Real-time Frequency-Domain Digital Signal Processing on the Desktop

Zack Settel & Cort Lippe

McGill University, Music Faculty
555 rue Sherbrooke Ouest
Montreal, Quebec H3A 1E3
CANADA
zack@music.mcgill.ca

University at Buffalo, Department of Music
Hiller Computer Music Studios
222 Baird Hall
Buffalo, NY, USA 14260
lippe@acsu.buffalo.edu

Abstract

For some years, real-time general-purpose digital audio systems, based around specialized hardware, have been used by composers and researchers in the field of electronic music, and by professionals in various audio-related fields. During the past decade, these systems have gradually replaced many of the audio-processing devices used in amateur and professional configurations. Today, with the significant increase in computing power available on the desktop, the audio community is witnessing an important shift away from these systems, which required specialized hardware, towards general purpose desktop computing systems featuring high-level digital signal processing (DSP) software programming environments.

Introduction

An new real-time DSP programming environment called Max Signal Processing (MSP) [Zicarelli, 1997], was released this past year for the Apple Macintosh PowerPC platform. This software offers a suite of signal processing objects as an extension to the widely used MAX software environment, and provides new opportunities for musicians and engineers wishing to explore professional-quality real-time DSP. Most important, MSP provides a number of frequency-domain processing primitives that allow for the development of sophisticated frequency-domain signal processing applications.

Working in MSP, the authors have developed a library of frequency-domain DSP applications for cross-synthesis, analysis/resynthesis, denoising, pitch suppression, dynamics processing, advanced filtering, spectrum-based spatialization, and phase vocoding. Much of this library was originally developed by the authors on the IRCAM Signal Processing Workstation (ISPW) [Lindemann, 1991], and has been discussed in previous papers [Settel & Lippe, 1994]. MSP is a direct descendant of ISPW Max [Puckette, 1991], but provides greater portability and increased functionality. The authors have made improvements to the library, while developments in new directions have been made possible by features of MSP which ameliorate exploration in the frequency domain. Techniques and applications will be presented and discussed in terms of both general algorithm and MSP implementation, providing a concrete point of departure for further exploration using the MSP environment.

1. Frequency-domain signal processing operations and techniques

1.1 Fundamental operations

The standard operations which are used when processing audio signals in the frequency domain typically include: (1) windowing of the time-domain input signal, (2) transformation of the input signal into a frequency domain signal (spectrum) using the Fast Fourier Transform (FFT), (3) various frequency-domain operations such as complex multiplication for convolution, (4) transformation of the frequency-domain signals back into the time domain using the Inverse Fast Fourier Transform(IFFT), (5) and windowing of the time-domain output signal. This section of the paper will discuss some of the basic operations and techniques used in the various applications developed by the authors.

Implementation

The FFT object stores time-domain signals as buffers of samples upon which the FFT analysis is done. For the purpose of discussion, the examples given in this paper make use of buffers of 1024 samples. Unlike time-domain signals, a frequency-domain signal is represented by a succession of spectral "frames". Like frames in a movie, the frames of FFT data represent a "snapshot" of a brief segment of an audio signal. A frame consists of a certain number of equally spaced frequency bands called "bins". The number of bins is equal to the size of the FFT buffer, thus the frames of FFT data have 1024 bins. Each bin describes the energy in a specific part of the audio signal's frequency range.

The FFT object in MSP, based on Miller Puckette's ISPW implementation, outputs each frame, bin-by-bin, using three sample streams running at the sampling rate. Thus, each bin is represented by three samples consisting of "real" and "imaginary" values, and the bin number (index). At any given instant, each of the FFT's three signal outlets, shown below, produce a sample describing the *nth* bin of the current FFT frame. The IFFT is the complement of the FFT and expects, as input, real and imaginary values in the same format as FFT output.

