Musical pattern generation with variable-coupled iterated map networks

BRET BATTEY

Center for Digital Arts and Experimental Media, University of Washington, Seattle, WA 98195, USA
E-mail: bbattey@u.washington.edu
URL: http://www.BatHatMedia.com/

This paper introduces the concept of variable-coupled iterated map networks and explores its application to generation of musical textures. Such networks consist of one or more interlinked nodes. Each node consists of an iterated map function with a time-delay factor that schedules successive iterations. The value broadcast by a node can drive the variables and time-delay factor of any other nodes in the network, including itself. Lehmer's Linear Congruence Formula, an iterated map normally used for production of pseudo-random numbers, is explored for its own potential as a pattern generator and is used as the iterated map in the nodes in the examples presented. The capacity of these networks to produce richly gestural behaviours and mid-term modulation of behaviour is demonstrated.

I. INTRODUCTION

Iterated maps, also known as nested functions, are functions in which a given map is repeatedly applied to an object. When each iteration of the map expresses an advance of the state of a system in time, such functions are called finite-difference equations, expressed generically in the form $X_{t+1} = f(X_t)$. X_0 is the initial condition, often called the *seed*. Such equations can exhibit some or all of a wide range of decay, steady-state, growth, periodic, and chaotic behaviours and fractal patterns (Kaplan and Glass 1995; Wolfram 2002). Compositional applications of specific functions of this type have been explored (for example in Gogins 1991; Leach and Fitch 1995).

A single iterated map attains its rich palette of behaviours through the mechanism of feedback, i.e. the previous state is the basis for the next state. However, natural systems often consist of numerous layers of nested feedback mechanisms. Acknowledging this, one is led to the idea of networking iterative maps serially and with feedback and meta-feedback linkages. Each iterative map will then be a semi-autonomous process, generating a sequence of states while potentially influencing and being influenced by other iterative maps in the network. In this case, the behaviour of each node will be related to the behaviour of the other nodes by the implicit logic of the system. It seems possible, then, that the behaviour of a networked node will be of greater interest than that of a node standing alone, but also that the relationships between the nodes – a multi-dimensional data flow arising from the whole system – could prove perceptually interesting when mapped to sonic materials. Given the examples we see in nature, we have reason to suspect that richly patterned behaviours can emerge from such systems. We can even hope that such systems could be configured to exhibit an alteration of activity plateaus, periods of dynamic change, and recurring self-similar elaborations. In other words, such networks may demonstrate higher-order change. This could be musically useful given that higher-order change – variation in the nature of the variations, if you will – is a crucial aspect of traditional musical form and gesture.

The behaviour of coupled iterative maps has been formally explored to some degree. The one-dimensional systems described in (Palacios 2001) and the coupled map lattice approach (Kaneko 1993) use some form of coupling function to alter the state of nodes based on the state of surrounding nodes. Thus they are related to but distinct from the variable-coupled networks described here, in which the state of a node can alter the variables in another node but does not directly alter that node's state. The idea of networked iterative maps also has conceptual points of contact with areas of inquiry such as neural nets, mutually inhibited artificial neuron systems (Laine 1997, 1999), and boolean networks (Kaplan and Glass 1995).

2. LEHMER'S LINEAR CONGRUENCE FORMULA

Though a wide range of iterated maps are available for musical exploration, the following investigation utilises Lehmer's Linear Congruence Formula (LLCF). The formula can be expressed as $x_{t+1} = (x_t a + b)$ modulo m. LLCF is widely utilised as a pseudo-random sequence generator. By setting m to the word size of the computer and carefully choosing optimal values for a and b, LLCF will generate an effective pseudo-random sequence (Ames 1992). However, by choosing values of a and b that are not

Organised Sound **9**(2): 137–150 © 2004 Cambridge University Press. Printed in the United Kingdom. DOI: 10.1017/S1355771804000226

optimal for randomisation, LLCF is useful as a pattern generator. With this approach, LLCF is capable of exhibiting a wide range of behaviours, from steady states to simple periodicity to highly complex patterns exhibiting self-similarity.

