EFFICIENT POLYNOMIAL IMPLEMENTATION OF THE EMS VCS3 FILTER MODEL

Stefano Zambon
Dipartimento di Informatica
Università di Verona
Italy
stefano.zambon@univr.it

Federico Fontana
Dipartimento di Matematica e Informatica
Università di Udine
Italy
federico.fontana@uniud.it

ABSTRACT

A previously existing nonlinear differential equation system modeling the EMS VCS3 voltage controlled filter is reformulated here in polynomial form, avoiding the expensive computation of transcendent functions imposed by the original model. The new system is discretized by means of an implicit numerical scheme, and solved using Newton-Raphson iterations. While maintaining instantaneous controllability, the algorithm is both significantly faster and more accurate than the previous filter-based solution. A real time version of the model has been implemented under the Pure-Data audio processing environment and as a VST plugin.

1. INTRODUCTION

Within the *virtual analog* research field, several efforts have been made to properly simulate the voltage-controlled filters (shortly, VCF) onboard the monophonic synthesizers of the 60's, such as Robert Moog's transistor-based VCF [1, 2, 3], or the diode-based VCF designed for the Electronic Music System *Voltage Controlled for Studio with 3 Oscillators*, known as VCS3, which is considered in this paper.

The first discrete time model of the EMS VCS3 VCF was presented in 2008 [4]. In that model the analog filter network was accurately represented through a nonlinear differential equation system, that was later discretized by means of an esplicit scheme using a fourth-order Runge-Kutta method. A similar system was reproposed in 2010 [5], where a passive digital filter network directly coming out from the analog structure was computed using fixed-point iterations. This computation was proven to be efficient enough to run in real-time meanwhile allowing variation at sample rate of the VCF control parameters, typically the cutoff frequency and feedback gain.

In this paper, an evolved simulation of the previous system [5] is proposed. Specifically, the system equations are reformulated in order to avoid the expensive computation of transcendent functions, hence obtaining a quasi-polynomial system which is then discretized using an implicit scheme and Newton-Raphson iterations. Overall, the speed improvement is of an order of magnitude. Moreover, due to the improved numerical behavior, the simulation computes accurate solutions for large values of the control parameters, i.e. where the fixed-point method failed to converge in reasonable time causing noticeable artifacts in the output.

By considering a specific VCF analog circuitry, obviously this study has not the generality of recently proposed techniques for the simulation of generic electrical networks [6, 7]. However, some of the employed recipes like the removal of transcendent functions and the specific implicit scheme design can be applied to other nonlinear systems, that need to be accurately simulated in real time.

This poster is organized as follows. In Sec. 2, the nonlinear differential state-space representation of the VCS3 VCF is shortly reviewed. Then, in Sec. 3, some algebraic manipulations are carried out to obtain an equivalent description containing polynomial functions, which substitute the hyperbolic tangent used in the original formulation. The resulting system is discretized with an implicit method that is proposed in Sec. 4, and whose implementation details are discussed. Finally, Sec. 5 shows some results that prove the improved performance of the proposed solution compared to the previous model.

2. MODEL

The VCF is a parametric filter, whose cutoff frequency and resonant behavior can be controlled respectively by varying the characteristic value of the nonlinear resistive components and the feedback gain. The behavior of the original circuitry can be described with a good approximation by the following differential equations system [5]:

$$\begin{cases} \dot{v}_{C_1} = \dfrac{I_0}{2C}\left(\tanh\dfrac{v_{IN}-v_{OUT}}{2V_T}+\tanh\dfrac{v_{C_2}-v_{C_1}}{2\gamma}\right) \\ \dot{v}_{C_2} = \dfrac{I_0}{2C}\left(\tanh\dfrac{v_{C_3}-v_{C_2}}{2\gamma}-\tanh\dfrac{v_{C_2}-v_{C_1}}{2\gamma}\right) \\ \dot{v}_{C_3} = \dfrac{I_0}{2C}\left(\tanh\dfrac{v_{C_4}-v_{C_3}}{2\gamma}-\tanh\dfrac{v_{C_3}-v_{C_2}}{2\gamma}\right) \\ \dot{v}_{C_4} = \dfrac{I_0}{2C}\left(-\tanh\dfrac{v_{C_4}}{6\gamma}-\tanh\dfrac{v_{C_4}-v_{C_3}}{2\gamma}\right) \\ v_{OUT} = (K+1/2)\,v_{C_4} \end{cases} \quad (1)$$

In this standard space-state representation, v_{IN} and v_{OUT} are respectively the voltage input and output signals; K is the feedback gain (ranging between 0 and 10 in the VCS3 synthesizer) and I_0 is a bias current setting the resistance values, hence the cutoff frequency of the filter. The state variable vector $\mathbf{v}_C = [v_{C_1}, \ldots, v_{C_4}]$ corresponds to the voltages through the four capacitors present in the electrical network. The other terms in Eq. (1) are the constants $\eta = 1.836$, $V_T = 26$ mV, $\gamma = \eta V_T = 48$ mV, and $C = 0.1$ μF. The system has a fixed point at the origin, corresponding to null charge at the capacitors [5].