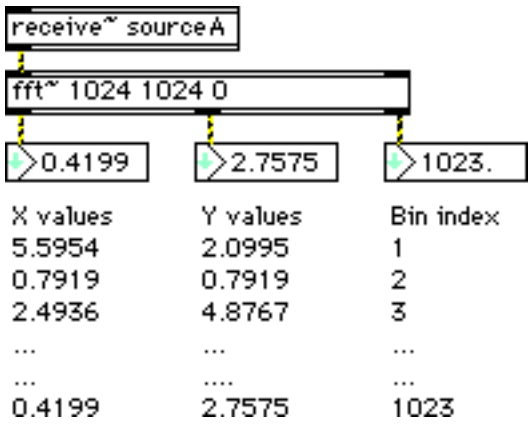

figure 1: sample-by-sample output of the FFT object

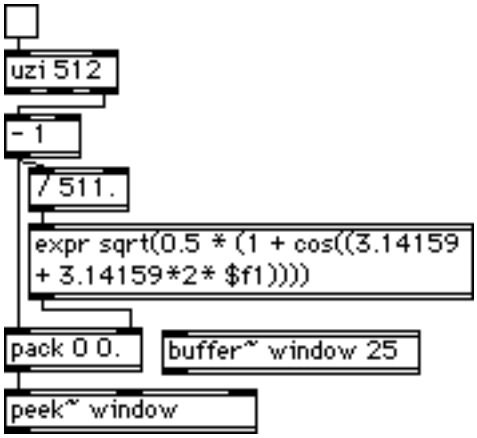

figure 2: windowing function generator

As seen in the figure above, the index values provides a synchronization signal, making it possible to identify bins within a frame, and recognize frame boundaries. The index values can be used to access bin-specific data for various operations, such as attenuation or spatialization, and to read lookup tables for windowing.

Windowing

It is necessary when modifying spectral data to apply an envelope (window) to the time-domain input/output of an FFT/IFFT pair, and to overlap multiple frames [Rabiner & Gold, 1975]. For simplicity's sake, the windowing operation shown below corresponds to a two-overlap implementation (two overlapping FFT/IFFT pairs); it is easily expanded for use in a four or eight-overlap implementation. Because of the flexibility of MAX and MSP, arbitrary windowing functions can be conveniently generated (see figure 2). In figure 3, note the use of the FFT frame index to read the lookup table-based windowing function in synchronization with the frame. The frame index is scaled between 0 and 1 in order to read the windowing function stored in an oscillator.

