortion. Our implementation modifies positive and negative signal excursions differently, creating the even harmonics that are characteristic of analog hardware, adding warmth and richness to the sound.

6.3 DC offset management

Like real analog circuits, our implementation deliberately introduces and manages DC offset to enhance asymmetric distortion characteristics. The DC offset is carefully controlled, with a smooth ramp to target values to avoid unwanted clicks when parameters change. This offset is later removed via a high-pass filter, similar to how coupling capacitors function in analog circuits.

7. Thermal drift and noise modeling

7.1 Total Harmonic Distortion (THD) drift simulation

Our implementation models the subtle thermal drift that occurs in analog circuits using a low-frequency modulation (LFO) that slowly varies the distortion characteristics:

```
// Simplified THD drift implementation
double applyThdDrift(double sample, double phase, double driftAmount) {
    // Slowly varying modulation factor
    double driftModulation = sin(phase);

    // Asymmetric processing based on signal polarity
    if (sample >= 0.0) {
        // Process positive values differently from negative
        sample += (1.0 + 0.15 * driftModulation) *
                  sample * sample * driftAmount;
    } else {
        sample -= (0.85 + 0.15 * driftModulation) *
                  sample * sample * driftAmount;
    }

    return sample;
}
```

This creates the “breathing” quality associated with high-quality analog hardware, where component behavior subtly shifts over time due to thermal effects.

7.2 Adaptive noise generation

We enhance realism by implementing a multi-stage filtered noise generator that produces authentic pink noise:

```
// Simplified pink noise generator
double generatePinkNoise(double white, double* b) {
    // Six-stage filter creating 1/f spectrum
    b[0] = 0.99886 * b[0] + white * 0.0555179;
    b[1] = 0.99332 * b[1] + white * 0.0750759;
    b[2] = 0.96900 * b[2] + white * 0.1538520;
    b[3] = 0.86650 * b[3] + white * 0.3104856;
    b[4] = 0.55000 * b[4] + white * 0.5329522;
    b[5] = -0.7616 * b[5] - white * 0.0168980;

    double pink = b[0] + b[1] + b[2] + b[3] + b[4] + b[5] + white * 0.5362;
    return pink * 0.11; // Scale to appropriate level
}
```

More importantly, the noise is applied dynamically based on signal levels, creating more pronounced noise on signal peaks - a characteristic behavior of analog circuits where higher signal levels excite more thermal noise in components.

7.3 Weight-based distortion coloration

Our implementation uses pre-computed weight tables to influence the distortion character, creating a distinctive sonic signature. These weights model the unique component interactions in analog circuits, where specific frequency bands exhibit characteristic distortion profiles.

8. Conclusion

Our implementation demonstrates that a carefully designed digital filter can capture many of the desirable characteristics of analog hardware filters. The combination of SVF architecture, non-linear feedback, resonance modeling, capacitor simulation, and adaptive distortion results in a filter that sounds convincingly "analog" while maintaining the precision and flexibility of digital processing.

Future work will focus on extending these techniques to model specific classic filter circuits more precisely, implementing component-level variation to simulate manufacturing tolerances, and exploring more sophisticated diode and transistor models.

References

[1] Zavalishin, V. (2012). "The Art of VA filter design." Native Instruments.

[2] Stilson, T., & Smith, J. (1996). "Analyzing the Moog VCF with considerations for digital implementation." Proceedings of the international Computer Music conference.

[3] Simper, A. (2013). "SVF linear trap optimised." Cytomic technical paper. This implementation of the State Variable Filter forms the basis of our SVF architecture.

[4] Voipio, T. (Mystran). "Non-linear Filter Techniques and Circuit Emulation." Signaldust. The Moog-style non-linear feedback and resonance modeling techniques were inspired by this work.

[5] Välimäki, V., & Huovilainen, A. (2006). "Oscillator and filter algorithms for virtual analog synthesis." Computer Music journal, 30(2), 19-31.

[6] De Sanctis, G., & Sarti, A. (2009). "Virtual analog modeling in the wave-digital domain." IEEE transactions on audio, speech, and language processing, 18(4), 715-727.

[7] Rollins, J.G., Bendix, P. (2000). "Computer software for circuit analysis and design" in *The electrical engineering handbook*, Ed. Richard C. Dorf. Boca Raton: CRC Press LLC.

[8] McCalla, W.J. (1988). *Fundamentals of computer-aided circuit simulation*. Springer.