3. NONLINEAR SYSTEM REFORMULATION

The main complexity in the model expressed by Eq. (1) is that any accurate system solution requires the computation of several hyperbolic tangents. This computation, especially on modern hard-

ware, can be several order of magnitudes more expensive than multiplying. Thus, it would be beneficial to rewrite equations containing only polynomial functions.

In order to do so[1], we exploit the self-similarity of the derivative of the hyperbolic tangent: $d/dt \tanh(t) = 1 - \tanh^2(t)$. Then, we proceed by assigning the values of the nonlinear terms in (1) to the new auxiliary state vector $\mathbf{x} = [x_1, \ldots, x_5]$:

$$\begin{cases} x_1 = \tanh \dfrac{v_{C_2} - v_{C_1}}{2\gamma} \\ x_2 = \tanh \dfrac{v_{C_3} - v_{C_2}}{2\gamma} \\ x_3 = \tanh \dfrac{v_{C_4} - v_{C_3}}{2\gamma} \\ x_4 = \tanh \dfrac{-(K+1/2)v_{C_4}}{2V_T} \\ x_5 = \tanh \dfrac{v_{C_4}}{6\gamma} \end{cases} . \tag{2}$$

The only auxiliary variable which does not capture a portion of (1) directly is x_4. Even if it is possible to set this variable to one argument of the hyperbolic tangent, doing so would require to include (hence to numerically compute) the derivative of the input signal v_{IN}. Since assumptions on the smoothness of the input signal cannot be made, it is preferable not to include incoming signals' derivatives into the system, as they are sensitive to noise and prone to amplification of the high frequencies.

Rather, we use the addition formula for the hyperbolic tangent and rewrite the term as

$$\tanh \frac{v_{IN} - v_{OUT}}{2V_T} = \frac{\tilde{v} + x_4}{1 + \tilde{v}x_4},$$

where $\tilde{v} = \tanh(v_{IN}/2V_T)$. In this way we are still left with one transcendent function in the system: since it includes only the signal v_{IN}, its (possibly parallel) computation across the input buffer can be decoupled by the solution of the system.

The system nonlinearities cause a bandwidth expansion on the signal, so that oversampling is necessary to avoid aliasing [5]. A good compromise between aliasing reduction and computational cost is the use of 8x upsampling, as shown in Fig. 1.

Taking the time-derivative of the new state vector $\mathbf{x}$, we obtain the following equations:

$$\begin{cases} \dot{x}_1 = \dfrac{\dot{v}_{C_2} - \dot{v}_{C_1}}{2\gamma}(1 - x_1^2) \\ \dot{x}_2 = \dfrac{\dot{v}_{C_3} - \dot{v}_{C_2}}{2\gamma}(1 - x_2^2) \\ \dot{x}_3 = \dfrac{\dot{v}_{C_4} - \dot{v}_{C_3}}{2\gamma}(1 - x_3^2) \\ \dot{x}_4 = \dfrac{(K+1/2)\dot{v}_{C_4}}{2V_T}(1 - x_4^2) \\ \dot{x}_5 = \dfrac{\dot{v}_{C_4}}{6\gamma}(1 - x_5^2) \end{cases} . \tag{3}$$

Finally, substituting into (3) the expressions for the previous state

[1]This transformation is generally applicable whenever the nonlinearities are compositions in their own of exponential functions.

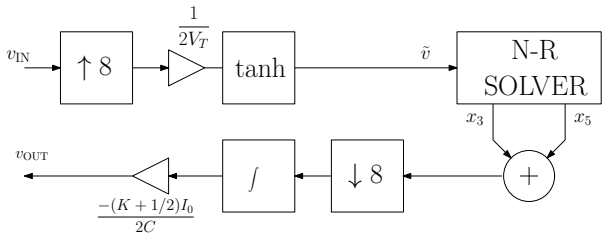

Figure 1: Block diagram illustrating the computational stages required before and after the nonlinear system solver: upsampling, nonlinear map, system solving, downsampling and integration.

