DYNAMIC STOCHASTIC WAVETABLE SYNTHESIS

Raphael Radna

Department of Music
University of California, Santa Barbara
Santa Barbara, CA
rradna@ucsb.edu

ABSTRACT

Dynamic Stochastic Synthesis (DSS) is a direct digital synthesis method invented by composer Iannis Xenakis and notably employed in his 1991 composition *GENDY3*. In its original conception, DSS generates periodic waves by linear interpolation between a set of breakpoints in amplitude–time space. The breakpoints change position each period, displaced by random walks via high-level parameters that induce various behaviors and timbres along the pitch–noise continuum. The following paper proposes Dynamic Stochastic Wavetable Synthesis as a modification and generalization of DSS that enables its application to table-lookup oscillators, allowing arbitrary sample data to become the basis of a DSS process. We describe the considerations affecting the development of such an algorithm and offer a real-time implementation informed by the analysis.

1. INTRODUCTION

Iannis Xenakis proposed Dynamic Stochastic Synthesis (DSS) as a time-domain method of producing "complex sonorities" with "numerous and complicated" transients [1]. In DSS, the cyclical portion of a periodic wave (*wave cycle*) is defined by a number of breakpoints in amplitude–time space. Waves are produced by linear interpolation between adjacent breakpoints. The breakpoints shift positions each period, continuously affecting the pitch, amplitude, and timbre of the synthetic tone produced and giving rise to its "dynamic" character (Fig. 1).

The "stochastic" element refers to the displacement of the breakpoints in both dimensions by random walks. Various probability distributions (Cauchy, logistic, etc.) can be applied, each affecting the movement of the breakpoints in its distinctive way [2]. Xenakis also specified high-level parameters to constrain the displacement: the random walk step size governs its magnitude, and elastic barriers reflect excessive values back within a specified range. These parameters influence the degree of similarity between successive wave cycles. If only slight variation is permitted, tones of stable pitch and timbre are produced; conversely, a parameter state that allows profound dissimilarities between wave cycles causes the output to tend towards noise. Additionally, the number of breakpoints used correlates with spectral brightness.

The original DSS software *GENDY* dates from the late 1980s and operated in a non-realtime capacity. Since then, several researchers have implemented DSS with experimental alterations

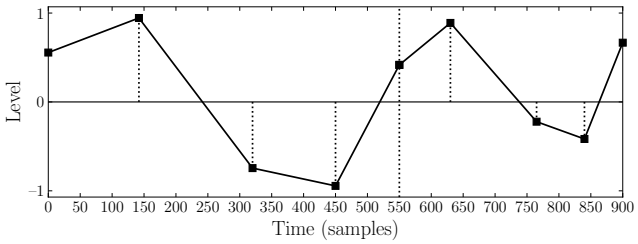

Figure 1: *Two contiguous DSS wave cycles. The second is a variation of the first, produced by stochastic displacement of its four breakpoints. The period of the second cycle is shorter, indicating an increase in pitch.*

that seek to enhance its sound or functionality in some way, including interactive operation and analysis [3], time-variant parameter automation [4], wave-cycle sequencing strategies [5], touch-sensitive and gestural interfaces [6], and applications of physical models to the algorithm [7, 8]. Our own previous DSS-related research culminated in the *Xenos* plug-in synthesizer, which introduced pitch quantization to DSS [9].

While these contributions have all helped to extend and sustain interest in DSS, none have addressed the inherent limitation of its basis in breakpoint interpolation synthesis. One possibility in this direction is the application of DSS to standard wavetable oscillators. Although DSS does not read a lookup table directly, it realizes equivalent sample data at runtime through breakpoint interpolation; this process bears similarity to dynamic wavetable techniques, such as scanned synthesis [10, 11]. Dynamic Stochastic Wavetable Synthesis (DSWS) thus uses the procedures of DSS to apply its characteristic, stochastic pitch and timbre evolution to arbitrary sample data, instead of generating abstract waves from linear ramps. In this way, DSWS reimagines DSS as an audio processor rather than a synthesizer, increasing the timbral range of the technique, enabling general DSS-based modulation, and facilitating interpolation between arbitrary timbres and DSS.

2. DYNAMIC STOCHASTIC WAVETABLE SYNTHESIS

The DSWS prototype described in this paper is implemented in Max/MSP using the GenExpr metalanguage for audio programming. The code is open source and available for download from https://github.com/raphaelradna/dsws.

