Musical Applications of New Filter Extensions to Max/MSP

Tristan Jehan, Adrian Freed, Richard Dudas

CNMAT, UC Berkeley, 1750 Arch Street, Berkeley, CA 94709, (510) 643 9990
{tristan,adrian,dudas}@cnmat.berkeley.edu

Abstract

We introduce three new Max/MSP signal processing extensions that efficiently implement IIR filters: biquadbank~, peqbank~ and resonators~. After describing their common properties, specific features and new support objects, we conclude the paper with a summary of their musical applications.

1. Introduction

We have developed three new filter extensions for the Max/MSP environment (Zicarelli, 1998), biquadbank~, peqbank~ and resonators~ and some ancilliary extensions for control-parameter interpolation, transformation and graphical display of phase and frequency response functions.

2. General Features

This section will describe features common to all of our filter objects; subsequent sections will describe each object individually.

Standard Max/MSP filter objects such as biquad~ and lores~ implement variants of a single second-order IIR filter. Our new objects implement multiple filters, the number of which can be changed in real-time. For biquadbank~ and peqbank~ the filters are cascaded in series: the output of one feeds the input of the next. Resonators~ implements a parallel bank of filters with a common input and summed output. Building multiple filters in a single object allows more efficient use of processor and memory resources than is possible patching together individual Max objects. The objects obtain their efficiency by exposing multiple opportunities for parallelism to the C compiler (Dowd and Loukides, 1993, Freed, 1993, Freed and Chaudhary, 1998). On PowerPC G3 processors this results in good cache utilization and efficient pipelining of the floating-point multiply and add units. Exposing such parallelism is essential to achieve the potential of new processor features such as the vector arithmetic of the PowerPC G4 or the very long instruction word (VLIW) of the Intel IA-64.

Other important advantages of implementing multiple filters in a single object include simplified patching, support for concurrent update of the control parameters and the ability to change the number of filters in real-time.

The standard Max/MSP filter objects do not implement smooth control parameter changes, resulting in audible clicks and *zippering*, which are unacceptable audible artifacts in time-varying filter applications (Ding and Rossum, 1995). Our new objects offer a smooth mode and a fast mode. The former is for time-varying applications and the latter, approximately 20% faster, is for fixed parameter situations.

Filter properties are controlled by sending a list of parameter values. The interpretation of the list data is determined by a message name preceding floating-point parameters. This allows the objects to interpret many variant ways of describing filters. A further advantage of binding all the control parameters into a single list is that the objects can update the filters simultaneously.

After each filter parameter update, the objects output a list containing the filter coefficients in a standard form for display or debugging. One of the authors has developed a new Max user interface object, filtergraph~, shown in Figure 1 graphing the frequency response of a cascade of biquads. Filtergraph~ will be available as part of a future MSP release.

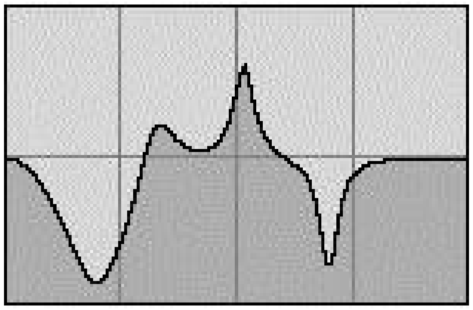
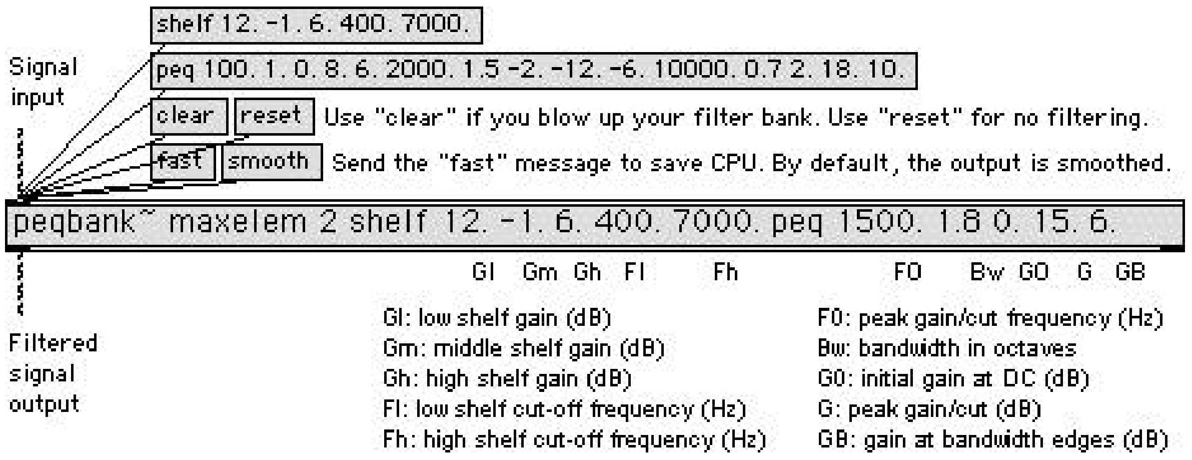


