NES-SPECTRALE: A SUITE OF MAX TOOLS FOR PROCESSING IN THE FREQUENCY DOMAIN

Associate Professor David Hirst
Melbourne Conservatorium of Music
University of Melbourne

ABSTRACT

This project’s aim was to explore the notion of “timbre processing” using Jitter matrices as the store and processing mechanism. The NES-Spectrale suite of Max/Jitter patchers is a structured set of tools for processing sounds in the frequency domain, and inverse transforming the results into the time domain, in the form of audio signals. This collection of software tools extends the work of Jean-François Charles, providing an efficient and novel approach to FFT-based processing.

1. INTRODUCTION

NES-Spectrale was created for people who want to process portions of sounds in the frequency domain. The workflow is as follows: Use a sound file or live audio as a starting point; Record a portion of it in the frequency domain (FFT); Process it in some way, in the frequency domain; Render the result as audio, in the time domain; Record the result.

Inspired by the work of Zack Settel and Cort Lippe (Settel and Lippe 1994), Jean-François Charles (Charles 2008), and the GRM Spectral Transform and Evolution suites of audio processors, the project adapted the patches of Jean-François Charles into a set of FFT processing tools using Ableton/Cycling 74’s Max environment, with Jitter matrices storing and processing the FFT. This project’s aim was to explore the notion of “timbre processing” using Jitter matrices as the store and processing mechanism.

The NES-Spectrale suite is as follows:

- Interpolate-periodic, freezes two slices of sound and interpolates between their frequency content over the set interpolation period;
- Multi-FX-05, lets you select one of five Jitter effects to apply to the FFT matrix;
- Playtwo, has two independent play heads that can be adjusted to play across different frequency bands at different rates;
- Playfour, the same, but with four play heads;
- Transform-Mx, uses the jit.mxform2d object, which performs a 2-dimensional matrix transform on an input matrix. It can be used to perform scaling, rotation, skewing, and perspective operations.

2. BACKGROUND RESEARCH

2.1. David Hirst

In 1985, I submitted a Masters thesis entitled Digital Sound Analysis and Synthesis Using the Short-time Fourier Transform (Hirst 1985). As the title suggests, this thesis explores the use of the short-time Fourier transform (STFT) techniques in digital sound analysis and synthesis, but with particular reference to musical applications. It built upon the work of Michael R. Portnoff whose 1980 doctoral dissertation was published in the form of two papers in the same IEEE journal (Portnoff 1981a, 1981b).

The first application of my Masters work to musical systems was in the analysis of acoustic phenomena. A FORTRAN program was written to analyse sound using the STFT, make some data reduction, then display the analysis data on a terminal to allow for human interaction in the evaluation of important sound information. This was quite a laborious process since the sound(s) had to be digitally recorded on a PDP 11/10 mini computer tape within the Music Department. The digital tape was then transported to La Trobe University’s computer centre, where an overnight batch process would run on a Vax 11/780 mainframe computer to mount the tape, read the sounds file(s), perform the analysis, and store the analysis data on disk. From there it could be read and displayed on a specific type of computer terminal, or visual display unit back in the Music Department.

A method of additive synthesis was devised, for the software synthesis program MUSIC4BF, that used the time-varying amplitude and frequency parameters of important partials. As a test of the analysis-synthesis system, a number of Marimba tones were resynthesized, employing differing types of mallets, and the audible results evaluated.

The third application of the analysis system was in the direct coding and modification of the short-time Fourier transform and subsequent inverse transform resynthesis using the Fast Fourier Transform. Different signal modifications tested included fixed and time-varying filtering, spectral bin shifting, pitch transposition independent of time, time-scale expansion independent of frequency, and signal mixing through transform addition. In this latter application a method of phase compensation...
was proposed that used difference equations in the calculation of phase change, interpolation techniques and the overlap-add synthesis procedure with a specified amount of overlap. At the time, none of this work was possible in real time.

The final test of any analysis-synthesis system is in its use in a musical context. With this in mind, the composition Betameta was created to provide an artistic end to a convoluted technical path. Betameta was stochastically composed using a changing beta function to organize the attack points, durations, melodic intervals and stereo placement of the sonorous material. The organization of timbre used both the Marimba simulation instrument and the extension of the instrument derived from the initial experiments. Betameta was selected for performance at the International Computer Music Conference (ICMC) at the Eastman School of Music, New York in 1983.

2.2. Mark Dolson

At the same 1983 ICMC at Eastman, Mark Dolson presented a paper on the ‘Musical Applications of the Phase Vocoder’. Also inspired by Portnoff, Dolson’s work was published in the form of a tutorial article in the Computer Music Journal (Dolson 1986). It went on to become the definitive reference article for the study of the “phase vocoder” (as it was beginning to be called then).

