

Killing Chipmunks with Parametric Dynamic Wavetable Synthesis

TOM SCHOUTEN

August 4, 2003

Abstract

In this paper we discuss some sound synthesis methods in the dynamic wavetable framework. We will take a closer look at approaches that combat problems regarding aliasing and pitch dependent formant locations, two of the major drawbacks of dynamic wavetable synthesis (DWS). DWS is a hybrid form of wavetable synthesis and additive/granular synthesis. We will limit ourselves to “holistic” DWS, which uses (semi) parametric methods to cancel pitch dependent formant location, as an alternative to basic additive or granular synthesis for signals with a clear single pitch.

1 Introduction

Wavetable synthesis is one of the first digital sound synthesis methods developed [refs]. The method stores a period of a sound wave in a table and plays it back at a variable pitch. The data in the table determines the timbre of the resulting waveform. A drawback of this form of synthesis is the fact that formant frequencies of the resulting waveform are proportional to the playback frequency. This is frequently called the “chipmunk effect” and is usually undesirable. In commercial synthesizers this artifact is reduced by using several wavetables for some pitch ranges and interpolating between them. This is referred to as multitimbral synthesis.

Another drawback of wavetable synthesis is aliasing in the case of polynomial interpolation. This is solved in practice by using a decimating filter. This

is also called bandlimited interpolation. This usually requires the storage of a large set of interpolation coefficients with different cutoff frequencies, since it is hard to impossible to do this in a parametric way.

Dynamic wavetable synthesis is a derivative of wavetable synthesis, with the addition that the wavetable is synthesised using some (parametric) method, often updated at a haptic rate of around 20Hz. This enables the construction of dynamic timbres. However, being a table playback method, the chipmunk effect is also present in this method and aliasing needs to be taken into account when the table is read out at a higher rate than the spacing between table samples.

In this paper we will illustrate some applications of DWS, with special attention on the chipmunk effect and aliasing. The first method deals with explicit additive synthesis using the fast fourier transform. In short we synthesize the spectrum of the periodic waveform directly, and introduce pitch by resampling the waveform on playback. When the pitch of the playback is known (i.e. by lookahead) the wavetable can be made explicitly bandlimited. If dilatation and contraction of the spectrum are parametric, we can use the playback pitch information to perform a formant correction before the waveform is transformed to the time domain. This approach has some advantages over synthesizing sinusoidal contributions with standard, direct FFT approaches such as FFT^{-1} [2], especially for near periodicity.

The second topic in this paper is an illustration of the general framework of scanned synthesis using dynamic wavetables. The idea is to derive a waveform, or timbre, from a (nonlinear) dynamic system.

The holy grail here is to find some way to change the formant spectrum of a wavetable, depending on the desired playback pitch. We will pursue this using circulant neural networks, as a generalization of a linear circular time invariant system.

2 DWS algorithm

2.1 DWS

The algorithm is very simple. The wave table is a vector function $x[k] \in \mathbb{R}^N$, with a sampling rate f_c which is a fraction of the system sample rate f_s . Sound is synthesized by interpolating the $x[k]$ at a desired pitch. To prevent discontinuities when the table is updated, the readout can be made to interpolate between $x[k - 1]$ and $x[k]$. The choice of N depends on the desired frequency resolution. When $N = 2048$ and $f_s = 1024f_k$ and f_s in the order of 40 kHz the entire audible spectrum can be represented, together with the interesting haptic frequencies. A choice of sampling subdivision and table length of 1024 is usually more than enough for practical purposes.

The computational cost in its basic form is relatively low. The playback resampling can be done using simple polynomial interpolation. Using more expensive methods like bandlimited interpolation is also possible, however for some uses (FDDWS) playback aliasing can be cancelled explicitly, so an expensive decimating filter is not necessary. Usually the cost of synthesizing the wavetable (using convolution methods) can be significantly reduced by using an FFT.

2.2 FDDWS

Since working in the frequency domain has a more intuitive feel, we will give the method where the table is transformed using an IFFT before resampling playback a different name: frequency domain dynamic wavetable synthesis (FDDWS).

One of the advantages over direct additive synthesis methods is the elegance of the scheme. We don't have to care about things like window length and spacing, and window transfer function, which are se-

rious drawbacks of standard, ad-hoc frequency domain processing methods. Since elegance is not necessarily an argument in engineering, we claim the real advantage is simplicity of the parametric synthesis phase, which will ease implementation and save computational cost when the expense of the spectrum synthesis far outweighs the cost of the resampling involved.

When employing frequency domain DWS, anti-aliasing is a trivial multiplication before performing an IFFT. The only real problem that needs to be taken into account is to make the spectrum "stretchable". When killing the chipmunks is not necessary, since the effect might be desired in some cases, it can be simply left out. In short, by introducing separate pitch and formant controls, a lot is possible. Combatting chipmunks with FDDWS is thus creative use of time/frequency contraction and dilatation.