Figure 1: Frequency response of a cascade of biquad filters

3. Biquadbank~

This is the multi-filter, cascaded generalization of the standard biquad~ object, with the optional smooth mode described above. The object is useful when direct control of the filter coefficients is required.


shelf 12. -1. 6. 400. 7000.
peq 100. 1. 0. 8. 6. 2000. 1.5 -2. -12. -6. 10000. 0.7 2. 18. 10.
clear
reset
fast
smooth
Signal input
peqbank~ maxelem 2 shelf 12. -1. 6. 400. 7000. peq 1500. 1.8 0. 15. 6.
Gl Gm Gh Fl Fh F0 Bw G0 G GB
Filtered signal output

Use "clear" if you blow up your filter bank. Use "reset" for no filtering.

Send the "fast" message to save CPU. By default, the output is smoothed.

Gl: low shelf gain (dB)	F0: peak gain/cut frequency (Hz)
Gm: middle shelf gain (dB)	Bw: bandwidth in octaves
Gh: high shelf gain (dB)	G0: initial gain at DC (dB)
Fl: low shelf cut-off frequency (Hz)	G: peak gain/cut (dB)
Fh: high shelf cut-off frequency (Hz)	GB: gain at bandwidth edges (dB)

Figure 2: Use of the peqbank~ object

4. Peqbank~

This object (shown in Figure 2) adds a real-time filter coefficient design layer to a cascaded biquad filter bank. It supports two shelving equalizers and an arbitrary number of parametric equalizers. This object implements the latest theoretical work on parametric equalizer design (Orfanidis, 1997), i.e., a transfer function that models that of an analog equalizer more accurately at the Nyquist frequency. The figure below compares the frequency response of the classical approach for a parametric peaking EQ with the analog simulation approach. The center frequency, gain, and bandwidth are the same in both cases. The analog simulation avoids unnecessary high-frequency attenuation and exhibits better symmetry around the peak frequency, as illustrated in Figure 3.

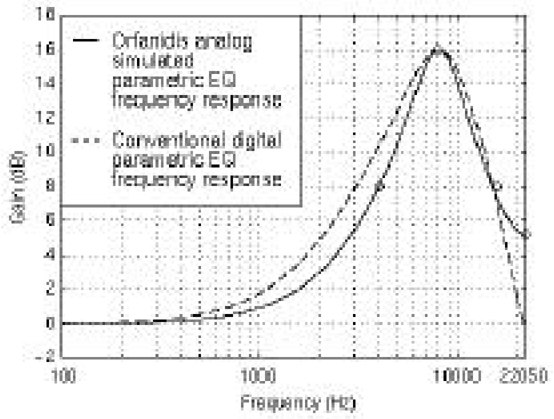


Figure 3: A comparison of the frequency response of simulated analog versus conventional digital parametric EQ implementations with the same center frequency, gain, and bandwidth.

