SELF-ORGANISED SOUNDS WITH A TREMOLO OSCILLATOR

Risto Holopainen
Dept. of Musicology,
University of Oslo
Norway
risto.holopainen@imv.uio.no

ABSTRACT

Tremolo is usually regarded as belonging to the domain of note embellishments. Rapid tremolo, taken into the audio range, is an interesting synthesis technique which is related to FM and granular synthesis. We present a tremolo oscillator, capable of a wide range of sonorities, and illustrate some of its capabilities in applications such as feature-based synthesis and sonification. A reference implementation in Csound is given. The tremolo oscillator is then put into a feedback system, where its output is subject to feature extraction, and the extracted features in turn are mapped to its control parameters. Chaotic orbits in this feedback system guarantee continuous variation, in contrast to the trivial periodic patterns that are easily obtained.

1. INTRODUCTION

There are at least two acceptations of the word tremolo: either it is understood to be a rapid alteration between two pitches, as in a trill, but possibly with a wider pitch separation, or it is taken to be a rapid amplitude variation, analogous to the pitch variation of vibrato. It is the pitch alteration that forms the basis of the tremolo oscillator.

A deceptively simple instrument, an oscillator which switches between two pitches, can produce a more varied range of sounds than one might at first suspect. Tremolo is a built in capability of many synthesizers, where it is realised as a square wave modulation of frequency and carried out by an LFO. A glide or portamento parameter is also frequently seen in synthesizers, where its function is to produce smooth pitch transitions (glissandi) between notes played legato. An obvious realisation of portamento is to use a lowpass filter to smooth the frequency control function. Apart from lowpass filtering, other filters can also be used to produce characteristic effects when applied to the frequency control function. These ideas are further explored in section 2, where the tremolo oscillator is introduced.

An instrument's full utility is only seen in concrete applications. We discuss how the tremolo oscillator may be used for sonification, in feature-based synthesis [1], and as part of a self-organising system, which combines aspects of algorithmic composition, chaotic systems, and sound synthesis.

More or less similar self-organising synthesis systems have been explored by other composers, sometimes with reference to cybernetics and ecological systems, as in the works of Di Scipio [2] and Bökesoy [3]. Other examples include the Gendyn system of Xenakis [4], where a waveform continually undergoes random perturbations, causing drift in pitch and timbre, and a synthesis system developed by Arun Chandra, also based on incremental modifications of waveforms with several voices acting differently depending on the current state of other voices [5].

The self-organising tremolo oscillator put into a feedback loop is a closed system; it doesn't take any input once the initial parameters have been specified. The greatest challenge for such closed self-organising systems is how to make them produce an interesting output in the absence of continuous input. We conjecture that the system is capable of producing non-trivial output only when it reaches a chaotic regime. Among the easily achieved trivial results are fixed points, cycles, and transients which eventually settle down on some regular pattern. That the system reaches a fixed point means that it gets stuck on some particular parameter values.

For effective usage of a synthesis model, one should know how control parameters map to perceived qualities. This is particularly true when the synthesis model is itself part of a more complex system. So first we give the details of how the tremolo oscillator works, then in section 3 its application in feature-based synthesis and sonification is briefly discussed. Finally, in section 4 we discuss various ways the tremolo oscillator can be used in a feedback system. In particular, the rate at which the feedback occurs is an important factor.

2. TREMOLO OSCILLATOR

The oscillator is designed to produce a tremolo between two tones. The tones could have any waveform, but in the following we will use a sinusoid. Fast modulation will cause audio rate FM. Smoothing of the frequency variable produces interesting glide effects, which are obtained by passing the instantaneous frequency variable through a one-pole lowpass filter.

When two alternating tones are played in sequence, various percepts arise depending on their pitch separation, and the repetition rate of the pattern. As has been demonstrated in several auditory experiments, a tone sequence with small pitch separation will tend to be heard as one single stream, whereas if the interval widens, they may instead segregate and form two streams, if the repetition rate is kept constant [6]. As the repetition rate increases, the percept fuses to a single complex tone.