2.1. Wavetable Segmentation

While DSS begins with the definition of breakpoints for linear interpolation, DSWS begins by segmenting a wavetable into a num-

ber of regions whose pitches and amplitudes will be individually manipulated. For simplicity and efficiency, we have chosen to divide the wavetable into an arbitrary number of segments of equal size. Our implementation allows for as few as one segment, in which case the entire wavetable is affected uniformly, or as many as 256. The number of segments is variable at runtime. As in DSS, a greater number of segments results in a brighter timbre (Fig. 2).

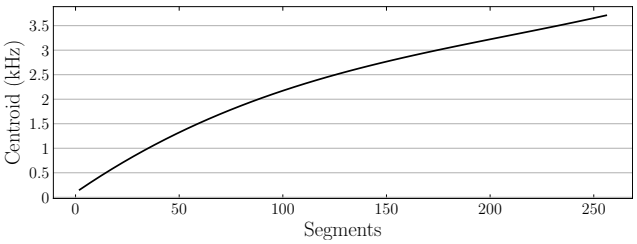

Figure 2: *The spectral centroid increases with the number of wavetable segments. Measurements were taken using a sinusoidal wavetable and with otherwise constant parameters: a steady pitch of A1 (55 Hz) and maximum amplitude fluctuation.*

2.2. Table Lookup, Modification, and Output

DSWS has at its core a wavetable oscillator that derives sample values by reading through a lookup table at a variable frequency [12]. It imposes DSS-like behavior on a wavetable of N samples by dividing it into M segments and applying individual, random, pitch and amplitude deviations to each segment. For each sample in the input wavetable $x[n]$, where $0 \leq n < N$, the index m of its containing segment is the greatest integer less than M multiplied by ϕ, its phase within the wavetable:

$$m = \lfloor M\phi \rfloor \,, \tag{1}$$

where the phase ϕ in range $0 \leq \phi < 1$ is given by

$$\phi = \frac{n}{N} \,. \tag{2}$$

The deviations are regenerated each period and stored in series P and A, respectively, each of length M. The pitch deviation $P[m]$ of the segment containing $x[n]$ modulates the base oscillator pitch p. These values, initially expressed as floating-point MIDI notes, are summed, converted into a frequency in Hz, and divided by the sampling rate f_s to produce a phase increment φ for table lookup:

$$\varphi = \frac{440 \cdot 2^{\frac{p+P[m]-69}{12}}}{f_s} \,. \tag{3}$$

The effective frequency therefore changes with each segment as the table is read and fluctuates around p. By contrast, the per-segment amplitude deviation is added to the wavetable data. To avoid introducing discontinuities into the wave cycle, an individual amplitude deviation a is derived for each input sample $x[n]$ by linear interpolation between the amplitude deviation $A[m]$ of its containing segment and that of the subsequent segment $A[k]$:

$$a = (1-\mu)A[m] + \mu A[k] \,, \tag{4}$$

with the index k of the subsequent segment given by

$$k = \begin{cases} m+1, & \text{if } m < M-1 \\ 0, & \text{if } m \geq M-1 \end{cases} , \tag{5}$$

and the interpolation parameter μ attained by

$$\mu = M\phi - m \,. \tag{6}$$

Finally, a (4) is added to $x[n]$ to produce output sample $y[n]$. To prevent clipping, any $y[n]$ greater than 1 or less than -1 is reduced or increased, respectively, by the amount d that it lies out of range:

$$y[n] = \begin{cases} 1-d, & \text{if } x[n]+a > 1 \\ -1+d, & \text{if } x[n]+a < -1 \\ x[n]+a, & \text{if } -1 \leq x[n]+a \leq 1 \end{cases} , \tag{7}$$

where

$$d = |x[n]+a| - 1 \,. \tag{8}$$

DSWS can thus be conceptualized in part as a segmented, stochastic wavefolder (Fig. 3) [13].