$\mathbf{v}_C$ yields the following quasi-polynomial nonlinear ODE system:

$$\begin{cases} \dot{x}_1 = \dfrac{I_0}{4C\gamma}\left(x_2 - \dfrac{\tilde{v} - x_4}{1 - \tilde{v}x_4}\right)(1 - x_1^2) \\ \dot{x}_2 = \dfrac{I_0}{4C\gamma}(x_3 - 2x_2 + x_1)(1 - x_2^2) \\ \dot{x}_3 = \dfrac{I_0}{4C\gamma}(-x_5 - 2x_3 - x_2)(1 - x_3^2) \\ \dot{x}_4 = \dfrac{I_0(K+1/2)}{4CV_T}(-x_5 - x_3)(1 - x_4^2) \\ \dot{x}_5 = \dfrac{I_0}{12C\gamma}(-x_5 - x_3)(1 - x_5^2) \end{cases} . \tag{4}$$

In this system all functions are polynomial, except for a divide in the first equation due to the need to avoid the derivative of the input signal.

We can recover v_{OUT} from the relation $\dot{v}_{C_4} = (-I_0/2C)(x_5 + x_3)$. Since in (1) v_{OUT} is proportional to v_{C_4}, numerical integration (see Fig. 1) is required as a final inexpensive step to compute the solution, furthermore preserving the *passivity* of the continuous integrator if the trapezoidal rule is used for the discretization [5], as we will do through Eq.(6).

4. NUMERICAL SOLUTION

We discretize (4) using standard techniques from numerical analysis [8]. The system can be written in vectorial form as

$$\dot{\mathbf{x}} = f(\tilde{v}, \mathbf{x}), \tag{5}$$

where f is a nonlinear vectorial function. Time discretization is performed with the Adams-Moulton 1-step method (i.e., the trapezoidal rule):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{T}{2}\left[f(\tilde{v}_n, \mathbf{x}_n) + f(\tilde{v}_{n+1}, \mathbf{x}_{n+1})\right] \tag{6}$$

which is numerically equivalent to the bilinear transformation [5]. The resulting numerical equation is implicit, since at each step we need to solve a nonlinear system of equations. More precisely, we have to find the zeroes of the vectorial function

$$\begin{aligned} F(\xi) &= \mathbf{x}_n + \frac{T}{2}\left[f(\tilde{v}, \mathbf{x}_n) + f(\tilde{v}, \xi)\right] - \xi \\ &= \mathbf{F}_{0,n} + \frac{T}{2} f(\tilde{v}, \xi) - \xi. \end{aligned} \tag{7}$$

The term $\mathbf{F}_{0,n}$ has been highlighted in the expression so that at each time step it can be updated efficiently with the recursive relation

$$\mathbf{F}_{0,n+1} = 2\mathbf{x}_{n+1} - \mathbf{F}_{0,n}. \tag{8}$$

The system (7) is solved using Newton-Raphson iterations, which guarantee second-order convergence in our case. At every time step we start with the initial guess $\xi_0 = \mathbf{x}_n$, then we update iteratively the linear solution

$$\begin{aligned} JF(\xi_i)\ \ \delta_{\xi_i} &= -F(\xi_i) \\ \xi_{i+1} &= \xi_i + \delta_{\xi_i}, \end{aligned} \tag{9}$$

where $JF(\xi_i)$ is the Jacobian of F at the i-th iteration, furthermore related to the Jacobian of f by the relation $JF = \frac{T}{2} Jf - \mathbb{I}$.

Convergence is checked against the L_∞ norm of the residual vector δ_{ξ_i} using a threshold of 10^{-8}, accurate enough for single-precision floating point computations. In [5] a different condition was employed, based only on the output value. Conversely, checking all the values of the state vector can provide more accurate results especially during the simulation of transients.

The computation of the term F and the Jacobian JF can be simplified if we split the terms of the system (4) into a vector of coefficients

$$\mathbf{c} = \begin{bmatrix} I_0/(4C\gamma) \\ I_0/(4C\gamma) \\ I_0/(4C\gamma) \\ I_0(K+1/2)/(4CV_T) \\ I_0/(12C\gamma) \end{bmatrix},$$

plus two vectors, respectively containing the differences and the quadratic terms in (4):

$$\mathbf{t} = \begin{bmatrix} x_2 - \frac{\tilde{v}-x_4}{1-\tilde{v}x_4} \\ x_3 - 2x_2 + x_1 \\ -x_5 - 2x_3 - x_2 \\ -x_5 - x_3 \\ -x_5 - x_3 \end{bmatrix},\ \mathbf{d} = \begin{bmatrix} 1 - x_1^2 \\ 1 - x_2^2 \\ 1 - x_3^2 \\ 1 - x_4^2 \\ 1 - x_5^2 \end{bmatrix}.$$