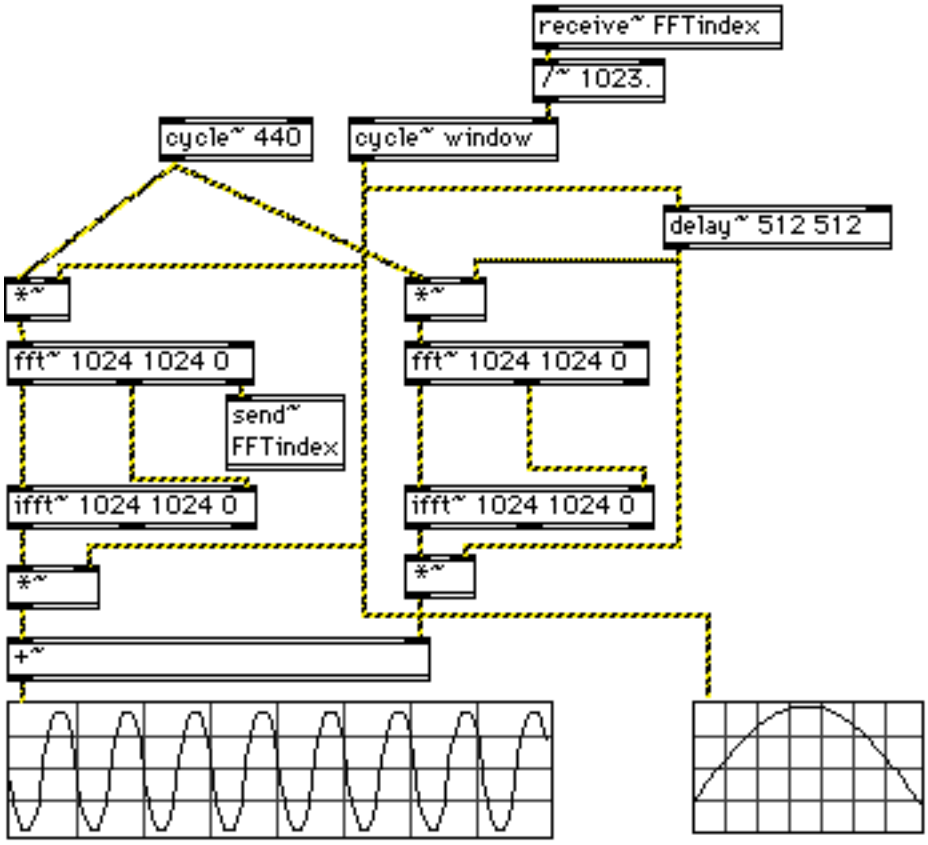

figure 3: typical two-overlap windowing of the input and output signals

1.2 Techniques

Bin-specific table lookup

Based on the implementation of MSP's FFT/IFFT objects, the authors have developed techniques for performing various operations on frequency-domain signals. The ability to access specific spectral components (bins) is central to performing frequency domain operations. The following figure illustrates how this can be accomplished using lookup tables. A 1024-point FFT, using a lookup table, offers control of 512 equalization bands across the frequency spectrum. Note that the FFT's frame index signal is used to read the lookup table without rescaling the index values.

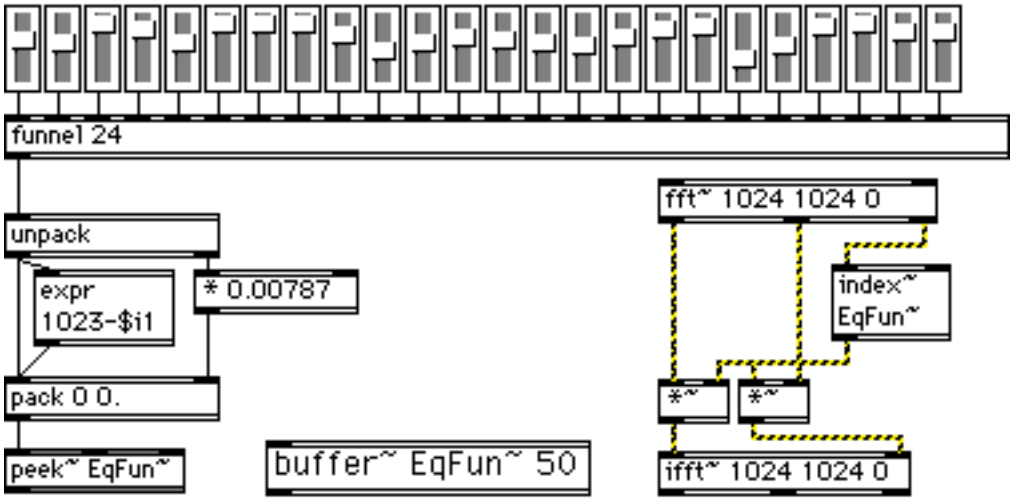

figure 4: using a lookup table to obtain bin-specific attenuation values in a graphic equalizer implementation

Flavors of convolution

Multiplying one frequency-domain signal by another (convolution) involves the operation of complex multiplication. This operation is at the heart of the many frequency-domain processing techniques developed by the authors, and provides the basis for cross-synthesis, filtering, spatialization, phase vocoding, denoising and other applications. A technique often used with convolution is the reduction of a signal's spectrum to its amplitude and/or phase information. Four examples of convolution are shown below; each one corresponds to a particular type of signal processing application.

