ERRANT SOUND SYNTHESIS

Nick Collins
Department of Informatics, University of Sussex
N.Collins@sussex.ac.uk

ABSTRACT

The SLUGens pack for SuperCollider 3 is a collection of novel algorithmic sound synthesis and processing UGens. This paper describes various non-linear dynamic oscillators, buffer data processors, breakpoint set synthesisers, and miscellaneous filters. Non-standard sound synthesis has been neglected in modern research programmes concentrated on spectral and physical modelling synthesis. Its goal is not the modelling and reproduction of sounds from perceptual or physical acoustical data but the potential of any algorithm, cast into the audio range. Yet where the composer seeks novel sound generation and control possibilities, much interesting territory remains available to explore in the context of what might be termed *errant sound synthesis*.

1. INTRODUCTION

Much sound synthesis is concerned with recreating existing instruments or acoustic events, though potentially any sound which can come out of a loudspeaker can be created. Without an acoustic model, this often leads to raw noisy sounds, yet these are still part of exploring new points in 'the uncharted galaxy of synthetic sound' [10, p345]. This paper reflects this pursuit, following previous experiments in breakpoint interpolation, wavetable and dynamic oscillator synthesis and variants categorised by Curtis Roads as 'non-standard'. Indirect control via the parameters of equations can be a liberating experience; it may enable the alternative expression of inspiring spectral transitions, of a type unencountered at the normal timescale and physics of airbourne acoustics. Non-standard synthesis methods should be given a fair chance to assert themselves again as independent abstract sound sources. To use Julius O. Smith's fourfold categorisation of synthesis algorithms [11], this work might fall under *processed recordings* as manipulating pre-existing buffers of sound data, or *abstract algorithms*. Some of the oscillators also show the influence of nonlinear mathematics relating to models of real physical systems; though as these are not originally sounding structures, they are sonifications.

My personal motivation stems from my earliest experiments with SuperCollider 3 plug-ins, converting various forms of Xenakis' General Dynamic Stochastic Sound Synthesis [14, 6] into the *Gendy* UGens. I have subsequently explored the sound synthesis potential of a number of alternative and even quirky procedures, and have released those with potential as open source software. The structure of this paper follows that of a catalogue of experimental synthesis methods, to accompany the publicly available SLUGens plugin pack for SuperCollider 3, which should be explored in combination with the text (`http://www.informatics.sussex.ac.uk/users/nc81/code.php`). All of the UGens described in this paper run in real-time; most run at a low average CPU cost (for example, on a 1.67GHz Mac PowerBook G4 PPC, FitzHughNagumo: 1.5%, DoubleWell3: 2%) though some are dependent on parameter choices (GravityGrid2 is variable with the number of masses: e.g., 4.5% (1 mass) 33% (10) 50% (20)). Most also have multiple versions enabling variations of core behaviour – DoubleWell$n$ where n=1 to 3, for example – these correspond to the UGens provided in the distribution, and each has a help file.

2. TECHNICALITIES

Non-standard synthesis routines are not usually tractable analytically to predict their time varying spectral properties; rather, they are empirically explored.

Certain guards have to be in place to cope with blow-ups in output values. Wrapping or folding (topologically, computer game wraparound or bounce back from a hard surface) were often employed to keep a parameter within the range [0,1] or [-1,1], though a sensible standard working range was empirically determined for normal operation to avoid the imposition of extra discontinuities through such measures.

Sometimes the outputs are highly aliased – these are not band-limited oscillators [12]. The UGens are designed such that standard outputs are at a sensible audio rate without extreme slew; but the nature of chaotic dynamics and the potential of noise must allow for wilder behaviour.

A number of the SLUGens have a reset argument which restarts calculation at a random, or parameterised, position. For the case of dynamical equations this is equivalent to rerunning the simulation from new initial conditions. Resetting also allows the imposition of further periodic behaviour on a system, in the form of hard (immediate) or soft (only if near a zero crossing or at a period boundary) sync. Yet because of the complex dynamics determining waveforms, alias free synthesis of the hard sync [1] would not seem practicable.

3. NON-LINEAR OSCILLATORS

Beyond direct acoustic modelling, there remains a rich world of dynamical equations which have interesting periodic or pseudo-periodic (often chaotic) solutions. Through appropriate parameter choices these can be run with periodicities at perceptually discriminable rates.