Requiring parametric spectrum dilatation or contraction might seem a serious disadvantage. We can also look at it as a trade-off for the added simplicity of the frequency domain processing. Since the transform lengths are fixed we are freed of window juggling. We hope the illustrations in the next sections will clarify this simplicity.

2.3 Time-Frequency Duality

All in all, DWS enables us to do pitch relative processing using the entire dsp toolbox in a simplified form because the data size remains constant. The slight increase in cost and the straightforward trade-offs involving aliasing and formant locations give us a very flexible and intuitive framework as return of investment.

When working in the complex field, we have a nice dual representation that allows us to switch between time and frequency representation without too much hassle, and interchange the dual operations of circular convolution and ring modulation.

3 Parametric Formant Preserving Wavetable Synthesis

We will illustrate this principle by explicitly synthesizing “constant-Q” formants.

4 Dynamical Systems

This section deals with the synthesis of tables using a parametric (nonlinear) dynamical system. This is in fact equivalent to Scanned Synthesis, pioneered by Max Mathews e.a. [ref]. In most cases we are interested in a periodic table without discontinuities. We could define the table on the complex unit circle, even in continuous form to enable some analytic manipulation using i.e. orthogonal polynomials or fractional linear maps, but we will leave that route for later. There are other ways to get to periodicity. We take the wavetable to be $x \in \mathbb{R}^N$.

The system is defined as an iterated map

$$x[n + 1] = f(x[n]). \tag{1}$$

We will also extend this equation to include an input, in the form of $x[n + 1] = f(x[n] + u[n])$ or the more general $x[n + 1] = f(x[n], u[n])$. For simplicity the output (the wavetable) is taken to be the state vector of the system.

4.1 Linear Systems

When we limit ourselves to linear maps $f = A \in \mathbb{R}^{N \times N}$ we get a “standard” linear state space system [1]. A way to introduce circular symmetry in f is to make A circulant. Application of A can then be computed in the frequency domain since it amounts to circular convolution. Working in the frequency domain will also enable us to interpret the working of the network to a large extent, since the FFT (denoted further as the matrix operator F) of the wavetable Fx has a very intuitive interpretation. The operator $A' = FAF^{-1}$ is diagonal containing the spectrum of this filter. Requiring circular symmetry thus brings us to FDDWS.

For the linear case (1) reduces to

$$x[n + 1] = Ax[n]. \tag{2}$$

With A circulant, $x' = Fx$ and the diagonal decomposition $C = FAF^{-1}$ we have a set of decoupled linear systems

$$x'_i[n + 1] = c_i x'_i[n]. \tag{3}$$

With c_i an eigenvalue of A corresponding to one of the eigenvectors of A (an FFT base function). Because we defined A to be real valued, each c_i has a complex conjugate \bar{c}_i . We now have the choice to take $A \in \mathbb{C}^N$ or to replace the eigenvalue decomposition $C = FAF^{-1}$ by its real blockdiagonal decomposition.

The effect of the system is to produce a wavetable composed of a set of harmonic sinusoids, whose phase and amplitude are modulated damped complex exponentials. Scanning x will approximately produce a set of near harmonically spaced damped sinusoids. The damping is affected by the magnitude of the c_i and the frequency shift away from harmonicity is determined by the phase of the c_i .

What we are interested in is to make the magnitude of the frequency spectrum of x (the magnitude of x') parametrizable, so we can cancel out the formant shift introduced by table readout. This would turn the DWS method into a direct, fully separated pitch and timbre synthesis method.

This can be done by changing the time scale of the x_i and c_i whenever the playback speed varies. Doing so will move the aliasing problem to this scaling step. If the c_i are fully parametric, this does not pose a real problem. Stretching the state vector x'_i will pose no problem either, except for the loss of information. Compressing x'_i does pose an aliasing problem. A way to solve this is to stretch the time domain state vector x_i , since frequency domain compression is equivalent to time domain stretching.

All this means the entire algorithm will be very much based on resampling, both for the time and the frequency domain representation of the state vector as for the final playback resampling. This of course turns us back to square one in some sense as to how far it is worth pursuing this approach instead of keeping the pitch and timbre method separated

in a source/filter approach, i.e. using bandlimited impulse synthesis together with an IIR filter bank or overlap add frequency domain filtering. On the other hand, when pursuing this route, we do have a separate pitch and formant control, together with a nice framework processing waveforms and spectra in a combined time/frequency approach.

4.2 Nonlinear Systems

In this section we will use a recurrent neural network with connectivity matrix $A \in \mathbb{R}^{N \times N}$ and squashing function $\sigma(t) \in \mathbb{R}$. We define f as a composition of $x' = Ax$ and $f_i(x) = \sigma(x'_i)$.

Like in the previous linear approach, we take A to be circulant. This network is thus equivalent to a filter followed by a waveshaper, in music-dsp speak.

4.3 Examples

4.3.1 Analog waveform modeling

It is of course debatable whether it is worth using an IFFT for waveform synthesis. However, since the transform is running at a haptic rate, the cost is proportional to $\log N$. Compared to filter based methods for synthesizing analog waveforms, this is not so expensive¹.