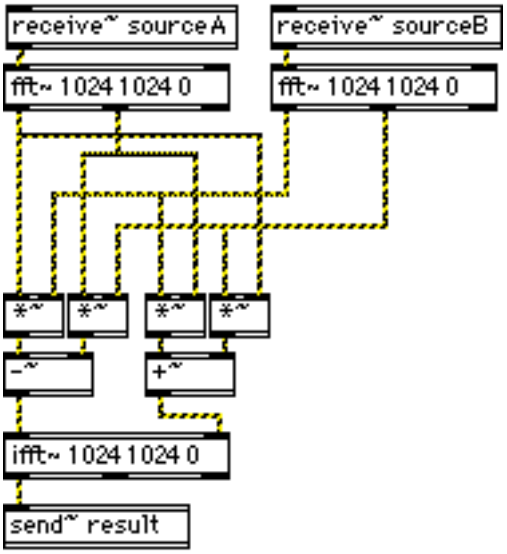

figure 5: simple convolution retaining the phases and amplitudes of each input source

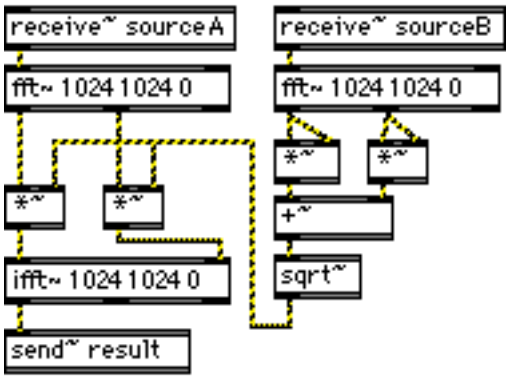

figure 6: convolution of phase/amplitude and amplitude-only spectra

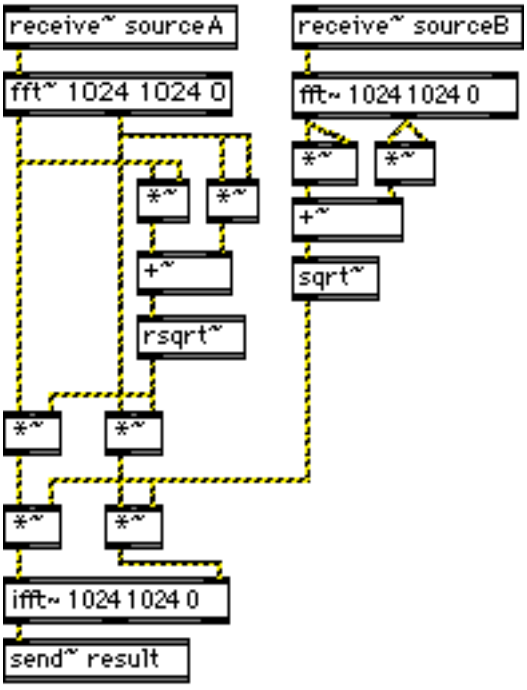

figure 7: convolution of phase-only and amplitude-only spectra

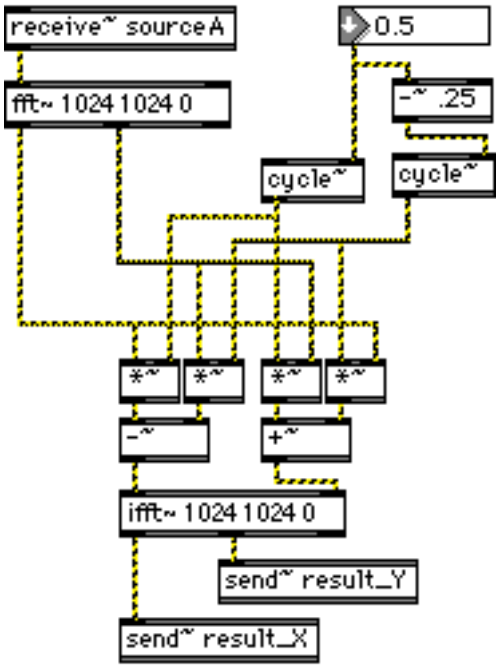

figure 8: phase rotation

2. Improvements to cross-synthesis: increasing spectral intersection

Cross-synthesis is based on convolution (the multiplication of one spectrum by another). As anyone who has worked with cross-synthesis knows, the choice of the two input signals is critical to the outcome of the operation. Input signals with little common spectra (spectral intersection) will produce poor results. In the worst case, input signals with no common spectra will produce silence. By using filtering and dynamics processing (compressor/limiter) techniques to redistribute the energy in a given spectrum, it is possible to modify the degree of spectral intersection when convolving two frequency-domain signals with dissimilar spectra. Two approaches are presented below.