In dealing with dynamical equations, the discretisation necessitated by computer calculation requires numerical approximation of solutions given starting conditions, by the use of discrete ODE (Ordinary Differential Equation) solvers. A number of schemes can be employed [13, pp. 32-4], the best known and most intuitive being the Euler approximation; another popular choice is the more highly interpolated fourth order Runge-Kutta scheme (and further options exist). It is possible to use a higher sampling rate than the audio rate for ODE solver calculations, but in practice, one audio sample at a time is the natural rate of calculation.

Where available as input arguments to the UGen, parameters of the dynamical system are often modifiable during calculation at k-rate (control rate). Indeed, in one case, the actual external force term is itself an arbitrary UGen. Any output frequency is hostage to the dynamics of the given system, in combination with the differential equation solver's update rate. But sufficient external periodic force or forced sync from resetting can impose a fundamental frequency if required. In some cases, resets can reduce wild behaviour from divergent trajectories for the ODE solver (under the numerical constraints).

The virtues of these nonlinear oscillators is often in their more pseudo-periodic and noisy outputs, particularly in chaotic regimes, where interactive control can exploit the close boundaries between erratic behaviour and (temporary) stability.

3.1. FitzHughNagumo

Neuronal models from computational biology and biophysics [5] are a rich source of behaviours, and may prove topical and popular in artistic aesthetics allied to the advance of the neuroscience of music and audition. For instance, Alice Eldridge [4] looks at the entrainment behaviour of a pair of simulated neurons, mapping the continuous output to discrete rhythms and pitched events. A two-dimensional simplification [1] of the biophysics of an individual neuron are the FitzHugh Nagumo equations, which model the behaviour of a neuron recharging and retriggering. A naive Euler ODE solver implementation of the FitzHugh-Nagumo neuronal model for oscillatory firing can provide an audio oscillator.

$$\frac{\delta u}{\delta t} = rateu * (u - 0.3333 * u^3 - w) \tag{1}$$

$$\frac{\delta w}{\delta t} = ratew * (b0 + b1 * u - w) \tag{2}$$

From these interlinked first order differential equations u is output, though w also gives interesting aural dynamics. The SLUGens also include a second experimental neural oscillator model, TermenWang [2, p355]; both FitzHughNagumo and TermenWang are highly sensitive to parameter choices and update rates.

3.2. DoubleWell

This UGen is a Runge-Kutta ODE solver implementation of the chaotic Forced Double Well Oscillator [13, pp. 441-7]. The equation governing the dynamics represents a two well potential, where the particle can settle into one of the two minima; but an additional forcing term applies to the whole system which can destabilise the particle from either well. There can be two forms of oscillation, locally around one of the minima, or globally between the minima.

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + \delta\frac{\mathrm{d}x}{\mathrm{d}t} - x + x^3 = F\cos wt \tag{3}$$

The periodic forcing term is on the right – in the DoubleWell3 UGen this is replaced by an arbitrary UGen input.

Figure 1 shows the immediate oscillatory output of the system over the first 200 samples with parameter values $F = 0.4$, $w = 1$, δ =0.25 (following [13, Example 12.5.2, pp. 444-5]). The system exhibits a chaotic transient (jumping erratically between the two potential wells) before settling into a pseudo-periodic orbit between them (actually a strange attractor in longer term behaviour); with a smaller forcing term (such as $F = 0.2$),

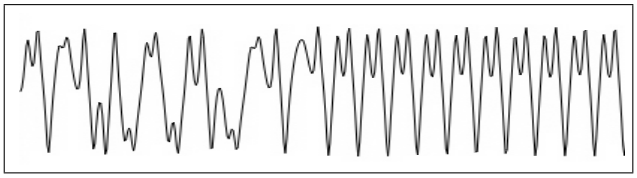

Figure 1. DoubleWell oscillator with $F = 0.4$, $w = 1$, δ =0.25.

it settles into oscillation within one well only after a chaotic transient, though which well is chosen is extremely sensitive to the initial conditions. In terms of audio, this can lead to a reduced amplitude DC offset oscillation; whilst the leak can be cured by a LeakDC UGen and the amplitude by compression, multi-well dynamics of greater amplitude are promoted by oscillator resetting or increased external force.