For sake of compactness, we have dropped the discrete-time index n in the previous equations. By (9), F can be computed from (7) using the obvious relation

$$f(\tilde{v}, \mathbf{x}) = \mathbf{c} * \mathbf{t} * \mathbf{d}, \tag{10}$$

while the Jacobian of f is written in terms of the new vectors as

$$Jf = -\begin{bmatrix} 2c_1(d_1+t_1x_1) & -c_1d_1 & 0 & c_1d_1f_\mathrm{v} & 0 \\ -c_2d_2 & 2c_2(d_2+t_2x_2) & -c_2d_2 & 0 & 0 \\ 0 & -c_3d_3 & 2c_3(d_3+t_3x_3) & 0 & c_3d_3 \\ 0 & 0 & c_4d_4 & 2c_4t_4x_4 & c_4d_4 \\ 0 & 0 & c_5d_5 & 0 & c_5(d_5+2t_5x_5) \end{bmatrix} \tag{11}$$

where $f_\mathrm{v} = (\tilde{v}^2 - 1)/(1 - \tilde{v}x_4)^2$.

Note that the linear system described by this Jacobian matrix does not have any particular structure. For this reason, we have to employ general linear solvers such as LU decomposition with pivoting or QR factorization. In our tests, LU with pivoting was slightly less accurate, occasionally requiring an extra iteration to converge, but generally 25% faster than QR decomposition.

5. DISCUSSION

The model has been implemented both as an offline Matlab simulation, as a C++ *external* running under the PureData [9] real-time audio processing environment and as a VST [10] Plugin. For the real-time versions, we have employed the library *libresample* [11] for accurate upsampling and downsampling, and the *Eigen* processing library [12] for linear system solving and parallelized vector computations. The real-time model requires less than 10% CPU power on an Intel Core2 Duo@2.4Ghz laptop, independently of the VCF parameter values.

The main advantage of the new algorithm resides in significantly faster, and parameter-independent computation times compared to the previous reference model [5]. We can give a rough quantitative comparison considering the approximate number of MPOS (multiplication per input sample) required by either implementation. At every step the previous algorithm requires the computation of 5 hyperbolic tangents, plus 4 discrete-time integrations and 10 multiply-and-accumulate (MAC) operations. If we approximate the cost of each hyperbolic tangent to 50 MPOS[2], then the cost for each fixed-point iteration amounts to about 280 MPOS. Since the average number of iterations is between 10 to 50, we can estimate the total cost per (upsampled) time step as ranging between 3000 and 15000 MPOS. In [5] some examples of the number of iterations required for different control values are given.

As opposed to the previous procedure, the proposed algorithm instead requires to compute just one hyperbolic tangent. Since this function can be precomputed on the input buffer at 1/4 the sampling frequency, in parallel to the system integration, the cost of this computation is around 5 MPOS. Building the right-hand term F as by (10) requires roughly 20 MPOS, including the divide, and the same cost is required for the computation of the Jacobian matrix (11). LU pivoting takes about 120 MPOS, resulting in a total cost of 165 MPOS per iteration. Finally, thanks to the better numerical behaviour of the Newton-Raphson method, only 3-6 iterations are usually required for convergence. Therefore, the total cost of the proposed algorithm is between 500 and 1000 MPOS, about one order of magnitude less than the fixed-point model.

As a by-product, the new model is more accurate for high values of the control parameters, especially the cutoff frequency. In fact, iteration upper bounds are inevitably reached by the fixed-point solver under these conditions [5]. The effect is illustrated in Fig. 2, where the cutoff frequency is linearly increased across the simulation. It can be easily noticed that the previous solver (left plots) behaves inappropriately around cutoff values amounting to 10 KHz, when the number of iterations starts to reach the upper bound set to 100 iterations. Within ranges of the control parameters that do not cause iteration explosion, both models introduce a relative error below -80 dB at every step, in ways that their output differences are inaudible.

6. CONCLUSION

We have transformed the equations proposed for the simulation of the VCS3 VCF, by deriving a quasi-polynomial system allowing an implicit integration based on Newton-Raphson iterations. Compared to the previous one, the new solution is significantly faster as well as more accurate when high values of control parameters are used, meanwhile enabling full controllability of the parameters at sample rate without artifacts.