In the 1986 article, Dolson provided two interpretations of the phase vocoder. Firstly as a filterbank interpretation, then as a Fourier transform interpretation, and showed how they are equivalent. He also discussed phase unwrapping, the trade off between time resolution and frequency resolution, and he provided some examples of applications such as analysis, time scaling and pitch transposition. In his conclusion, he briefly mentioned further processing possibilities:

In addition to simple time scaling and pitch transposition, it is also possible to perform time-varying time scaling and pitch transposition, time-varying filtering (e.g., cross synthesis), and nonlinear filtering (e.g., noise reduction), all with very high fidelity. The phase vocoder analysis capabilities alone can be extremely useful in applications ranging from psychoacoustics to composition … (Dolson 1986: 24-25)

2.3. Zack Settel and Corte Lippe

By 1994, Zack Settel and Corte Lippe, working at IRCAM in Paris, were able to take advantage of specialized real-time signal processing equipment, the IRCAM Signal Processing Workstation (ISPW), to explore real-time musical applications which “made use of FFT/IFFT-based resynthesis for timbral transformation in a compositional context” (Settel and Lippe 1994: 171).

The FTS-Max programming environment developed by Miller Puckette at IRCAM (Puckette 1991) enabled Settel and Lippe to develop algorithms and basic operations to process the FFT of an audio signal. Using the overlap-add technique, their basic order of operations was thus:

1. Window the input signals
2. Transform the input signals into the spectral domain using the FFT
3. Perform operations on the resulting spectra
4. Resynthesis of the modified spectra using the IFFT
5. Window the output signal

(Settel and Lippe 1994: 172)

The processes they explored were: high-resolution filtering; low dimensional control of complex spectral envelopes; cross synthesis (multiplying two spectra); mapping qualities of one signal onto another; frequency dependent spatialization; and a frequency-dependent noise gate.

2.4. Jean-François Charles

Jean-François Charles’ ‘Tutorial on Spectral Sound Processing Using Max/MSP and Jitter’ (Charles 2008) explores graphical spectral analysis and synthesis in real-time using the Jitter matrix processing capabilities of the Max visual programming environment.

Charles takes an FFT of a signal and stores the FFT data in a two “plane” matrix, one plane for amplitude and one plane for phase (Charles 2008: 90). Each plane consists of a 2D grid where the grid height is the number of frequency bins (half the FFT size) and the grid length is the analysis windows (or total frames). Once the FFT of a sound is stored in this way, operations can be performed on the FFT matrices using standard Jitter transformations.

The simplest type of sound modification is recording and playback, where the playback speed can be varied. Charles points out that when playing back at slower speeds, there can be an observably audible “frame effect”. To combat this effect Charles implements two methods to interpolate spectra between two recorded FFT frames. The first method makes use of the jit.xfade object to cross-fade between two spectral frames. The second method is a controlled stochastic spectral synthesis, with a probability of picking up values from the next frame specified as a fractional value given by the user. This choice is made for each bin. This stochastic method has the advantage in that it can be extended over a number of frames (e.g. 5 frames) to create a blurring effect.

Charles then goes on to outline graphically based transformations of the FFT representation in the form of direct transforms, use of masks, interactions between sounds, and mosaicing. He also provides examples of real-time freezing, automatic cross-fading, and melody-to-harmony creation using a modified freeze tool. The topics of transient detection and signal segmentation are also covered.
3. NES-SPECTRALE DETAILS

3.1. NES-Spectrale overview

The NES-Spectrale suite extends the work of Jean-François Charles by implementing specific examples of FFT processing using the special capabilities of Jitter matrices and their operators.

The normal way of utilizing NES-Spectrale is as follows: Record a portion of sound as a series of successive spectra in the frequency domain (FFT); Process it in some way - in the frequency domain; Render the result as audio, in the time domain; Record the result.

The NES-Spectrale suite is summarised in Table 1.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NES-Playtwo</td>
<td>has two independent play heads that can be adjusted to play across different frequency bands at different rates</td>
</tr>
<tr>
<td>NES-Playfour</td>
<td>has four independent play heads that can be adjusted to play across different frequency bands at different rates</td>
</tr>
<tr>
<td>NES-Interpolate-periodic</td>
<td>freezes two slices of sound and interpolates between their frequency content over the set interpolation period</td>
</tr>
<tr>
<td>NES-Interpolate-periodic-02-ST</td>
<td>This is a stereo version of NES-Interpolate</td>
</tr>
<tr>
<td>NES-Distort-Del-Fbk</td>
<td>Combines non-linear distortion, multiplication and addition to the FFT index, and adds delay and feedback, frequency bin by bin</td>
</tr>
<tr>
<td>NES-Multi-FX-05</td>
<td>lets you select one of five Jitter effects to apply to the FFT matrix</td>
</tr>
<tr>
<td>NES-Transform-Mx</td>
<td>uses the jit.mxform2d object, which performs a 2-dimensional matrix transform on an input matrix. It can be used to perform scaling, rotation, skewing, and perspective operations</td>
</tr>
</tbody>
</table>

Table 1. NES-Spectrale suite of transformations.