Figure 1 presents examples of patterns generated by LLCF over subsequent iterations. Figure 2 depicts the output space of LLCF as a is varied from 0 to 3. Both examples were generated with $m = 1$. With these charts, some basic observations can be made regarding the scenario where $m = 1$. In the case of $a < 1$, each x_t is scaled downward by a, so with $b = 0$, the result will always be an exponential decay from x_0 towards 0, i.e. $x_t = x_0 a^t$. Of course, with $b > 0$ and $a = 0$, the system always resolves immediately to b since the past terms are zeroed. When $a + b < 1.0$, x_t converges to a steady

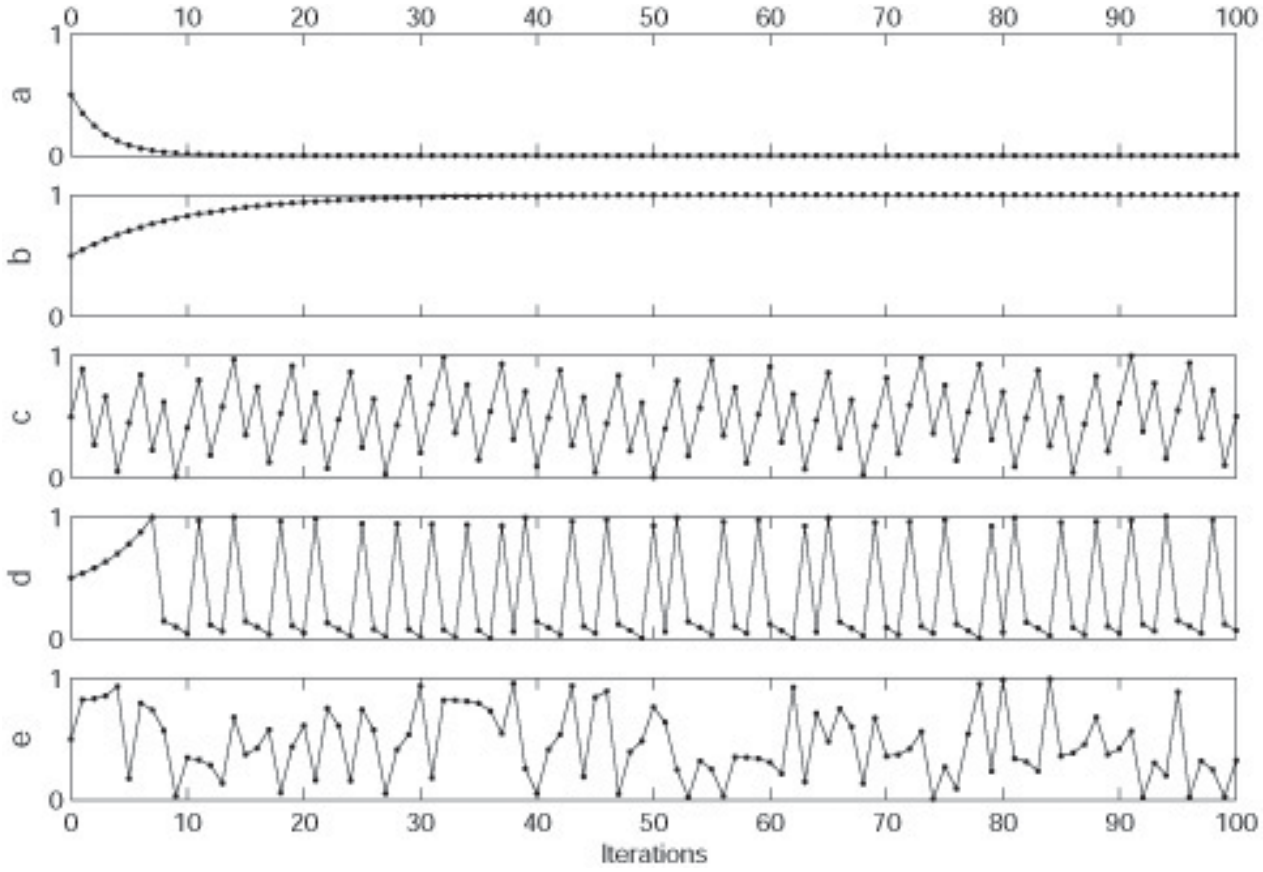


Figure 1. A subset of possible behaviours exhibited by Lehmer's Linear Congruence Formula with seed $x_0 = 0.5$ and $m = 1.0$. (a) $a = 0.7$, $b = 0.0$; (b) $a = 0.9$, $b = 0.1$; (c) $a = 0.98$, $b = 0.4$; (d) $a = 1.23$, $b = 0.92$; (e) $a = \pi$, $b = 0.25$.