2.1 Amplitude spectrum smoothing

An amplitude spectrum with deep peaks and notches (the case for a pitched sound) makes a good candidate for spectral smoothing. With MSP's FFT implementation, it is quite easy to apply a second-order low-pass filter to an amplitude spectrum. The filter's cutoff frequency, and damping parameters control the degree of spectral smoothing, averaging the energy across empty bins, thereby reducing the sharp notches or peaks which are harmonically distributed across the spectrum of pitched signals. The smoothed spectrum shown below will combine via multiplication much more effectively with the spectrum of another sound—particularly when the other sound also has pronounced peaks and notches distributed differently across its spectrum. For example, this technique can prove useful in crossing the amplitude spectrum of a singer with the spectrum of a pitched instrument, such as a saxophone.

figure 9: *top: notched amplitude spectrum of Sound A*
middle: Sound A's amplitude spectrum smoothed to reduce notches (zeros)
bottom: notched spectrum of Sound B

Note that the degree of spectral intersection between sounds A and B is much greater after smoothing has been applied to sound A's amplitude spectrum.

2.2 Bin-independent dynamics processing

The compressor/expander

Boosting weaker components of an amplitude spectrum is another technique which increases the potential degree of spectral intersection in cross-synthesis. The authors have implemented a compressor/expander which operates independently on each frequency bin of a spectrum [Settel & Lippe, 1995]. As shown below, each bin's amplitude is used as an index into a lookup table containing a compression/expansion function; the value read from the table is then used to alter (scale) the particular bin's amplitude. The degree of dynamics processing is controlled by biasing or scaling the lookup table index.

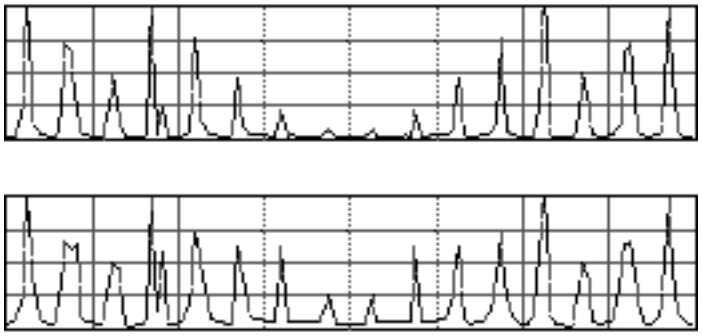

figure 10: dynamics processing to boost weaker spectral components

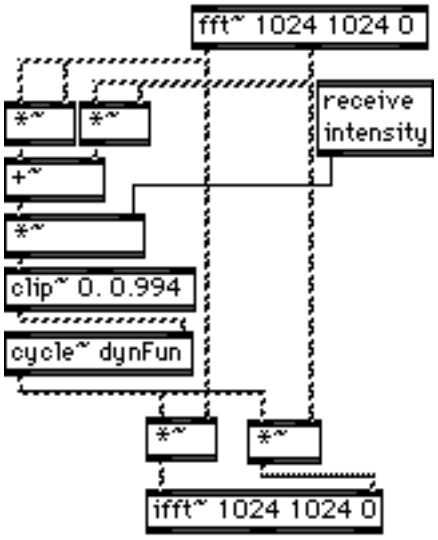

figure 11: MSP dynamics processor implementation

Note: compression/expansion functions may also be used to attenuate stronger amplitude components. This can be an effective check against extreme amplitude levels which result when crossing two sounds with similar spectral energy distributions.