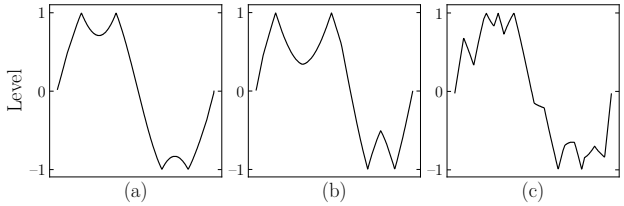

Figure 3: *Asymmetrical amplitude folding of a sinusoidal wavetable resulting from a DSWS process with (a) two segments, (b) four segments, and (c) sixteen segments.*

2.3. Deviation Generation and Iteration

The per-segment pitch and amplitude deviations are produced by random walks that are iterated every wave cycle. New values are generated in the range $[-1, 1]$ according to some probability distribution (we use uniform randomness for demonstration purposes) and added to the previous deviations to produce new ones. Two parameters influence this process: the step size scales the random value, constraining its magnitude and thus limiting the difference between deviations across cycles, and the barrier position defines the random walk space, i.e., the minimum and maximum possible deviation values. The sum of the previous deviation and new random value can in general fall outside of the range defined by the barriers; following Xenakis's design, any such sums are reflected back into range in the manner of (7) and (8).

Because the random value can be either positive or negative, the deviation can either increase or decrease from one wave cycle to the next, regardless of the step size. The barrier position parameter limits the range of the random walks symmetrically around a single value; the amplitude walks center around zero, while the pitch walks center around p, the base oscillator pitch. As a result, reducing both barrier position parameters to zero reproduces the input wavetable at a constant pitch. The parameters of the pitch random walk are specified in equal-tempered semitones, and those of the amplitude random walk are specified as proportions of the full amplitude range.

2.4. Single-Segment Pitch Fluctuation

Manipulating individual sections of a wavetable in the manner of DSWS introduces knees in the output wave at the segment bound-

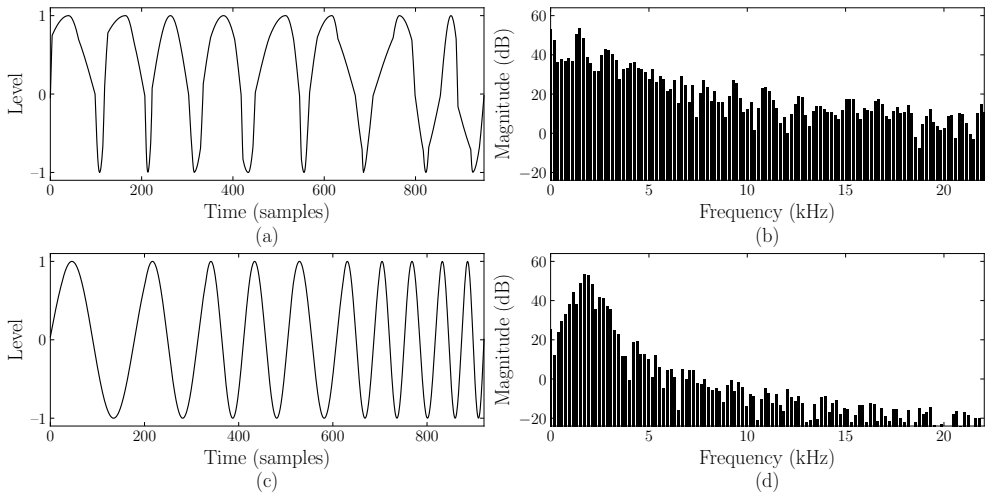

Figure 4: *Effects of DSWS pitch fluctuation on the waveform and spectrum in (a), (b) standard and (c), (d) single-segment modes. The deformations visible in waveform (a) are caused by eight extreme pitch deviations per wave cycle. These scatter partials throughout spectrum (b), including by aliasing. Because the frequency of waveform (c) modulates only once per cycle, its sinusoidal shape is preserved, producing spectrum (d), which shows less energy in the high-frequency range despite otherwise identical parameters: a center pitch of C5 (523.25 Hz), pitch barrier range of ± two octaves, pitch step size of six semitones, and no amplitude fluctuation.*

aries, causing high-frequency distortion. While the amplitude fluctuation writes these directly into the sample data, the pitch fluctuation also causes them implicitly by modifying the phase increment for each segment, potentially tens or hundreds of times per cycle. As a result, pitch fluctuation is not generally timbre-neutral, but also affects the spectrum of the resulting tone.