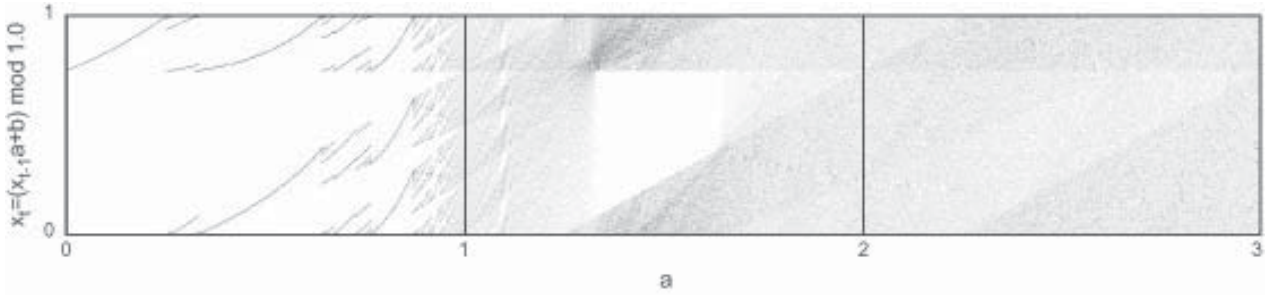


Figure 2. A map of a segment of the a-parameter space of Lehmer's Linear Congruence Formula. With $m = 1.0$ and $b = 0.75$, a varies from 0 to 3. At each increment of a, the seed x_0 was chosen randomly between 0 and 1, the system was iterated 200 times, and the points were plotted in the corresponding column. Grey levels indicate how often a point was generated, with darker points indicating higher occurrence.

state. When $a + b > 1.0$, simple periodicity arises. It is clear that the most complex behaviour arises when $a > 1$. Any $b > 1$ functions the same as b mod 1, so there is no need to consider cases other than $0 <= b <= 1$.

The author has used LLCF for generating pitch, amplitude and rhythmic patterns in both interactive and non-realtime algorithmic compositions. In a typical application, m matches the data range of the particular parameter being controlled, i.e m might be set to 128 and the output of the formula mapped directly to MIDI velocity. This type of approach, though quite simple and direct, can in itself prove fruitful. However, it also has the shortcoming that it provides no high-level modulation of behaviour. It should be noted at this point that linear modulation of the variables in the formula does not linearly impact the system. Rather, small changes in the variables typically cause dramatic changes in the behaviour of the system. So the approach of modulating a variable to induce predictable change over time is not applicable in most configurations of LLCF.

3. VARIABLE-COUPLED MAP NETWORKS

A variable-coupled map network is composed of one or more nodes, each consisting of an iterated map function with implementation of a time delay variable that schedules the timing of subsequent iterations of the function. The state of a node, broadcast when the function is iterated, can be routed to inputs on any node in the network, including the originating node. Potential types of node inputs include direct access to the node state, control of variables of the iterated map function, and control of the delay time. In all of the examples in this paper, each node operates independently, in the sense that the option of setting the node state directly is not utilised; the state changes only via normal iteration of the node and not by direct alteration by an external input.

Except where otherwise noted, the networks in this paper were implemented in LISP, leveraging Rick Taube's *Common Music* musical data classes and MIDI-file output (Taube 2003). For simplicity, the model was restricted to iterated functions in which both outputs and inputs range in value between zero and one. This allowed direct connection of outputs to inputs without rescaling issues. Each node variable could receive only one input. Each node contained inputs for the function variables. It also contained input for a delay scalar, which was mapped to the range between minimum and maximum delay times established when the node was initialised. When a node was updated at its scheduled time, first the current state of the node was broadcast from the node output and target inputs were updated. Then, based on the current value of the node's variables – which may have been updated in the broadcast step – the new state was calculated and the next update of the node was scheduled.

Thus, since we are using LLCF as the iterated map in these examples, m, a, b and x will all remain in the range from 0 to 1 – the range in which a single LLCF's behaviour is simple and short-term periodic.

For the purposes of creating note patterns, the iteration of a node can be used to trigger a function, such as one to generate a note. There are numerous possible approaches to this, but the model used here was that the state value of the triggering node determined the pitch of a triggered note. Other nodes in a network might be used to determine duration, amplitude, or other parameters for a sound. Or, multiple nodes may provide data for controlling and directing a more complex sound-generating function.

Note that the following observations of network behaviours are specific to the use of LLCF as the iterated map and likely will not apply when a different map is utilised.

4. SELF-DRIVEN NODES

We will use the term *driven* to describe a node that receives data at its variable inputs. *Undriven* describes a node that receives no inputs. We will use the term *autonomous* to describe a node that has no inputs from other nodes, and *semi-autonomous* to describe nodes that have inputs from other nodes. Clearly, an autonomous node may be undriven, meaning that it is a simple iterative map function. However, an autonomous node may also be driven, meaning that the output of the node is routed to one or more of its own inputs. In this case we can say that that the node is *self-driven*.

Using LLCF for the map type in a node, there are three different self-driven node configurations available. These can be described in the following shorthand, where a node label ('1' in this case) appears on the left and an arrow points to the target node and variable name appearing on the right. [1 → 1a] represents the routing of the output of the node 1 to the a-variable input of node 1. Note that this effectively transforms the node function into $x_{t+1} = (x_t^2 + b) \bmod m$. [1 → 1b] transforms the node function into $x_{t+1} = (x_t a + x_t) \bmod m$. [1 → 1a, 1b] transforms the node function into $x_{t+1} = (x_t^2 + x_t) \bmod m$.