3.3. WeaklyNonlinear

This UGen instantiates a naive Euler ODE solver implementation of the Weakly Nonlinear Oscillator with external perturbation ([13, p.215] [9, p.189]). The equation:

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - \alpha(x^\gamma + \beta)\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^\delta + {w_0}^2 x = input \tag{4}$$

has a number of special cases, including the Duffing equation ($\delta = 0$, $\gamma = 3$, $\beta = 0$, $input = 0$) and van der Pol oscillator ($\delta = 1$, $\gamma = 2$, $\beta = -1$, $input = 0$). The nonlinear term is not calculated if α is zero. Otherwise it is generated at additional CPU cost. I have added the right-hand term as an additional external forcing input; when non-zero this makes the system increasingly chaotic since time dependence adds an extra dimension to the phase space.

This equation has proved to be a rich source of dynamic behaviours, and all parameters can be controlled by k-rate UGens. The equation constant w_0 is the radian frequency of a linear oscillator in the absence of the nonlinear term – for small α this can remain an approximate target frequency desired for oscillation even with nonlinear perturbations. As an input to the oscillator it is expressed by the user as a frequency in Hertz, then converted to angular frequency internally.

3.4. GravityGrid

Newtonian equations for the force of gravity between a number of particles quickly give rise to interesting dynamics. For the GravityGrid UGen eight outer points equidistant on the perimeter of a square accelerate under gravity an inner moving particle, whose distance from the centre gives the amplitude of the output waveform (Figure 2). Position folding is used at the boundary to avoid the moving particle escaping the square.

In the GravityGrid2 UGen an arbitrary number of influencing masses can be created, at (x,y) positions and with mass values given by a buffer of data passed to the UGen. This data can be dynamically updated to change the playing surface. The moving particle is restricted to the square (even if the fixed masses are outside the square) and its velocity is itself folded to avoid runaway momentum. The CPU cost is dependent on the number of masses; these UGens can be expensive to run.

The original GravityGrid UGen in particular contained a number of idiosyncratic mathematical errors [2] in its implementation,

[1] Suppressing the effect of separate chemical ion flows.

[2] An error in the folding equation due to the behaviour of the ANSI

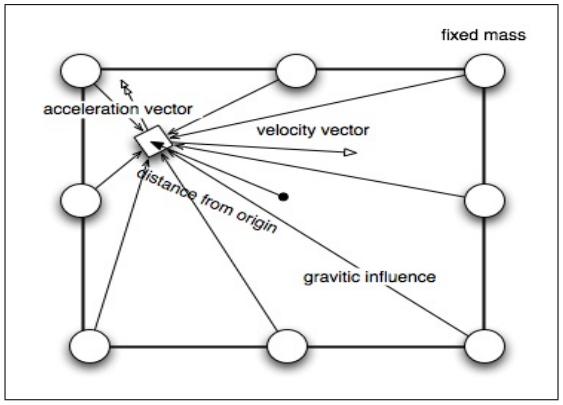

Figure 2. GravityGrid oscillator showing eight fixed perimeter masses and the moving mass; distance from the origin at the centre is the amplitude output.

which, however, give it a rich soundscape. GravityGrid2 is an attempt to rationalise this, introducing in particular the speed limits.

Varied dynamics are possible depending on the selection of positions and masses, from smooth quasi-stable oscillations to noise. More complex dynamics can occur by retriggering or changes in the controlling masses.

4. BREAKPOINT SETS

Breakpoint sets have long been an interest of mine - some of my earliest computer music research was in devising a synthesis engine based on spline interpolation of breakpoints, exploring schemes for the interpolation of breakpoint sets. Breakpoints are an indirect route to wavetables, allowing a smaller number of parameters (one x and y for each breakpoint) in specifying the wavetable function shape. The spectrum is only analytically predictable from breakpoint positions for the case of equally spaced abscissae [8].

4.1. KmeansToBPSet1

Successive iterations of a (soft) k-means clustering algorithm [7] on data in a 2-D space ([0,1] by [0,1]) are used in this UGen to form the basis of successive periods of a waveform for synthesis. Each mean location is taken as a (time, amplitude) breakpoint, though the number of data points contributing to clustering can be much larger; the size of the background data set is an initialisation parameter for the synthesis but the number of means can be dynamically changed within a bound.