4. DESCRIPTION OF NES-SPECTRALE PROCESSORS

NES-Playtwo

NES-Playtwo has two independent play heads. They can be adjusted to play across different frequency bands using the Lo_Freq and High_Freq sliders.

NES-Playfour

NES-Playfour has four independent play heads. They can be adjusted to play across different frequency bands using the Lo_Freq and High_Freq sliders.

This is merely a four head version of NES-Playtwo with the same kind of controls.

With each of these processors, playing direction can be forward or reverse.

NES-Interpolate-periodic

NES-Interpolate-periodic (Fig. 3) freezes two slices of sound and interpolates between their frequency content over the set interpolation period. By default, this is 3 seconds, but the value can be varied from very short to very long. As long as the soundfile is playing, this periodic snapshot of the spectrum of the sound will continue, with interpolation happening between the latest two snapshots. Turn “Interpolation” ON and power spectrum 1 will appear and then power spectrum 2 will appear after 3 seconds. As Figure 3 shows, the interpolated spectrum

1 Note that the screen shots in this article depict the general layout only. For detailed views, download the NES-Spectrale suite from https://davidhirst.me/software/ and run the patchers using Max 8 from https://cycling74.com
will appear below the other two. Turn Bypass FX off, and you will hear a progression from one timbre to the other. The wet/dry control determines how much of the original and the effect you can hear.

Figure 3. NES-Interpolate-periodic.

“Elapsed Time” shows the current time since the last spectral snapshot. When it reaches the interpolation time, a new spectral snapshot happens, and a new interpolation begins between the latest two spectra.

“Denoise” allows you to remove frequency bands below the set amplitude threshold. “Freeze 1” and “Freeze 2” allow you to manually jump to a new freeze point in the looping sound file. “Play 1”, “Play 2”, and “Play Inter” permit you to play either snapshot alone, or the interpolation between the two. It is normally set to “Play Interp”.

NES-Distort-Del-Fbk

NES-Distort-Del-Fbk combines non-linear distortion, multiplication and addition to the FFT index, and adds delay and feedback, frequency bin by bin. Figure 4 shows that you can draw in the function tables to change the feedback and delay on a per FFT bin basis. This is similar to the transfer function used in waveshaping synthesis, however the distortion is in the frequency domain rather than the time domain.

Figure 4. NES-Distort-Del-Fbk effect from the NES-Spectrale suite.

Drawing in the ‘distortion shape’ function will create more distortion the further the plot moves away from a 45 degree type line, often yielding unpredictable results. Experiment, especially in the very low values, to produce unusual, distorted sounds. You can always get back to the original line by clicking ‘Open’ to open a pop-up menu of functions and choosing the ‘Default – no distortion’ option.

NES-Multi-FX-05

NES-Multi-FX-05 lets you select one of five Jitter effects to apply to the FFT matrix. There is a TAB to select the effect, and each has its own interface, with parameters you can change.

Before getting into the details of the effects and how they are controlled, first a comment regarding the novelty of these processes. The first three patches, already described above, store the FFT in a Jitter matrix form, but are really using the matrix as an efficient storage mechanism to perform fairly traditional playback of audio when converted back from frequency domain form. With NES-Multi-FX-05, we are really using Jitter transforms that were designed for visual transformation to change the stored FFTs in a way that may not have any acoustic basis whatsoever, but which can nonetheless yield interesting and creative musical results. It is really five patches in one, and with any one of the five different transforms, experimentation is the key. The results will depend on that experimentation and be highly dependent on the nature of the sound source. With experimentation in mind, the common elements of the five effects will be summarized, then there will be a section on each effect.

There is a “limiter” built into the patch. Its controls can be accessed by pressing the “Open” button on the left hand side below the “monitor stereo input” button. Its level can be adjusted here too. Processing the FFT can lead to loud sounds. The limiter is there to account for this, but it is always wise to start with all the levels in the chain turned down, then slowly bring the levels up to a comfortable listening volume – starting at the input source and working towards the output destination.

Each of the effects will now be summarized in turn. Note that the descriptions are from the Max Jitter documentation, so they apply to the visual effect on the FFT window. The sonic effect can only be predicted by trying various settings out.

Plur

Peace Love Unity Rave

Figure 5. NES-Multi-FX-05: Plur
Use the jit.plur object to perform linear interpolation on incoming matrix frames. This object resamples an image and then interpolates back to the original size in the following manner.