Figure 3 presents output maps of these three configurations. [1 → 1a] generates stable states or short-term periods for all values of b. [1 → 1b] emphases exponential climbing behaviours with lower values of a, breaking up into more complex patterns as a moves towards 1. [1 → 1a, 1b] emphasises exponential climbs across the full range of seed values, and it is susceptible to falling into a 0-state, from which it cannot escape.

Note that [1 → 1a, 1b] provides inputs for all node variables, making the node *fully driven*, in contrast with the other two configurations which are *partially*

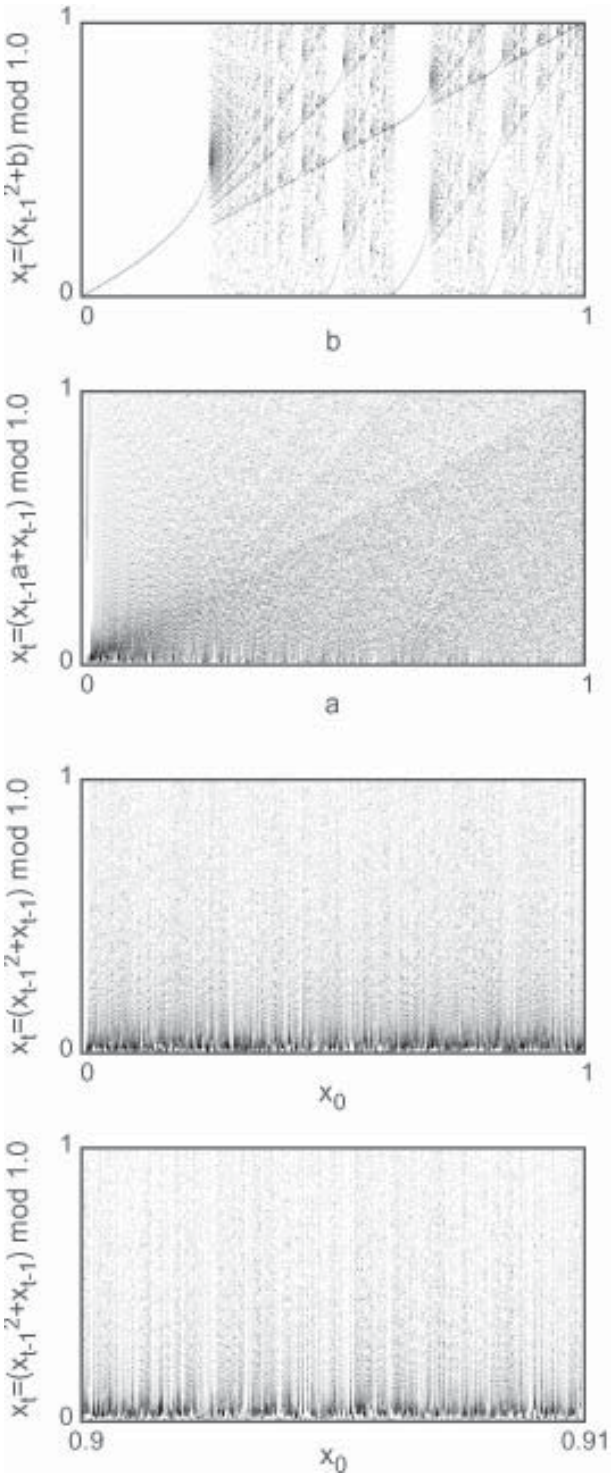


Figure 3. Outputs of functions that can be derived by linking the output of an LLCF node to one or more of its own inputs: [1 → 1a], [1 → 1b] and [1 → 1a, 1b]. For each column, a random seed was chosen and the map was iterated thirty times initially to discard initial transients. Then the map was iterated 200 times and the points were plotted in the corresponding column. Grey levels indicate how often a point was generated, with darker points indicating more frequent occurrence. The fourth plot demonstrates that [1 → 1a, 1b] exhibits the same general characteristics even as we zoom in on a narrower range of seed values.

driven. In the fully driven condition, the initial value of the variables is irrelevant; the initial seed for the node is the only determinant of its behaviour. This is due to the fact that the initial step – broadcasting of node states – will set all variables prior to the first iteration, when they will be referenced. Figure 3 shows that for [1 → 1a, 1b], the system is extremely sensitive to the initial state, meaning that significantly different behaviour arises with only a small shift in the seed. This means that the same configuration and initial values may produce a different output stream on a different operating system or even in different software on the same operating system, since even very small differences introduced in the sequence of floating point values will produce different subsequent values.