To begin with, let us consider an oscillator with four control parameters, two for the frequencies, F_1 and F_2, and two for their respective durations, $t_1$and t_2. Then the generated sinusoid at sampling frequency f_s is:

$$x_n = \sin(\varphi_n), \tag{1}$$

$$\varphi_n = \varphi_{n-1} + \frac{2\pi}{f_s} f_n \tag{2}$$

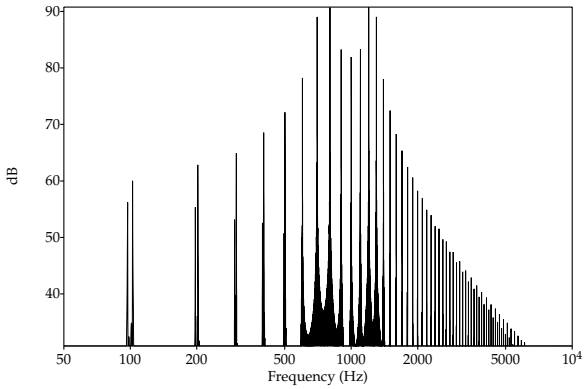

Figure 1: *Spectrum showing two formants, using the parameters* $f_c = 100$, $F_1 = 750$, *and* $F_2 = 1250$ *Hz.*

with instantaneous frequency f_n. To produce a smoother glide, the frequency is filtered,

$$f_n = (1-b)g_n + bf_{n-1}, \quad b \in [0,1) \tag{3}$$

so that *b* controls the smoothness. The tremolo is given by

$$g_n = \begin{cases} F_1 & \text{if } n(\mathrm{mod}T) < t_1 \\ F_2 & \text{if } n(\mathrm{mod}T) \geq t_1 \end{cases} \tag{4}$$

where $T = t_1 + t_2$ is the overall period.

2.1. Audio Rate Tremolo

When the tremolo reaches audio rates, its two tones become perceptually fused. Assuming that there is no frequency smoothing, and further that both F_1 and F_2 are exact multiples of the fundamental frequency $f_o = 1/T$, a pitched sound will be heard at f_o with formants at F_1 and F_2. If the fundamental does not approximately divide both of the formant frequencies, the sound instead becomes inharmonic, but the formants will still be more or less present.

Audio rate tremolo can be understood as FM with a square wave modulator. When thought of this way, it becomes apparent that the carrier frequency is

$$f_c = 1/T, \tag{5}$$

the modulator frequency is a square wave with fundamental at the average frequency,

$$f_m = \frac{F_1 + F_2}{2}, \tag{6}$$

and the deviation

$$\delta = \frac{|F_1 - F_2|}{2} \tag{7}$$

determins the modulation strength. So, in this formulation,

$$x_n = \sin(\omega_c n + \frac{\delta}{f_m} g_n) \tag{8}$$

where $\omega_c = 2\pi f_c / f_s$. Judging from the harmonic content of the geometric square wave g_n with no aliasing suppression, this FM equation would seem to produce intolerable aliasing. There is a simple trick to use in situations where either the amplitude or the frequency needs to be changed suddenly. In the case of amplitude, one would wait until a zero crossing, where the sudden amplitude change will do least harm. Similarly, since a sudden frequency change is associated with a discontinuous first derivative, the frequency can be swapped at positive or negative peaks of the waveform, where its derivative vanishes. In this oscillator played with an audio rate tremolo however, the benefit from switching frequency at signal peaks is just as much reduced by the jitter it causes. It can be used as an effect, potentially leading to longer intervals between frequency changes and a staggered sonic contour if any parameters change dynamically.

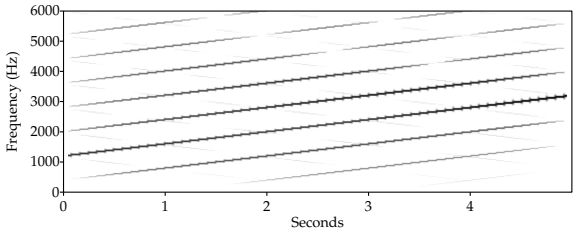

Figure 2: *Tone durations crossfaded during sound, emphasising* F_1 *at the beginning and* F_2 *at the end. Here with* $f_c = 400$ *Hz,* $F_1 = 1200$, $F_2 = 3200$ *Hz.*

As long as the carrier is lower than both F_1 and F_2, there can be formants at these frequencies, as is seen in figure 1. If, however, $f_c > \min(F_1, F_2)$, then less than one whole period of either F_1 or F_2 will be covered until the instantaneous frequency is switched. This will produce a spectrum with a strong component at f_m, and pairs of partials at frequencies

$$k(f_c - \epsilon) \pm f_m, \; k = 1, 2, 3, \ldots, \tag{9}$$

where ϵ is a comparatively small positive number causing a downwards shift of the partials, which has been experimentally observed.

The effect of unequal durations for the two tones can be derived by introducing weighting factors according to their proportions t_i/T. An effect akin to Shepard tones can be heard if *T* is kept constant, but the proportional durations of the two tones are changed, so that $t_1 = wT/2$ and $t_2 = (1-w)T/2$, for $w \in [0,1]$. When sweeping *w* across this range, there is a glissando combined with a refilling of lower partials (figure 2).