Using a constant-amplitude phase-only spectrum

Finally, it is possible to maximize a spectrum's potential for intersection with another spectrum by forcing the amplitude information in each bin of the spectrum to a constant value of one (unity). Phase information remains unmodified. The resulting constant-amplitude phase-only spectrum will combine with the spectrum of another sound wherever there is energy in the other sound's spectrum. Figure 7, shown earlier, illustrates a technique that maximizes the potential for spectral intersection: a phase-only constant-amplitude spectrum is crossed with an amplitude spectrum

In any spectrum, the phase of "empty" (near 0 amplitude) bins is undefined. Consider two spectra with dissimilar spectral energy distributions. When forcing the amplitude of one sound's empty bins to unity and performing cross-synthesis with the amplitude spectrum of another sound, the resulting spectrum will tend to contain components whose phase is undefined (random). Thus, the resulting sound will be stripped of any pitch or stable frequency information.

Controlling the degree of spectral intersection

By combining the dynamics processing technique with the constant-amplitude forcing techniques described above, the degree of amplitude forcing towards unity can be continuously controlled. Thus, it is possible to specify how much of a given spectrum's original amplitude information, if any, will be used in a cross-synthesis operation with another spectrum.

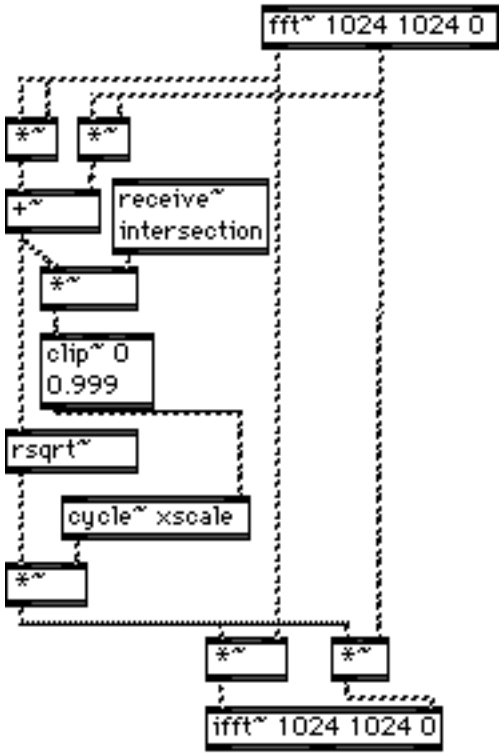

figure 12: the amplitude spectrum "intersection" parameter

3. Audio-rate control of FFT-based processing

The Max/MSP environment has two run-time schedulers: the Max "control" scheduler, which is timed on the basis of milliseconds, and the MSP "signal" scheduler, which is timed at the audio sampling rate [Puckette, 1991]. In FFT-based processing applications, where changes to the resulting spectrum are infrequent, MSP's control objects may be used to provide control parameters for the processing. This is both precise and economical. In the FFT-based equalization application shown in figure 4, a lookup table is used to describe a spectral envelope that is updated at the control rate.

However, updating lookup tables at the control rate has band-width limitations. The rapidity with which a lookup table can be altered is limited, giving the filtering certain static characteristics. Using 512 sliders to control individual FFT bins, drawing a filter shape for a lookup table with the mouse, or changing the lookup table data algorithmically provides only limited time-varying control of the filter shape. In addition, the amount of control data represented in a lookup table is large and cumbersome. Significant and continuous modification of a spectrum, as in the case of a sweeping band-pass filter, is not possible using MSP's control objects, since they can not keep up with the task of providing 1024 parameter changes at the FFT frame rate of 43 times a second (at the audio sampling rate of 44,100 samples per second).

Keeping in mind that the FFT data being filtered is signal data, a more dynamic approach to filtering is to update lookup tables at the signal rate (the audio sampling rate). Using simple oscillators for table lookup, well-known wavefrom generation and synthesis techniques can be used to provide dynamic control of filtering. FM, AM, waveshaping, phase modulation, pulse-width modulation, mixing, clipping, etc., all have the potential to provide complex, evolving waveforms which can be used as spectral envelopes to provide a high level of flexibility, plasticity, and detail to filtering. The use of operations such as stretching, shifting, duplication (wrapping), inversion, retrograde and nonlinear distortion (waveshaping) also provide comprehensive means for modifying these spectral envelopes (waveforms). Most important, the parameters of these waveform-based techniques are few, familiar and easy to control. Additionally, lookup tables can be read in non-linear fashion; control is not restricted to the linear frequency scale, thus making possible the implementation of a constant-Q bandpass filter.