5. Resonators~

This object implements parallel banks of two-pole resonators. Real-time coefficient design is performed to map frequency, gain, and decay rate into filter coefficients:

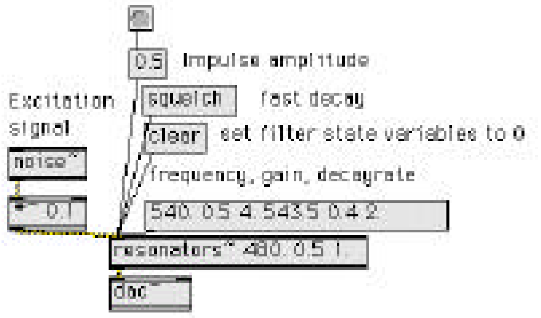


Figure 4: Use of resonators~

Impulses sent to resonators~ as floating-point messages are scaled internally to obtain the correct impulse response of the resonator bank. Signal inputs are scaled according to a more conservative criterion that lowers the gain of narrow band filters to avoid output overflow for inputs with energy at center frequencies of sharp resonators.

The res-transform object was developed to store and manipulate models of resonance (Potard, et al., 1985, Potard, et al., 1986). It accepts list messages specifying resonance model data in a variety of formats. The transformations supported include spectral envelopes, pitch transposition, bandwidth and gain scaling. The res-transform object outputs the transformed model on receipt of a bang message.

In Figure 5, the res-transform output is sent to a resonators~ object and also to an extended version of the Max LCD object. A new drawing primitive was added to display the models in a convenient 2-D form developed originally as an extension to MacMix (Freed, 1985, Freed, 1987) for use on the Reson8 digital signal multiprocessor (Barriere, et al., 1989).

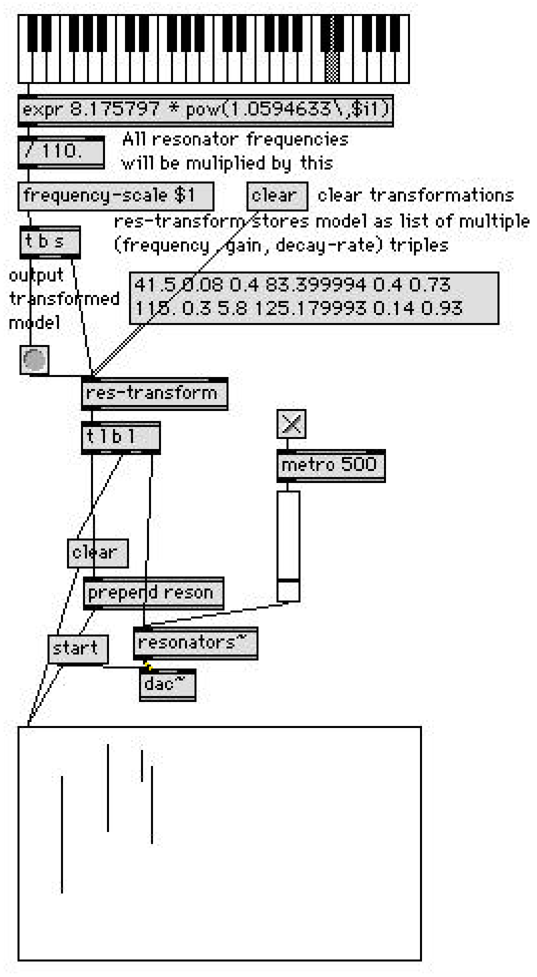


Figure 5: Use of res-transform and display of resonance data

The smoothing function of resonators~ (and the other filter objects) operates by linear interpolation in the filter coefficient space. This efficient scheme has no filter instability problems (Moorer, 1989). However, for large filter parameter changes the interpolated filters may be unsatisfactory from a musical perspective. We therefore employ a nested interpolation scheme where coefficient interpolation operates for short vectors of the order of 2ms and filter specification parameters are interpolated at slower rates determined by the Max event scheduler.

This interpolation model required us to create a new object, list-interpolate, which is useful in many different situations. This object accepts lists and outputs an interpolation of the last lists received according to an interpolation factor sent as a float message. As well as smoothing parameter changes, this object may be used for timbral interpolation.