For each cycle of the waveform, the soft k-means algorithm is used to update the clustering by one step (the amount, the *softness*, is itself a parameter of synthesis). To determine oscillator output, after sorting the means into time order (and the imposition of points at (0,0) and (1,0) for continuity), the means are taken as breakpoint sets (scaled over the y range [-1,1]). The BPs are used via linear interpolation to construct output synthesis based on the oscillator phase.

Re-triggering can be engaged either by forcing a new data points set or new starting means. Means and data points are drawn at random or according to positions passed in by an additional buffer. Hard sync is avoided by storing an update flag - the new data or means are imposed at the beginning of the next cycle before the soft k-means update calculation. [3]

C fmod function for negative numbers, a repulsive rather than attractive gravity, a discontinuity with x position in the sign taken for the output amplitude, and a rough trigonometric approximation for resolving components of the acceleration vector in the two directions.

Num	Instruction
0	Interpolate from last to new breakpoint over PARAM *5000 samples (i.e. down to about 10 Hz, sampling rate dependent)
1	New random breakpoint with amplitude from -PARAM to PARAM
2	New breakpoint by perturbing last breakpoint amplitude by PARAM
3	New breakpoint by interpolating (t=PARAM) from last breakpoint amplitude to its inversion
4	New breakpoint by interpolating (t=PARAM) last two breakpoints
5	New breakpoint by damping last breakpoint amplitude (multiply by PARAM)
6	New breakpoint at amplitude PARAM
8	Do next command if probability PARAM
9	Goto instruction PARAM in the program listing

Table 1. Table of virtual machine instructions used by the Instruction UGen.

4.2. Instruction

The Instruction UGen explores some possibilities in live coded synthesis by creating and interpolating successive breakpoints [4] according to certain instructions. A buffer holds the current program of virtual instructions, which can be swapped at any moment. The synthesis is fundamentally breakpoint synthesis with linear interpolation. This virtual machine UGen is a live codable version of 'instruction synthesis' [10] as explored by Paul Berg, Gottfried Koenig and others most intensely in the 1970s.

The program buffer holds instructions on the synthesis server in the form of successive COMMAND/PARAM number pairs. The limited command set is given in Table 1.

In most cases, sensible values for PARAM are from 0.0 to 1.0; but instruction 9 uses a direct index into the instruction buffer. To give an example of combining instructions, a zero amplitude breakpoint could be created by instruction (3,1) then (4,0.5). The virtual machine could crash, for instance if only instruction 8 was used in the buffer (essentially in fact, if 0 is never called to advance time). A safeguard in the UGen means that if more instructions are run than samples in a calculation block, a 0 instruction is forced to cover all remaining samples in the block.

A wide range of different noise sounds can be explored via this algorithm; the simplicity of the virtual machine makes it very easy to algorithmically generate new programs, and it is possible to algorithmically swap the buffer, delineating timbral events.

[3] The electronica artist Cylob (Chris Jeffs) featured this UGen on his blog via a demo track built using his Cylob Music System sequencer to control the oscillator (http://cylob.blogspot.com/2007/03/kmeanstobpset1.html)

[4] The idea of calculating just from one breakpoint to the next at a time to generate the waveform is also implicit in my treatments of Xenakis' GENDYN algorithm.

4.3. Wave Terrain and VMScan2D

The idea of a sound synthesis virtual machine has also been adopted to create a scanning (read pointer controlling) UGen, VMScan2D, whose paths are determined by the 'program'. The x and y outputs can act as controls to an independent WaveTerrain UGen; the terrain is read by linear interpolation of (x, y) position explicitly on a two dimensional buffer describing the amplitude surface function $z(x, y)$.

5. MISCELLANEOUS SOUNDSTREAM AND BUFFER PROCESSING UGENS

5.1. SortBuf

Inspired by Alex McLean's (unpublished) work on the use of iterations of a sort algorithm for the generation of permutations of event material in algorithmic composition, this UGen successively sorts the contents of a buffer using the naive O(N^2) bubblesort algorithm. The speed of sorting (number of sorts each cycle of the buffer) is a parameter of the processing - faster sorts can be CPU heavy; the algorithm is amortised and the state of the sort is stored between sample frame calculations and blocks.