2.2. Filtering the control frequency

The effect of lowpass filtering the instantaneous frequency is obvious; it simply makes the transitions smoother. Note that as the instant frequency curve is smoothed, the square wave not only looses strength in high partials, but its amplitude also decreases. Thus, in the limit of a highly smoothed square wave, we have a weakly modulated sinusoid at f_m Hz, which now plays the role of carrier, with modulation at f_c Hz. As noted, the geometric square wave will introduce aliasing problems, which are particularly annoying in audio rate modulation. Here the lowpass filtering of the control frequency comes to rescue; even modest filtering (with small values of *b* in eq. 3) improves the situation.

Allpass filters with audio rate coefficient modulation have recently been introduced as a synthesis or processing method [7]. We propose a different nonstandard usage—passing the instantaneous frequency control function through an allpass filter. Here we

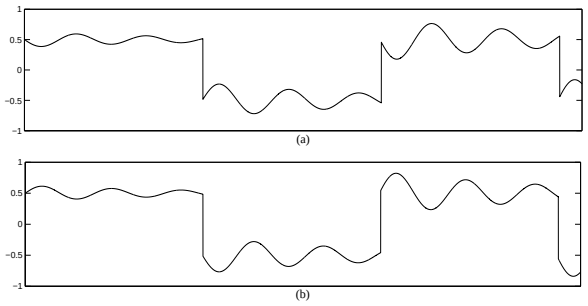

Figure 3: *Square wave with (a) allpass and (b) bandpass filter.*

use the second order allpass filter with tunable parameters. Since a second order allpass filter $A(z)$ can be used for bandpass or bandreject filters [8] with $H(z) = (1 \mp A(z))/2$ (subtract for bandpass, add for bandreject), we borrow the corresponding terms center frequency and bandwidth although they only apply to the phase response of $A(z)$. The interest of submitting the square wave instantaneous frequency to an allpass filter is that it introduces oscillations at the center frequency. It could be tuned to low frequencies for vibrato effects, or audio frequencies for extra modulation. The bandwidth parameter controls a trade-off between large, but fast-decaying oscillations when set to large values (proportional to the center frequency), and smaller amplitude, but more persistent oscillations for narrow bandwidths.

As can be seen in figure 3, the allpass causes the signal to oscillate in the direction opposite to the jump. To produce an oscillation in the same direction, thus giving a smoother profile, one can use the filter $2I(z) - A(z)$, where I is the identity operator. This is a kind of bandpass filter, combined with unit gain at DC, which is the crucial design criterion here. The peak gain of the filter at the resonant frequency is also a useful parameter to control. To do so, just mix the direct signal with more or less of its bandpass filtered version ($\rho \geq 0$ is the amount of bandpass, or $\rho < 0$ for bandreject):

$$H(z) = (1 + \rho)I(z) - \rho A(z), \quad \rho \in [-1, 1] \tag{10}$$

Particularly for low frequency resonances, the filter driven by the square wave will exhibit a start transient before it settles into a stable oscillation pattern. It is also worth pointing out that the visually edgy curve in the top of figure 3 does not necessarily correspond to a perceptually more edgy pitch profile than the curve in the bottom of the figure.

2.3. Comparison with granular synthesis

For slow tremolo rates, the impression of the tremolo may be similar to that obtained by granular synthesis of two alternating tones. In this case, each tone is multiplied with a window, and placed in temporal succession, possibly with some overlap. With granular synthesis, the phase could run independently in two oscillators, which are tapped in turn to the output stream. If adjacent (i.e. non-overlapping) rectangular windows are used, this formulation will introduce phase discontinuities.

If the granular synthesis is carried out with half-overlapping hanning windows, the result is in fact identical to mixing two amplitude modulated sinusoids. Then the signal becomes

```
instr 1 ; TREMOLO OSCILLATOR

   iamp = p4   ; amplitude
   ifc  = p5   ; 1/T, carrier (Hz)
   ifm  = p6   ; (F1+F2)/2, modulation (Hz)
   idev = p7   ; |F1-F2|/2, deviation (Hz)
   ita  = p8   ; t1/T, dur. prop. (0,1)
   iB   = p9   ; filter-coef (0,1)

   kf1    init    0
   kph    phasor  ifc
   iF1    =       ifm - idev
   iF2    =       ifm + idev
   kg     =       (kph < ita) ?  iF1 : iF2
   kfrq   =       (1-iB)*kg + iB*kf1
   kf1    =       kfrq
   aos    oscil   iamp, kfrq, 1  ; sinusoid
   kl     linen   1, 0.01, p3, 0.02
          out     kl*aos
endin
```

Figure 4: *Csound instrument.*

$$x(t) = A(t)\cos(\omega_1 t) + (1 - A(t))\cos(\omega_2 t) \tag{11}$$

$$A(t) = \frac{1}{2}(1 - \cos(\omega_c t)) \tag{12}$$

which produces a spectrum with six components, three partials symmetrically situated around each formant frequency. If the formants ω_1, ω_2 are integer multiples of the fundamental ω_c, the spectrum is harmonic with a clear pitch.