6. Applications

Models of Resonance

Edmund Campion used resonators~ in the world premier of “Play-Back” on June 7, 1999. Jean-Baptiste Barrière used the same object in a May 1999 installation called “Autoportrait in motion” presented at the Festival International des Musiques Électroacoustiques de Bourges in France.

SDIF Synthesizers

We are using resonators~ to create a reference implementation of the SDIF standard in Max/MSP (Wright, et al., 1999).

Open Sound Edit

Open Sound Edit (Chaudhary and Freed, 1999) uses Open Sound Control (Wright and Freed, 1997) to message a resonator model synthesizer built around resonators~.

Guitar effects

We have used peqbank~ and resonators~ extensively to develop new polyphonic guitar effects based on vocal-tract modeling, frequency localized distortion and coordinated equalization.

Organ Sound Synthesis

We are using resonators~ to model the decay of low-pitched organ tones.

7. Future Work

We are extending resonators~ to support more general damped exponential models coded in SDIF (Wright, et al., 1999); of particular urgency is the incorporation of new methods to provide more user control over input scaling. We are developing new, complementary objects for FIR filtering and convolution.

We have implemented resonators~ in a new real-time music programming environment, Open Sound World (OSW) and will soon port the remaining filter objects to the OSW C++ extensions framework (Chaudhary, et al., 1999).

8. Acknowledgements

We gratefully acknowledge the support of Gibson Guitar Inc, and the O'Neill foundation.

9. References

J.-B. Barriere, P.-F. Baisnee, A. Freed, and M.-D. Baudot (1989), “A Digital Signal Multiprocessor and its Musical Application,” presented at 15th International Computer Music Conference, Ohio State University.

A. Chaudhary and A. Freed (1999), “Visualization, Editing and Spatialization of Timbral Resources using the OSE Framework,” presented at Audio Engineering Society 107th Convention, .

A. Chaudhary, A. Freed, and M. Wright (1999), “An Open Architecture for Real-Time Audio Processing Software,” presented at Audio Engineering Society 107th Convention, .

Y. Ding and D. Rossum (1995), “Filter morphing of parametric equalizers and shelving filters for audio signal processing,” *Journal of the Audio Engineering Society*, vol. 43, num. 10, pp. 821-6.

K. Dowd and M. K. Loukides (1993), *High performance computing.* Sebastopol, CA: O'Reilly & Associates.

A. Freed (1985), “MacMix: Mixing Music with a Mouse,” presented at Proceedings of the USENIX Association Second Computer Graphics Workshop, Monterey, CA, USA.

A. Freed (1987), “Recording, mixing, and signal processing on a personal computer,” presented at Proceedings of the AES 5th International Conference: Music and Digital Technology, Los Angeles.

A. Freed (1993), “Guidelines for signal processing applications in C,” *C Users Journal*, pp. 85(9).

A. Freed and A. Chaudhary (1998), “Music Programming with the new Features of Standard C++,” presented at International Computer Music Conference, Ann, Arbor, Michigan.

J. Moorer (1989), personal communication.

S. J. Orfanidis (1997), “Digital parametric equalizer design with prescribed Nyquist-frequency gain,” *Journal of the Audio Engineering Society*, vol. 45, num. 6, pp. 444-55.

Y. Potard, P.-F. Baisnée, and J.-B. Barriere (1985), “Models of Continuity between Synthesis and Processing for the Elaboration and Control of Timbre Structures,” presented at International Computer Music Conference, Vancouver.

Y. Potard, P.-F. Baisnée, and J.-B. Barriere (1986), “Experimenting with Models of Resonance Produced by a New Technique for the Analysis of Impulsive Sounds,” presented at International Computer Music Conference, La Haye.

M. Wright and A. Freed (1997), “Open Sound Control: A New Protocol for Communicating with Sound Synthesizers,” presented at International Computer Music Conference, Thessaloniki, Greece.

M. Wright, S. Khoury, R. Wang, and D. Zicarelli (1999), “Supporting the Sound Description Interchange Format in the Max/MSP Environment,” presented at International Computer Music Conference, Beijing, China.

D. Zicarelli (1998), “An Extensible Real-Time Signal Processing Environment for Max,” presented at International Computer Music Conference, Ann Arbor, Michigan.