To better isolate perceived pitch and timbre transformations, we can treat the entire wavetable as a single segment for the purpose of pitch fluctuation, regardless of the number of segments used for amplitude fluctuation. In this case, the pitch fluctuation occurs once per cycle, i.e., at oscillator frequency in Hz. Since this rate typically still falls within the microsound timescale, rapid pitch modulation remains perceptible while timbral distortions are reduced (Fig. 4). This method produces more volatile pitch movement for the same step size and barrier position parameters, because the frequency of the wave cycle depends on a single pitch deviation instead of the average of several. It also contradicts DSS, which stipulates an equal number of pitch and amplitude fluctuations per cycle, but may be subjectively preferable in its adaptation to wavetable synthesis due to its ability to preserve the shape, and therefore timbre, of a particular wavetable. We thus propose single-segment pitch fluctuation as the default behavior of DSWS.

3. DISCUSSION

This section elaborates on our DSWS prototype, providing insight into certain design choices and their ramifications, and suggesting possible alternatives or areas for further development.

3.1. Wavetable Selection and Sound Quality

Sine, triangle, square, and sawtooth wavetables are included in our DSWS implementation. The classical waveforms were produced by additive synthesis using 64 harmonics, which, at a sampling rate of 44.1 kHz, prevents aliasing for fundamental frequencies up to approximately 344.5 Hz. Further antialiasing measures were not taken, as the linear interpolation of DSWS, like that of DSS, ultimately produces its own aliasing artifacts [2]. A more complete implementation could apply additional solutions for antialiased wavetable synthesis, such as those described in [14], and offer band-limited interpolation algorithms for producing the per-sample amplitude deviations [15].

Our implementation can also generate noise wavetables using uniform randomness, load external single-cycle sample data in .wav format, or use an empty wavetable. Since using a wavetable with all values zeroed produces linear ramps between amplitude deviations, we can say that DSS is a special case of DSWS.

3.2. Wavetable Segmentation Method

Although we divide the wavetable into equal parts, other segmentation methods could be applied and affect the results in distinct ways. A piecewise linear approximation algorithm, for example, could fit the segment boundaries to places of pronounced change in the slope of the sample data (Fig. 5). Furthermore, a method that optimizes the fit within a specified error tolerance, as in [16], could determine the ideal number of segments and their boundaries for arbitrary wavetables. Further investigation is needed to assess the significance of the wavetable segmentation method in DSWS.

3.3. Optimization Considerations

The DSWS prototype reads the sample data from a source wavetable, transforms them, and writes the results into a second wavetable, which is read at the oscillator output. Although every wave cycle is regenerated from the source data in this manner, continuity between them is maintained because the deviations are influenced by their own previous values. Our implementation uses a second wavetable primarily for the purpose of visualization; an alternative and possibly more efficient design would calculate the values

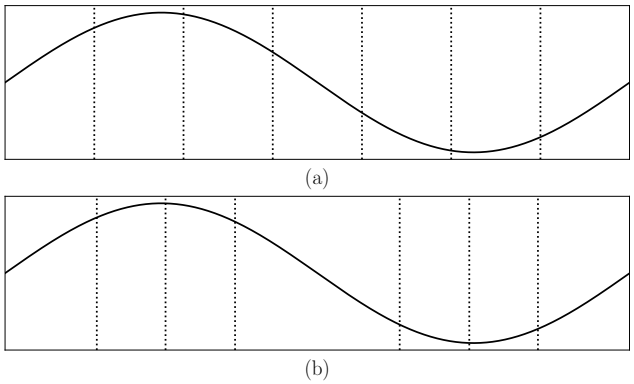

Figure 5: *Segmentation of a sinusoidal wavetable into seven parts using (a) equal distribution and (b) piecewise linear fit via global optimization of the least squares method* [17].

continuously and write them directly to the output buffer, without otherwise storing them.

The per-segment pitch and amplitude deviation data defining the transformation have the same memory footprint as the amplitude–time breakpoints of traditional DSS. In a polyphonic implementation using this method, each voice would have unique arrays of deviations but read a single source wavetable, avoiding expensive array-copy operations. Because the barrier position parameters control the random walk space, i.e., the degree of pitch or amplitude deviation from the source, this approach also affords interpolation between the original wavetable and its transformation.

3.4. Limitations

A complete reproduction of all DSS features is beyond the scope of this work. Therefore, certain components of the mature form of the technique, namely second-order random walks and variable probability distributions, have not been implemented. This decreases the number of parameters, which simplifies the prototype, but also reduces its sound design potential.