Another related way to achieve just one formant in an harmonic sound is to set the amplitude to zero for the second tone. Now the repetition rate is the fundamental, and the frequency of the single grain determins the formant. Its bandwidth will be related to the time proportion of tone versus gap; the shorter the tone, the wider its bandwidth. By introducing the second tone again, but with possibly asymmetric time proportions and unequal amplitudes, various balances between the two formants can be created—but there is an inevitable trade-off: as the bandwidth of one formant narrows, the bandwidth of the other is bound to increase (provided the two grains do not overlap). Note that this differs from classical formant techniques such as FOF [9], which only introduces one formant per oscillator.

For an extension to more than two formants, a longer tone sequence, possibly with stochastic pitch clouds, could replace the two tone sequence. Clarence Barlow has used such techniques in some of his compositions [10].

2.4. Csound implementation

A stripped-down version of the tremolo oscillator, with lowpass filtering of the control frequency but no allpass filter, is shown here in a Csound implementation (figure 4). Note that it is straightforward to change the init-rate parameters to control-rate for dynamic changes. It may also be useful to rewrite the instrument as an opcode. A phasor runs at f_c Hz, and instead of specifying t_1, t_2, there is a parameter controlling the relative proportion of the first duration.

In the Csound implementation, the control rate may be slower than the sample rate, but it should be kept significantly higher than

f_c. For all of the applications discussed in the rest of the paper, a version written in C++ has been used, where everything runs at the sample rate (except for the example discussed in section 4.2).

3. DATA-DRIVEN APPLICATIONS

From the discussion of the tremolo oscillator it should be clear that its parameters have different functions with respect to sonic results in different regimes. For slow tremolo rates, the two pitches at F_1, F_2 are prominent, but for audio rate tremolo, these two frequencies no longer contribute to pitch, which instead depends on their average, f_m. Even more important is the pattern repetition rate $1/T$, which turns into the pitch f_c. Hence it is far from obvious which versions of these parameters the user should be presented with—at least as long as both these extremes are equally likely to be used. In a graphical user interface, a solution could be to have one slider controlling *T*, and another controlling $f_c = 1/T$, and automatically updating the complementary parameter inveresly to the one that the user adjusts.

Some care has to be taken if note durations may change over time. This is solved by only updating the internal oscillator variables t_1, t_2 at the end of a period. If one tries to update these durations more often than once each period, some of those parameter updates will be inconsequential.

Next we describe experiments using the tremolo oscillator in two cases of data-driven synthesis, with data taken either from sound signals or any arbitrary time series. In the first case, synthesis parameters are sought to match a given input sound. How close a match is possible depends much on the synthesis model in question. The tremolo oscillator is a bit too limited to allow closely matched reconstructions of arbitrary sounds, but controlling its parameters with an input sound remains an interesting control strategy.

3.1. Feature-based synthesis

When using an external sound file as input to the tremolo oscillator, various mappings between its analysed features and the synthesis parameters are conceivable. Although the tremolo oscillator is incapable of rendering all types of sounds, there should be some mappings that produce closer resemblances between the input and the synthesized sound. For artistic purposes a close mimicking of the input may not be essential, but some of its dynamics is likely to be captured. This is similar in spirit to an adaptive audio effect [11], albeit only loosely coupled to the input sound.

The problem of how to obtain close similarities to analysed sounds does not have obvious solutions; we can only offer general suggestions. The simplest part is to match amplitude. An RMS extractor simply specifies what gain to apply. For pitch and spectral matching there is probably no procedure that works as well for any arbitrary sound. The zero crossing rate (ZCR) or a pitch tracker could be used to find an average frequency, corresponding to f_m in eq. 6 above. Then an estimate of the spectral bandwidth centered on f_m could determine the values of F_1 and F_2. Alternatively, a pitch extractor could be mapped to the inverse repetition rate, though, as seen above, if the sound is harmonic, this limits the ratios of the two tones to f_c to integers (or rational numbers). If the idea of imitative synthesis is dropped, any extracted feature may be mapped to any control parameter in an ad hoc fashion.