7. ACKNOWLEDGMENTS

The authors would like to thank Marco Civolani for his inspiring comments on this work.

[2] Benchmarked on an Intel Core2Duo@2.4Ghz with gcc 4.6.

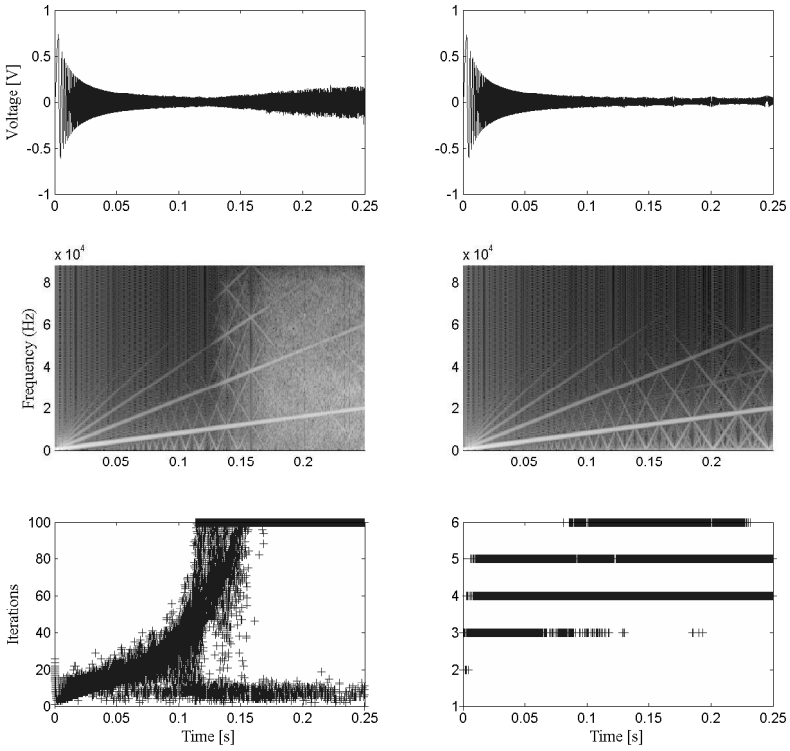

Figure 2: Comparison of the proposed model (right) versus the previous implementation (left). The input signal is a sinusoidal sweep ranging from 0 to 20 KHz, while the cutoff frequency is varied between 10 and 20KHz. The feedback gain is kept constant ($K = 1$) during the simulation. Sampling frequency set at 176400 Hz.

8. REFERENCES

[1] T. Stilson and J.O. Smith, "Analyzing the Moog VCF with considerations for digital implementation," in *Proceedings of the International Computer Music Conference*, 1996, pp. 398–401.

[2] A. Huovilainen, "Nonlinear digital implementation of the Moog ladder filter," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004, pp. 61–64.

[3] F. Fontana, "Preserving the structure of the Moog VCF in the digital domain," in *Proc. Int. Comput. Music Conf*, Copenhagen, Denmark, 2007, pp. 27–31.

[4] M. Civolani and F. Fontana, "A nonlinear digital model of the EMS VCS3 voltage-controlled filter," in *Proceedings of the 11th International Conference on Digital Audio Effects*, Helsinki, Finland, 2008, pp. 35–42.

[5] F. Fontana and M. Civolani, "Modeling of the EMS VCS3 voltage-controlled filter as a nonlinear filter network," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 4, pp. 760–772, 2010.

[6] D.T. Yeh, J.S. Abel, and J.O. Smith, "Automated Physical Modeling of Nonlinear Audio Circuits for Real-Time Audio Effects-Part I: Theoretical Development," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 4, pp. 728–737, 2010.

[7] F. Fontana and F. Avanzini, "Computation of delay-free nonlinear digital filter networks: Application to chaotic circuits and intracellular signal transduction," *Signal Processing, IEEE Transactions on*, vol. 56, no. 10, pp. 4703–4715, 2008.

[8] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*, Springer Verlag, 2007.

[9] M. Puckette, "Pure Data," in *Proc. International Computer Music Conference*, Thessaloniki, Greece, 1997.

[10] Steinberg Soft and Hardware GMBH, "Steinberg virtual studio technology (vst) plug-in specification 2.0 software development kit," 1999.

[11] D. Mazzoni, "libresample," http://ftp.debian.org/pool/main/libr/libresample/.

[12] Gael Guennebaud, Benoit Jacob, et al., "Eigen v3," http://eigen.tuxfamily.org, 2010.