The resampling process uses two different attributes along each axis of the matrix frame. The ‘step’ attributes determine how the frame is divided at output. For instance, with x_step set to 5, there will be a resampled rectangle every 5 cells along the horizontal axis. The ‘range’ attributes determine how that resampled rectangle is interpolated. If x_range is less than or greater than x_step, different corner points are used for the interpolation calculation, than are used in the resampling. This tool has been found to be useful for creating rhythmic effects: either rapid or slow.

Streak

![Figure 6. NES-Multi-FX-05: Streak](image)

The jit.streak object uses a specified probability to determine the chance that a given matrix cell’s value will be extended to subsequent cells (with an optional scaling factor). The result is a pointillistic variation of the source sound.

Slide

The jit.slide object performs cellwise temporal envelope following.

![Figure 7. NES-Multi-FX-05: Slide](image)

The slide down factor (default = 1) refers not to spatial change along the y axis, but rather with respect to amplitude (e.g. brightness of color channel for image data).

The slide up factor (default = 1) refers not to spatial change along the y axis, but rather with respect to amplitude (e.g. brightness of color channel for image data). The net audio effect is one of blurring the sound.

Sprinkl

jit.sprinkle probabilistically determines whether a matrix cell will be displaced by a random amount along the horizontal or vertical axes to produce a "cloud" of data surrounding the original cell values.

![Figure 8. NES-Multi-FX-05: Sprinkl](image)

The probability that any given cell will be displaced is determined by the ‘prob’ variable (default = 0). The displacement range along the horizontal axis uses ‘x_range’ (default = 0), and the displacement range along the vertical axis uses y_range (default = 0). The audio result is pointillistic, like the streak effect, but if the playback is slowed down a lot, new “melodies” can be generated through the frequency changes.

Altern

Overlays a screen onto incoming matrices. The original matrix values are revealed through "gaps" in the screen.

![Figure 9. NES-Multi-FX-05: Altern](image)

The width of the vertical band "gaps" in the screen is specified by colwidth (default = 1). The height of the
horizontal band "gaps" in the screen is determined by rowheight (default = 1), and the luminance threshold as an average of plane values is varied using thresh (default = 0). Input matrix values above this threshold are displayed as a grid of rectangular "gaps" between the overlaid horizontal and vertical screen stripes. Values below the threshold are displayed. (Usually leave this on ‘0’ to get an image.)

‘xinterval’ specifies the spacing of "gaps" in the screen along the horizontal axis (default = 1), and ‘yinterval’ the spacing of "gaps" in the screen along the vertical axis. (default = 1).

This ‘screen’ or ‘grid’ type of effect is reminiscent of the grids used by Xenakis, but with respect to time and frequency only. Altern can be used to create rhythmic effects or changes in timbre, or both.

**NES-Transform-Mx**

NES-Transform-Mx uses the jit.mxform2d object, which performs a 2-dimensional matrix transform on an input matrix. It can be used to perform scaling, rotation, skewing, and perspective operations. Jit.mxform2d is a complicated transformational object, and I have tried to label the interface sliders in plain language to describe the function. Note that their values have been limited to a useable range. The best way to learn the functions is to try and vary each one on its own by adjusting it a small amount. The colour bar visual will guide you as to what each parameter is doing – skewing, scaling, or rotating. The observed effect is totally dependent on the very important “Boundmode” parameter, so its values and meanings are reproduced in the Readme file for the suite.

This tool is especially adept at glissandi-type effects.

![Figure 10. NES-Transform-Mx](image)

**5. CONCLUSION**

The NES-Spectrale suite is a collection of FFT-based signal processors. NES-Spectrale utilizes an efficient way of storing an FFT in a Max/Jitter matrix, and it also facilitates fast processing of these matrices using Jitter’s built-in matrix manipulation functions. In addition to speed and efficiency, using Jitter matrices affords novel ways of manipulating sound in the frequency domain. Especially useful are the NES-Interpolate-Periodic; NES-Playfour; NES-Transform-Mx; and variable speed playback processing tools. With extended practice, the user can learn some very inventive and creative applications for NES-Spectrale.

As a recent extension to the work, several NES-Spectrale transformations have been written as plugin devices for Ableton Live 10: NES-Interpolate.amxd and NES-SpectralDelayDistort.amxd

These have been carefully programmed to allow for multiple instances in Ableton Live.

![Figure 11. NES-Interpolate.amxd Max for Live device.](image)

**6. REFERENCES**


Author/s:
Hirst, D

Title:
NES-spectrale: a suite of max tools for processing in the frequency domain

Date:
2019

Citation:

Persistent Link:
http://hdl.handle.net/11343/233749

File Description:
Published version