3.2. Sonification

Other time series data than an audio signal can be used for time-varying control of the synthesis parameters. We have tried a sonification of sunspot numbers, recorded monthly from 1749 to the present [1]. The data consists of the monthly average, and standard deviation. In sonification, a good mapping from data to sound should bring out patterns in the time series, and preferably in such a way that the sound signal easily lends itself to interpretation if the mapping is known. In the sunspot time series, the well-known eleven year cycles are easily found by visual inspection, and should be at least as easy to hear.

There are no missing values in the sunspot series, which contains over 3000 data points. We choose to keep the durations t_1, t_2 equal and constant, with a rate of, say, $T = 0.01$ sec, and map the sunspot data to frequency. If the average activity is mapped to f_m, it appears appropriate to map the standard deviation to the frequency deviation δ of eq. 7. It follows that $F_1 = f_m + \delta$ and $F_2 = f_m - \delta$. For the mapping from the monthly average activity $\bar{s}_n$ we use

$$f_m = f_o \exp(\beta \bar{s}_n) \tag{13}$$

with some suitable constants f_o for lowest possible frequency and $\beta > 0$, in order to scale the range of the time series ($\bar{s}_n$ is approximately in the interval (0, 250), and so is the standard deviation). From this, the maximum frequency deviation should be less than the lowest frequency of f_m, hence $\delta < f_o$.

This mapping does bring out the solar dynamics, especially the inactive periods stand out by having the same pitch (f_o), whereas the height and shape of active periods vary.

The tremolo oscillator could be used for sonification of data of higher dimension as well, for instance introducing a third dimension by varying the ratio of tone durations. Time series with missing values could be handled by muting the oscillator at those points. Unequal sampling intervals in the time series are also easily handled by updating parameter values at corresponding times.

4. SELF-ORGANISING COMPOSITION

Now we turn to the problem of how to achieve extended musical structures with a single tremolo oscillator. For want of better terminology, these structures may be called emergent, and the system that generates them can be called self-organising. The point is, that we usually do not know in advance what kind of dynamics the system will exhibit, whence an experimental approach is called for.

This algorithmic composition system consists of a signal generator (here the tremolo oscillator), a feature extractor, and a mapping component. The feature extractor analyses the output of the oscillator. Then the features are mapped to the tremolo oscillator's synthesis parameters, which are updated. So this is a feedback system of the type shown in figure 5.

The four parameters that control the oscillator is written as a time dependent vector

$$\pi_n = \{F_1, F_2, t_1, t_2\}_n \tag{14}$$

[1] The data are available at http://spaceweather.com/glossary/sunspotnumber.html

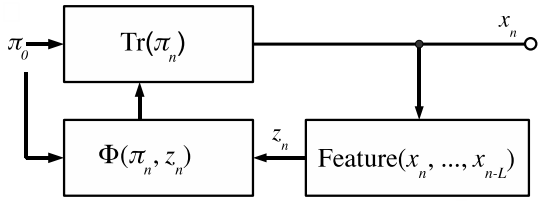

Figure 5: *Tremolo oscillator with feedback from a feature extractor, and a mapping using the previous parameter vector and the extracted feature.*

for the two frequencies and their respective durations at time *n*. The parameter vector is updated by a function of its previous state and an extracted feature, z_n:

$$\pi_{n+1} = \Phi(z_n, \pi_n). \tag{15}$$

Below, the zero crossing rate (ZCR), fundamental frequency and voicing are chosen as features, because they depend on the oscillator's control parameters. The zero crossing rate is analysed over a window of arbitrary length, and its current value is updated for each sample.

It can be useful to have a hierarchy of control rates, from full sample rate, optionally via a slightly slower control rate, to a block rate which is practical for routines such as FFT-based feature extraction and output to sound file. Additional irregular events provide yet another level of the (most likely) slowest control rate. If necessary, overlapped segments could be used for higher temporal resolution for block rate feature extraction.

With this model, a wide range of sonic characters become available. Particularly interesting are those cases where the system produces ever changing streams of pitches and timbres. Under some mappings, the system may become chaotic. A simple test for this is whether two realisations with infinitesimally differing initial conditions (π_0 *vs.* $\pi_0 + \epsilon$) diverge over time. If two such solutions are played simultaneously in stereo, at first they sound identical, then the stereo image widens, and soon after they become uncorrelated. Chaotic solutions in this tremolo oscillator system are characterised by certain persistent patterns, forming recurrent but varying musical motives.