Figure 4 and audio example 1 on the *Organised Sound* Volume 9 CD present a glimpse of the temporal behaviour of the [1 → 1a, 1b] configuration.

5. NETWORKS WITH TWO NODES

We can describe a variable-coupled map network itself as being fully driven when all nodes in the network are fully driven. With two nodes, this leaves us with one fully *cross-driven* configuration, meaning a configuration in which no node is self-driven. We depict the node connections as follows, with the outer set of brackets enclosing the whole network:

[[1 → 2a, 2b], [2 → 1a, 1b]]

There are two *mixed-drive* configurations, in which self- and cross-connections are both utilised:

[[1 → 1a, 2b] [2 → 1b, 2a]]
[[1 → 1b, 2a] [2 → 1a, 2b]]

Because these systems drive all variables, the only way to establish their behaviour is through setting of the initial seeds. As it turns out, both the cross-driven example and the example in which *a* is cross-driven will generate the same pattern relative to their respective seeds, i.e. they will move in parallel. In the case of [1 → 1a, 2b], [2 → 1b, 2a], on the other hand, the two nodes quickly converge on the same value and from then on generate the same sequence. In other words, these fully driven networks do not exhibit significantly different patterns from autonomous nodes, and one must be aware of the potential for network configuration to lead to convergence of node behaviour.

There are numerous potential configurations of partially driven networks of two nodes. A subset of these is *hierarchical* networks in which there is no feedback between nodes. Control flows from one node to another and not back. These arrangements would be [[1 → 2a]], [[1 → 2b]], and [[1 → 2a, 2b]]. As there is no feedback, neither LLCF node will depart from its simple cyclical behaviour. The upper node of the

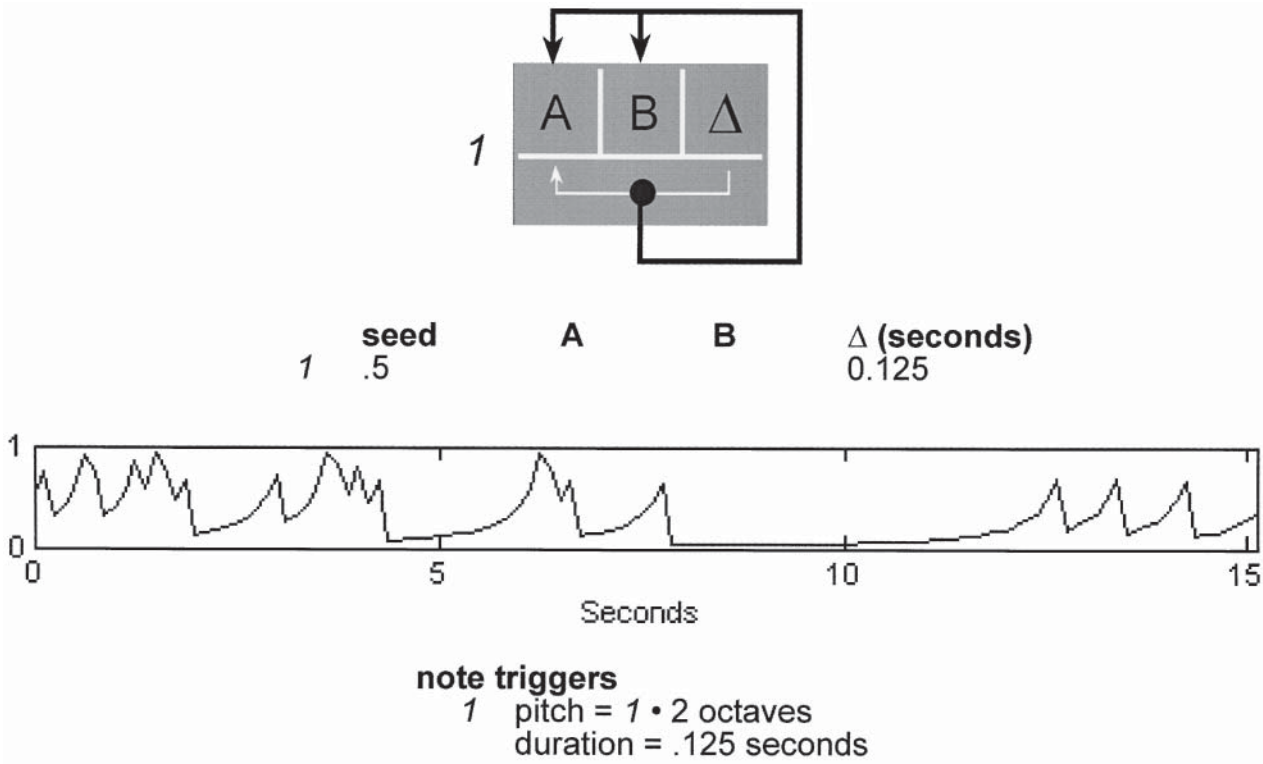


Figure 4. Diagram, initialisation, output, and note triggering of a network in which the output of an LLCF node sets the *a* and *b* values for the next iteration of the same node. The time-delta input for the node controls the time between iterations. In this case, this time factor remains constant since no nodes are connected to the time-delta input. Italicised numbers are node labels. In the trigger description, the • symbol indicates a mapping of a node output to data, in this case, the mapping of node *1* to a two-octave range of MIDI notes.