The sorting process causes a gradual distortion, as the target buffer gets sorted into increasing sample values over time, a destructive operation on the buffer. Since new buffer contents can be set at any time, a form of SuperCollider language controlled hard sync is possible by sending successive copy messages to the buffer, restoring its contents from a default dataset.

5.2. Breakcore

This UGen was written in 50 minutes (C code, help file and SuperCollider class) in front of a live audience at the TOPLAP jam at transmediale 2005. It allows repetitions of recent contents of a delay line like the machine gun fire stutters of the *breakcore* style. It works, but is in a somewhat scruffy state of presentation due to its live origins. It is maintained in the distribution as an artistic proof of concept.

5.3. LPCError

In a twist on Linear Predictive Coding Analysis [10, 3], this UGen outputs the short-term 64 sample (block-size) linear prediction of the signal. The only parameter is p, the number of filter coefficients (1-64) with which to model the signal within blocks. In practice, the UGen operates as a sort of lossy analysis-resynthesis distortion.

5.4. LTI and NL

LTI is a UGen which allows the specification of an arbitrary linear time-invariant filter. The feedforward and feedback coefficients are passed in via respective buffers, thus representing the general LTI filter difference equation in the time domain. This is not a pole/zero view, so time domain coefficients must be calculated independently to work from the z-plane backwards. A corollary is that stability is not guaranteed, though this is also part of the fun. Independent SuperCollider code is supplied to try out possible filter coefficients, which generates an impulse response and frequency and phase response plots.

NL and NL2 generalise this to arbitrary non-linear filters in powers of x (previous input) and y (previous output), without and with cross-terms respectively. Blow-up detection is built in, and current and buffered previous outputs are reset to zero if absolute output or rate of change of output exceeds certain user-specified guard values.

6. ERRANT

This paper has presented some strange and alternative ideas for abstract sound synthesis procedures.

An adventurer never settles down but is always seeking more adventures. Some future ideas would be:

- To use arbitrary potential wells for particles. [5]
- To employ all sorts of sorting algorithms.
- In the WeaklyNonlinear UGen, to allow arbitrary polynomial sums of x and $\frac{\mathrm{d}x}{\mathrm{d}t}$.
- Coupled networks of non linear oscillators.

This empirical style of sound synthesis might be dubbed *errant synthesis*; in the old sense of seeking out adventure, and the newer sense of deviating from a standard.

7. REFERENCES

[1] E. Brandt. Hard sync without aliasing. In *Proc. Int. Computer Music Conference*, 2001.

[2] G. J. Brown and D. Wang. Neural and perceptual modeling. In D. Wang and G. J. Brown, editors, *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. John Wiley and Sons/IEEE Press, Hoboken, NJ, 2006.

[3] P. R. Cook. *Real Sound Synthesis for Interactive Applications*. AK Peters, Wellesley, MA, 2002.

[4] A. Eldridge. *Collaborating with the behaving machine: simple adaptive dynamical systems for generative and interactive music*. PhD thesis, University of Sussex, 2007.

[5] W. Gerstner and W. Kistler. *Spiking Neuron Models*. Cambridge University Press, Cambridge, 2002.

[6] P. Hoffmann. The new GENDYN program. *Computer Music Journal*, 24(2):31–38, 2000.

[7] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, 2003.

[8] Y. Mitsuhashi. Piecewise interpolation techniques for audio signal synthesis. *J. Audio Eng. Soc.*, 30(4), 1982.

[9] A. Pikovsky, M. Rosenblum, and J. Kurths. *Synchronization: A Universal Concept in Nonlinear Sciences*. Cambridge University Press, Cambridge, 2001.

[10] C. Roads. *The Computer Music Tutorial*. MIT Press, Cambs, MA, 1996.

[11] J. O. Smith. Viewpoints on the history of digital synthesis. In *Proc. Int. Computer Music Conference*, 1991.

[12] T. Stilson and J. Smith. Alias-free digital synthesis of classic analog waveforms. In *Proc. Int. Computer Music Conference*, 1996.

[13] S. H. Strogatz. *Nonlinear Dynamics and Chaos*. Addison-Wesley, Reading, MA, 1994.

[14] I. Xenakis. *Formalized Music*. Pendragon Press, Stuyvesant, NY, 1992.

[5] Reminiscent perhaps of Bob Sturm's physics sonification experiments.