It should be noted that this kind of system is different from the most familiar chaotic systems and common uses of them in musical composition [12]. Neither is it a flow (the solution of an ordinary differential equation), nor is it quite similar to any of the commonly cited low dimensional maps, such as the logistic map, circle map, etc. Even such oscillator-like systems as the circle map [13] or the standard map are difficult to use for direct sound synthesis, mainly because the phase space is full of sharply delimited regions with strongly contrasting behaviour. The originality of a system such as that shown in figure 5 lies in the interwoven combination of timescales—the sample rate and a slower note level rate, corresponding to the period length *T*. Note that if, in eq. 15, the term z_n is dropped, it reduces to a four-dimensional map. In fact, the feature extractor increases the map's dependence on past states, thus increasing the system's dimension. But it also acts as a smoothing filter, with the practical consequence that chaos tends to be suppressed.

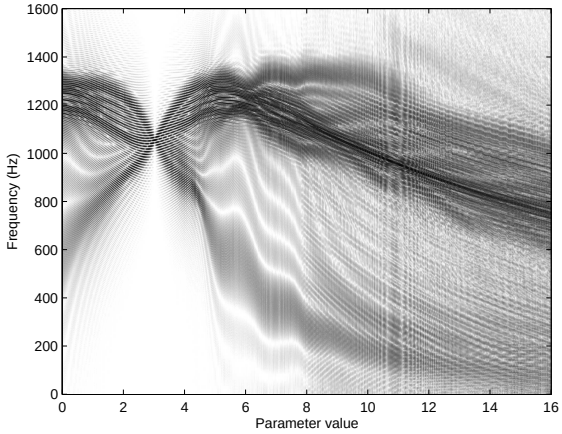

Figure 6: *Spectral bifurcation plot of the map (eq. 16, 17) as a function of the parameter* p.

4.1. Smoothed map-like system

Depending on the particular mapping, this type of systems can exhibit rather different behaviour. Consider as an example that the parameter vector is updated for each sample, and depends only on an extracted feature, the ZCR as analysed over a 40 ms window. There is no explicit dependence on the previous parameter state, although it will of course be reflected in the oscillator's output. For the mapping, we use

$$F_{k,n+1} = A + B\cos(2\pi p z_n - \phi_k), \quad k = 1, 2 \tag{16}$$

with $A = 600,\ B = 550$ Hz for the frequencies, and

$$t_{k,n+1} = C - D\sin(2\pi p z_n + \phi_k), \quad k = 1, 2 \tag{17}$$

where $C = 0.150,\ D = 0.145$ seconds for the tone durations, and $\phi_1 = 0$, $\phi_2 = \pi/2$ for phase offsets. The free parameter *p* controls at once the amount of feedback and the degree of nonlinearity. All feedback happens through z_n, which is the zero crossing rate of the current output of the tremolo oscillator.

Traditionally, the parameter dependencies of chaotic systems is plotted in bifurcation diagrams. An alternative to this, suitable for acoustic systems, is the spectral bifurcation diagram [14]. It is similar to a spectrogram, but instead of time on the horizontal axis it shows the parameter value. Each spectral slice is plotted after the system's transients have died out. A spectral bifurcation diagram showing the amplitude spectrum at each value of *p* displays some qualitatively different behaviours; note for instance the crossing where the two tones have the same frequency near $p = 3$ (figure 6).

Here the parameter *p* controls how folded or nonlinear the map is. A similar, though much simpler system was introduced in an earlier paper [15]. In that case, the oscillator generated a single sinusoid with variable frequency, controlled by a nonlinear mapping of its ZCR from the immediate past. A common trait in these systems is that the nonlinearity needs to be comparatively strong before any interesting behaviour can occur. As pointed out, the window length of the feature extractor acts as a smoothing filter,

which effectively quenches any chaotic tendencies unless the nonlinearity is sufficiently strong.

This smoothing effect implies that for the same mapping, the dynamics will be very different depending on the window length of the feature extractor. In the absence of a feature extractor in the loop, the system is likely to produce gritty or noisy sounds already at low degrees of nonlinearity. This happens if the equations 16–17 are modified so that they rely on previous parameter values but not on the extracted feature; then the system produces pitched but noisy sounds over most of the parameter range, only interrupted by short windows of purer tones.

However, a flow-like map results if we use the Euler method of solving an ordinary differential equation:

$$\pi_{n+1} = \pi_n + h\Phi(z_n, \pi_n). \tag{18}$$

For the frequency variables, the flow-like map (18) is a good opportunity to add some extra glissandi.