hierarchy will modulate the lower node between different short-term periodic patterns.

The node at the bottom of the hierarchy can self-drive, as in [[1 → 2a] [2 → 2b]]. The bottom node in these configurations will usually generate patterns in the class of those generated by a self-driving autonomous node. Or each node can be partially self-driving, but still arranged in a hierarchy of control, such as: [[1 → 1a, 1b, 2a] [2 → 2b]].

We break the hierarchical pattern by establishing feedback from low in the hierarchy to the node at the top of the hierarchy, as in [[1 → 1a, 2a] [2 → 1b]]. In these configurations, there is a strong tendency for the top and bottom node to lock into a parallel pattern. An exception is the configuration [[1 → 2a] [2 → 1a]], which always creates simple cycles or steady states.

6. THREE OR MORE NODES

As more nodes are added, potential coupling patterns expand rapidly and it becomes increasingly difficult to provide any kind of analytical generalisation regarding the connections. In practical usage, trial-and-error is the most likely method for discovering compositionally useful patterns. One can develop a heuristic sense of how different systems behave, particularly with the help of the above observations regarding single and dual node configurations. A plotter for visualising output patterns is helpful, particularly as it can help reveal non-audible nodes that are converging or not behaving in a predicted manner.

Figure 5 and audio example 2 present a partially driven network of three nodes. The network contains feedback to all nodes, so it is not hierarchical. However, given the symmetry of the system around node *2* and the fact that node *2* is used to trigger notes, one might interpret this as a system in which nodes *1* and *3* form a higher hierarchical level driving node *2*. In any case, though the configuration produces some new behaviour vocabulary, it does not provide any mid-term modulation of behaviour.

Now that we have three nodes, we can leverage this data to control additional sound parameters. Audio example 3 was generated with the figure 5 configuration, but whenever a note was generated, the current states of nodes *a1* and *a3* determined note amplitude and duration, respectively. Now a much richer and arguably more musical sequence is generated. Even with a constant delay time for all nodes, this approach often provides an engaging and surprisingly gestural articulation of the line. Note that the rendering process for all audio examples for this article discarded notes with durations shorter than 0.05 seconds.