Additionally, because the per-segment pitch fluctuations affect table read frequency and are not written explicitly into the sample data, the built-in waveform display object in Max/MSP does not represent the horizontal distortions they produce. These are visualized, however, in the oscilloscope-style display.

4. CONCLUSION AND FUTURE WORK

We have described DSWS, an experimental synthesis method exploring the application of Iannis Xenakis's DSS algorithm to table-lookup oscillators. We also presented a prototype demonstrating its basic principles. The technique could be developed further, e.g., by integrating additional probability distributions, segmentation methods, and interpolation algorithms; or by combining with sophisticated wavetable techniques, such as wavetable crossfading and multiple-wavetable synthesis [18]. Furthermore, by expanding DSS into a sample-processing paradigm, DSWS suggests implementation as a filter that applies an iterative window of stochastic pitch and amplitude distortions to streaming audio input.

5. REFERENCES

[1] Iannis Xenakis, *Formalized Music: Thought and Mathematics in Composition*, Pendragon Press, Stuyvesant, revised edition, 1992.

[2] Marie-Hélène Serra, "Stochastic Composition and Stochastic Timbre: *GENDY3* by Iannis Xenakis," *Perspectives of New Music*, vol. 31, no. 1, pp. 236–257, 1993.

[3] Peter Hoffmann, "The New GENDYN Program," *Computer Music Journal*, vol. 24, no. 2, pp. 31–38, 2000.

[4] Andrew Brown, "Extending Dynamic Stochastic Synthesis," in *Proc. 2005 Int. Computer Music Conf.*, Barcelona, Spain, 2005, pp. 111–114.

[5] Sergio Luque, "The Stochastic Synthesis of Iannis Xenakis," *Leonardo Music Journal*, vol. 19, pp. 77–84, 2009.

[6] Nick Collins, "Implementing Stochastic Synthesis for SuperCollider and iPhone," in *Proc. Xenakis Int. Symposium*, 2011.

[7] Luc Döbereiner, "Phingen: A Physically Informed Stochastic Synthesis Generator," in *Proc. 2011 Int. Computer Music Conf.*, Huddersfield, UK, 2011, pp. 57–60.

[8] Emilio Rojas and Rodrigo Cádiz, "A Physically Inspired Implementation of Xenakis's Stochastic Synthesis: Diffusion Dynamic Stochastic Synthesis," *Computer Music Journal*, vol. 45, pp. 48–66, 2022.

[9] Raphael Radna, "Xenos: Xenharmonic Stochastic Synthesizer," M.S. thesis, University of California, Santa Barbara, 2022.

[10] Robert Tubb, Anssi Klapuri, and Simon Dixon, "The Wablet: Scanned Synthesis on a Multi-Touch Interface," in *Proc. 15th Int. Conf. Digital Audio Effects (DAFx-12)*, York, UK, 2012, pp. 192–199.

[11] Tendsin Mende, Lars Engeln, Matthew McGinity, and Rainer Groh, "Creative sound modeling with signed distance fields," in *Mensch und Computer 2022 - Workshopband*, Bonn, 2022, Gesellschaft für Informatik e.V.

[12] Curtis Roads, *The Computer Music Tutorial*, MIT Press, Cambridge, MA, 1996.

[13] Fabián Esqueda, Henri Pöntynen, Vesa Välimäki, and Julian Parker, "Virtual Analog Buchla 259 Wavefolder," in *Proc. 20th Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, 2017, pp. 192–199.

[14] Günter Geiger, "Table Lookup Oscillators Using Generic Integrated Wavetables," in *Proc. 9th Int. Conf. Digital Audio Effects (DAFx-06)*, Montreal, Canada, 2006.

[15] Julius O. Smith, *Digital Audio Resampling Home Page*, http://www-ccrma.stanford.edu/~jos/resample/, January 28, 2002.

[16] Bernd Hamann and Jiann-Liang Chen, "Data Point Selection for Piecewise Linear Curve Approximation," *Computer Aided Geometric Design*, vol. 11, pp. 289–301, 1994.

[17] Charles Jekel and Gerhard Venter, "pwlf: A Python Library for Fitting 1D Continuous Piecewise Linear Functions," 2019.

[18] Andreas Franck and Vesa Välimäki, "Higher-Order Integrated Wavetable Synthesis," in *Proc. 15th Int. Conf. Digital Audio Effects (DAFx-12)*, York, UK, 2012.