4.2. Block-rate map

Finally we consider a system where the parameter update takes place at a block rate, typically with a block size *N* of 2^9 – 2^{13} samples. This is a practical consequence of using an FFT-based feature extractor, which operates on sample buffers of such sizes. If the feature extractor block length is not equal to or an integer multiple of the total period *T*, this will cause an interference between these two periods, itself a source of complexity. In the example given below, $t_1 = t_2 = 0.05$ and $N = 1024$ samples for the FFT window length. With a sample rate of 48 kHz, *T* is 4800 samples, so there will be about 2.3 FFT frames for each tone of the oscillator, causing a constant drift; sometimes just one tone is covered, sometimes a bit of both.

If $x_n = Tr(\pi_n)$ is the output of the tremolo oscillator with given parameters, the parameter update now takes place at a much slower rate than the sample rate, so the oscillator gets to generate samples undisturbed by frequent parameter changes. Let z_m be the feature extracted at times $m = 0, 1, \ldots$ where the block sample rate is f_s/N so that $m = \text{INT}(n/N)$ where INT denotes the integer part. Then the parameter update can still be written as in eq. 15, with the main difference being the slower sample rate.

There can be many interacting components in this type of systems, making a thorough analysis quite complicated. Meanwhile, it is precisely when the system becomes sufficiently complex, that it begins to be capable of producing really interesting textures. In order to keep the analysis tractable, we restrict the map to only the frequency variables, and keep durations fixed. The tremolo oscillator does use the lowpass filter, but not the allpass filter in this example. Consider the map

$$F_{k,n+1} = A + K(1 - 2\hat{f}/f_s)\cos(2\pi\hat{v}w - \phi_k), \tag{19}$$

again with $\phi_1 = 0, \phi_2 = \pi/2$, and $A = 700$, $K = 650$, and where $w \geq 0$ is a free parameter. Autocorrelation is used for feature extraction, where $\hat{f}$ is the estimated fundamental frequency, and $\hat{v} \in [0, 1]$ is the voicing [11], i.e. the normalised amplitude of the first peak of the autocorrelation function at lag corresponding to $1/\hat{f}$.

In figure 7, the prediction time of this system (eq. 19) is shown. The prediction time is approximately related to the inverse of the greatest Lyapunov exponent; it is the time before two orbits with slightly different initial conditions diverge and their distance becomes comparable to that of the attractor [16]. Here the initial values of the two frequencies are slightly perturbed (by $\epsilon = 10^{-6}$); the difference of the two trajectories thus generated is measured at the output of the oscillator. The prediction time is taken to be the time until the magnitude of the difference of the two signals becomes comparable to the signal's amplitude. For $w < 0.3$, the prediction time appears to be unlimited; in other words, the system is stable for low parameter values. Whenever the prediction time takes a finite value, this is an indication of chaos at that parameter value. The effect of using the frequency smoothing filter (3) can be seen: where there are peaks in prediction time for the unsmoothed system, the filtered version has shorter prediction times. In other words, the filter may induce or increase chaos. At $w = 1.17$ there is a peak in the prediction time; actually it is finite but reaches as much as 23 seconds. The plotted prediction times in figure 7 depend on the particular initial values chosen and the size of the perturbation ϵ.

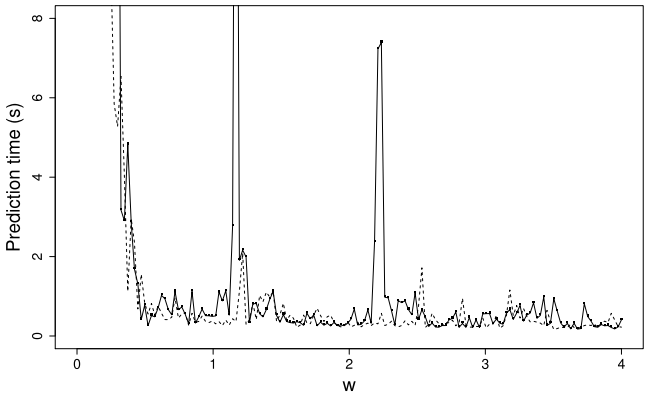

Figure 7: *Prediction time as a function of the parameter* w*, using eq. 19. Solid line: without filter, dashed: with lowpass filter and* $b = 0.99$.

Further variations of these systems would be interesting to explore: the block rate and sample rate maps may be combined, different mappings can be tried out, other feature extractors can be used. And of course, other signal generators may be inserted in place of the tremolo oscillator. There are lots of possibilities for variation, and little is yet known about this type of systems.