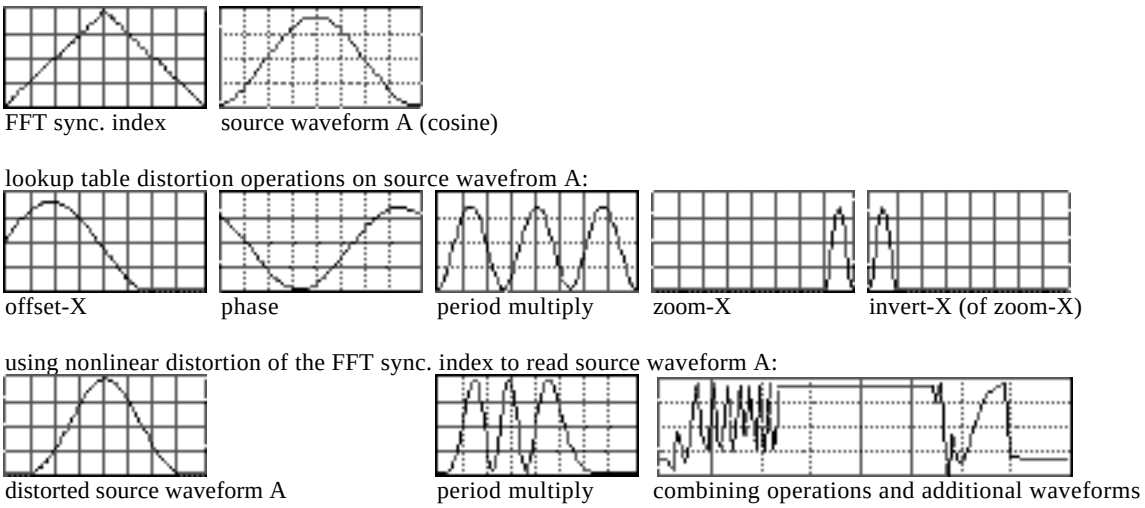

figure 13: generating spectral envelopes via simple operations on waveforms

While the discussion in this section has been limited to filtering applications, audio-rate control of FFT-based processing applies equally well to all FFT-based applications where a high degree of processing control is required at the frame rate. The authors have implemented the above mentioned techniques in the following applications: spatialization (bin-by-bin), denoising, dynamics processing, and cross-synthesis. We believe that these techniques hold great promise for control-intensive FFT-based applications.

Conclusion

The introduction of MSP offers new possibilities for musicians and researchers to develop real-timefrequency-domain digital signal processing applications on the desktop. In addition, the use of simple waveform-based techniques for low-dimensional audio-rate control of FFT-based processing has great potential: the parameters are few, familiar and easy to control, and direct mappings of real-time audio input from musicians to the control of FFT-based DSP is made possible.

Acknowledgments

The authors would like to thank Miller Puckette, David Zicarelli and the Convolution Brothers for making this work possible.

References

Lindemann, E., Starkier, M., and Dechelle, F. 1991. "The Architecture of the IRCAM Music Workstation." Computer Music Journal 15(3), pp. 41-49.

Puckette, M., 1991. "Combining Event and Signal Processing in the Max Graphical Programming Environment." Computer Music Journal 15(3), pp. 68-77.

Puckette, M., 1991. "FTS: A Real-time Monitor for Multiprocessor Music Synthesis." Music Conference. San Francisco: Computer Music Association, pp. 420-429.

Rabiner and Gold, 1975, "Theory and Application of Digital Signal Processing ", Prentice-Hall

Settel, Z., and Lippe, C., 1994. "Real-time Timbral Transformation: FFT-based Resynthesis", Contemporary Music Review Vol. 10, United Kingdom, Harwood Academic Publishers, pp. 171-179.

Settel, Z., and Lippe, C., 1995. "Real-time Musical Applications using Frequency-domain Signal Processing" IEEE ASSP Workshop Proceedings, Mohonk New York.

Zicarelli D., 1997. "Cycling74 Website", www.cycling74.com.