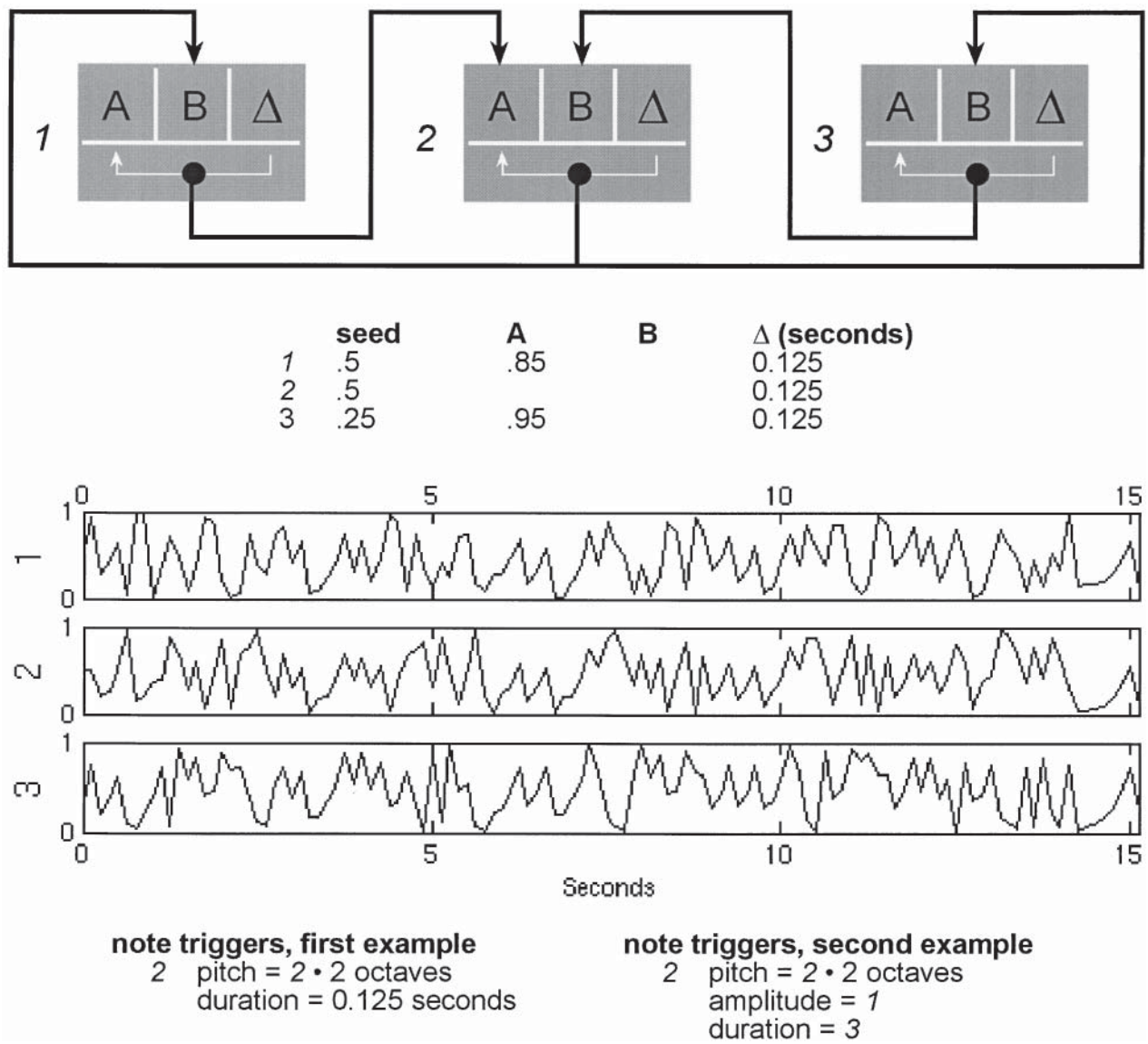
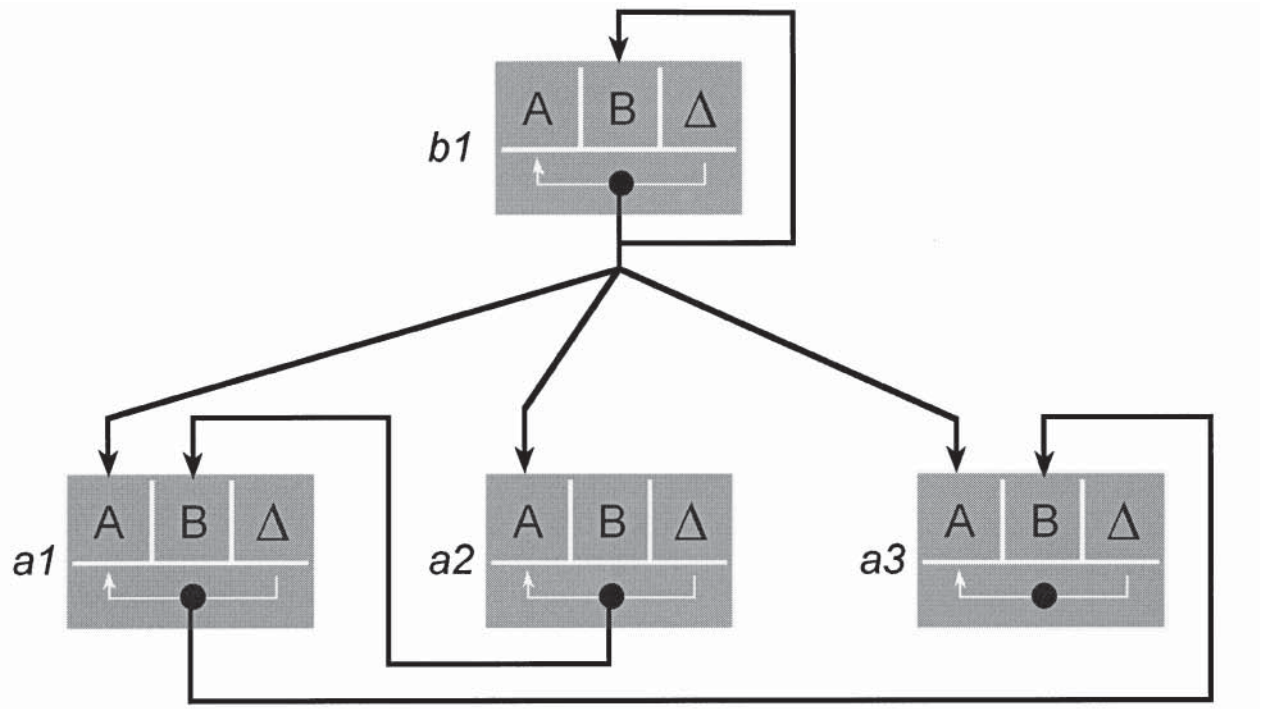


Figure 5. A partially driven, non-hierarchical system of three nodes.