5. CONCLUSIONS

A tremolo is qualitatively very different from audio rate square wave FM, but in the tremolo oscillator, these are just the extremes of a single parameter continuum. This simple model allows for a surprisingly large range of sounds, which can be exploited by direct control, sonification, or in a self-organising algorithmic composition system.

The tremolo oscillator and all applications of it has been implemented in C++. For practical reasons, the program operates offline and outputs to soundfiles only, although the instrument is efficient enough to run in real time without any problems. The main ideas are simple, as can be seen in the Csound implementation (section 2.4).

In the self-organising system, the feedback is taken care of internally in the computer, with no acoustic intermediary signal. One may question the need for feature extraction in this case. After

all, the actual synthesis parameters are available for inspection. In spite of that, we propose to use very simple feature extraction as a means to simplify the control of a tremolo oscillator, seen as a self-organising algorithmic composistion system.

Throughout this paper the tremolo oscillator has been discussed, first on its own, then in various applications where it receives input from external sources. The feedback type of system could very well be extended so that it also takes an input source, be it from a gestural controller or an audio signal. Further work will be directed at other self-organising and chaotic systems, similar to the ones presented in the last section. But it should be noted that any signal generator may substitute for the tremolo oscillator, just as other mappings or feature extractors may be used. Some of the fascination with the tremolo oscillator lies in the fact that it is a *proto-sequencer*, with a ridiculously short memory restricted to two notes. However, this seems sufficient for the system to bridge the gap from merely a generator of timbres and textures to more complex patterns.

6. ACKNOWLEDGEMENTS

The author would like to thank Rolf Inge Godøy for his many pertinent suggestions and Sverre Holm for comments on an early draft of this paper.

7. REFERENCES

[1] M. Hoffman and P. Cook, "Feature-based synthesis for sonification and psychoacoustic research," in *Proceedings of the 12th International Conference on Auditory Display*, London, UK, June 2006.

[2] A. Di Scipio, "'Sound is the interface': from interactive to ecosystemic signal processing," *Organised Sound*, vol. 8 (3), pp. 269–277, 2003.

[3] S. Bökesoy, "Synthesis of a macro sound structure within a self-organizing system," in *Proc. of the Int. Conf. on Digital Audio Effects (DAfX-07)*, Bordeaux, France, 2007.

[4] I. Xenakis, *Formalized Music. Thought and Mathematics in Music*, Pendragon Press, Stuyvesant, revised edition, 1992.

[5] A. Chandra, "Creating von Foerster's non-trivial machine in digital audio," in *Proc. of the ICMC*, Copenhagen, Denmark, 2007, vol. 1, pp. 173–178.

[6] A. S. Bregman, *Auditory Scene Analysis*, The MIT Press, 1990.

[7] J. Kleimola, J. Pekonen, H. Penttinen, V. Välimäki, and J. Abel, "Sound synthesis using an allpass filter chain with audio-rate coefficient modulation," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, 2009, pp. 305–312.

[8] U. Dutilleux, P. Zölzer, "Filters," in *DAFX. Digital Audio Effects*, U Zölzer, Ed., chapter 2, pp. 31–62. Wiley, 2002.

[9] X. Rodet, "Time-domain formant-wave-function synthesis," *Computer Music Journal*, vol. 8, no. 3, pp. 9–14, 1984.

[10] C. Barlow, "ISIS, an alternative approach to sound waves," in *Proceedings of the 2005 International Computer Music Conference*, Barcelona, Spain, 2005, pp. 660–663.

[11] V Verfaille, *Effets audionumériques adaptatifs : théorie, mise en œuvre et usage en création musicale numérique*, Ph.D. thesis, Université Aix-Marseille II, 2003.

[12] R. Bidlack, "Chaotic systems as simple (but complex) compositional algorithms," *Computer Music Journal*, vol. 16, no. 3, pp. 33–47, Fall 1992.

[13] G Essl, "Circle maps as simple oscillators for complex behavior: II. Experiments," in *Proc. of the 9th Int. Conf. on Digital Audio Effects (DAfX-06)*, Montreal, 2006, pp. 193–198.

[14] W. Lauterborn and E. Cramer, "Subharmonic route to chaos observed in acoustics," *Physical Review Letters*, vol. 47, no. 20, pp. 1445–1448, November 1981.

[15] R. Holopainen, "Feature extraction for self-adaptive synthesis," *Sonic Ideas/Ideas Sónicas*, vol. 1, no. 2, pp. 21–28, 2009.

[16] T. Tél and M. Gruiz, *Chaotic Dynamics. An Introduction Based on Classical Mechanics*, Cambridge University Press, 2006.