Since we're not too much concerned with killing chipmunks, the only thing we need is an anti-aliased IFFT. Most analog style waveforms have a simple fourier transform. Again this can lead to new schemes for synthesizing new "analog sounding" waveforms and is in fact no different from general parametric spectral synthesis.

¹The filter order required to get a decent bandwidth vs. aliasing trade-off for virtual analog is significant, and at least comparable to the complexity of the IFFT. On the other hand, implementing a virtual analog synth is a lot easier to do with just an increase in sample rate. Increasing the sample rate has a lot more advantages. It simplifies building blocks, allows for a smaller wordlength, allows for low order filters to be used to tame nonlinear elements, allows for sigma-delta modulation to perform bit depth reduction when increased precision is necessary, i.e. for high Q recursive filters, etc...

4.3.2 Piano modeling

Modeling the piano is not a simple task. However, some of the features of the piano, like the inharmonicity and triple strings per note fit well into our scheme.

We will use the simplest form of a circulant linear state space system (FDDWS), with nothing but a spectral envelope and a phase shifter (multiplication with unit norm complex numbers) in the feedback path. Reading a table three times with frequencies corresponding to $1, 2 + \Delta_2, 3 + \Delta_3$ to simulate the effect of triple strings already produces a very piano-like sound.

We can introduce inharmonicity by modulating the FFT bin corresponding to basis function $e^{\frac{j2\pi n}{N}}$ by $e^{j(an+bn^2)}$. This will produce a frequency dependent shift, which will shift (a) or bend (b) the harmonic series upward ($a > 0$ or $b > 0$) or downward ($a < 0$ or $b < 0$) from harmonicity. This nonlinearity can even be made signal dependent, i.e. dependent on the power of the system.

Modeling the attack is almost as important as the strings. We use a second FDDWS system with a gaussian envelope in the feedback path, reduced by an overall gain. The wavetable is initialized with a random spectral content when a new note is struck.

4.3.3 Waveform feedback

Since we know the pitch of the scanning playback, we can scan a waveform, do some processing (i.e. a reverb) and scan the output back into the table. This allows for some more coupling between the real world and our DWS model.

Some nice feedback effects can be obtained when both scanning frequencies are not inverses, or a multiple read-out scan is combined with a single read-in scan. Generating a new waveform by stretching or compressing the previous one and feeding it back into the loop can generate a lot of interesting sounds, even natural sounding drum sounds. By adding a static limiter and a lowpass filter in the feedback loop, stability is ensured and some spectral control is possible. The lowpass filter also smooths the edges.

The update equation is $x[k + 1] = L \circ D \circ R\{x[k]\}$ where R is a resampling stretch or compress oper-

ator, D is a waveform limiter (i.e. a hard clipper) and L is a lowpass filter. By windowing $x[k]$ with a cosine window before scanning it for final output, most of the waveform discontinuities are removed and the frequencies are shifted upward which gives extra punch.

Driving the same system with a constant period input, instead of an impulsive excitation, giving the update equation $x[k+1] = L \circ D \circ R\{x[k] + u\}$, with u the constant input, can generate interesting fractal like waveforms, reminiscent of synced sawtooths.

When the limiter is replaced by a power normalizer and the feedback loop contains both a lowpass and a highpass filter, $x[k+1] = N \circ L \circ D \circ R\{x[k]\}$, with N the power normalization operator, a stable oscillator is constructed. This oscillator can produce a wide variety of chaotic sounds. Due to the normalization, the system is nonlinear but energy conserving, and due to the filters, the frequency content can be localized.

Some more variants are possible. Changing the update function to $x[k+1] = N \circ L \circ D \circ (1+aR)\{x[k]\}$, gives an extra control a that is a tuning parameter for the amount of folding applied.

4.3.4 Random remarks

In waveguide string modeling, there are usually two polarizations that are combined, both with different properties. Using the positive and negative part of the spectrum to express something similar in DWS, enables us to use complex FFT's. This is a bit of a moot point, but it enables easier processing in PD, since the data will be in a more natural form, plus we sort of get two signals for the price of one. In other words, if the positive and negative spectrum will receive a different phase shift (so both left and right partials differ slightly in frequency after scanning) we get amplitude modulation for free.

5 Conclusion

Despite the chipmunk effect which sometimes can be a big drawback, dynamic wavetable synthesis is an interesting approach because it enables us to use fast periodic transforms to build systems that can be eas-

ily understood as to how they behave from a time or frequency perspective. When we can counter the chipmunk effect by adjusting system parameters, it is a very useful technique because it eliminates some of the ad-hoc methods used to compensate for the changing periodicity in transform based techniques. In short, if the formant distribution of a wavetable can be shifted in a straightforward parametric way, we can cancel the chipmunk effect completely. Also, when working in the frequency domain, anti-aliasing is "free", in a sense that knowing the readout frequency enables us to explicitly cancel out the frequencies that are bound to fold.