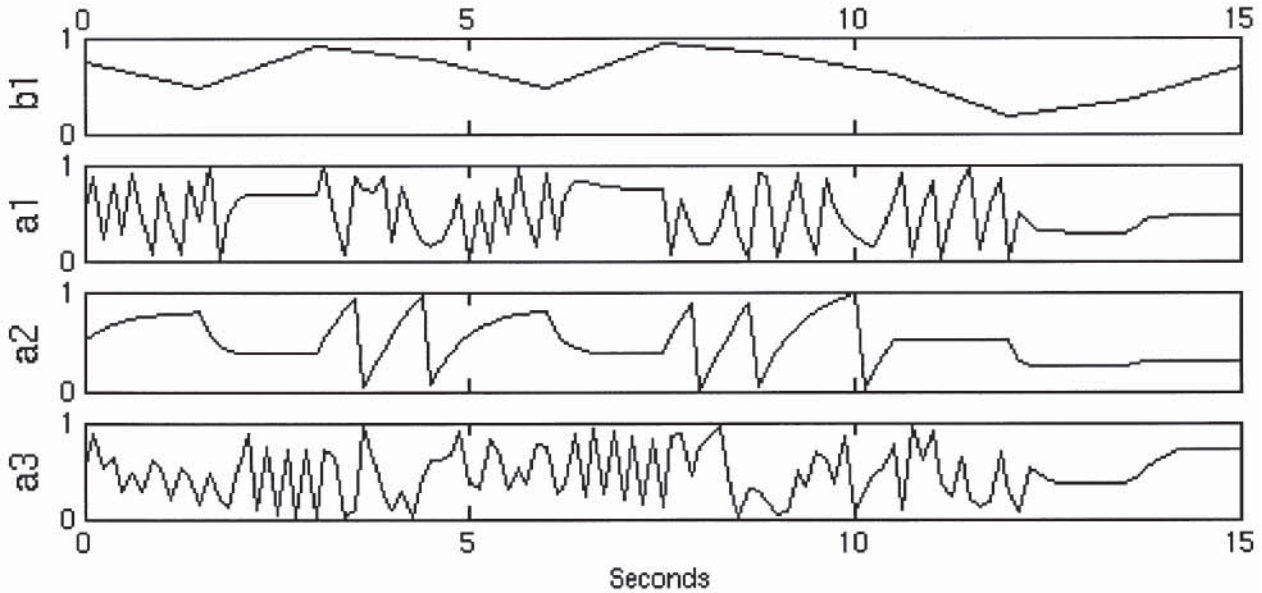
Hierarchy is one way to approach providing mid-term modulation of behaviour. Figure 6 and audio example 4 present a partially driven network consisting of a bottom tier of three nodes, each of which is driven in the *a* parameter by a single, self-driven control node. The control node runs at a slower rate than the bottom tier nodes. Because this control node is driving its own *b* variable, it will normally produce relatively complex behaviour. Nodes *a1* and *a3* are both fully driven, and their outputs, though not identical, clearly modulate together to create similar classes of behaviour as the control node changes. *a2* is autonomous with regards to its *b* variable, so the individual configuring the system can control that variable to tune behaviour. Notice in both the graphs and the rendered audio that there are now plateaus of different behaviours.

Figure 7 and audio example 5 extend the idea of the previous example by establishing two fully autonomous control nodes with symmetrical control of the two outer nodes of the lower tier. The two control nodes again run at a slower rate than the lower tier, but they also run at different rates from each other. Given that control descends from the upper level to *a1* and *a3* and then to *a2*, the network could also be interpreted as a 3-tier system of descending control. However, the nodes *a1*, *a2* and *a3* all operate at the same speed and provide duration, note trigger and pitch, and amplitude, respectively, so they can also be interpreted to form a single tier. The audible outcome of the network is rich and shifting in gestural character.

In figure 8 and audio example 6, the above network is elaborated to be fully driven with upward links breaking the strict descending hierarchy.



	seed	A	B	Δ (seconds)
b1	.75	.95		1.5
a1	.5			0.125
a2	.5		.2	0.125
a3	.5			0.125



note triggers
2 pitch = *2* • 2 octaves
amplitude = *1*
duration = *3*

Figure 6. A hierarchical node provides ongoing modulation of behaviour of a tier consisting of three cross-driven nodes.

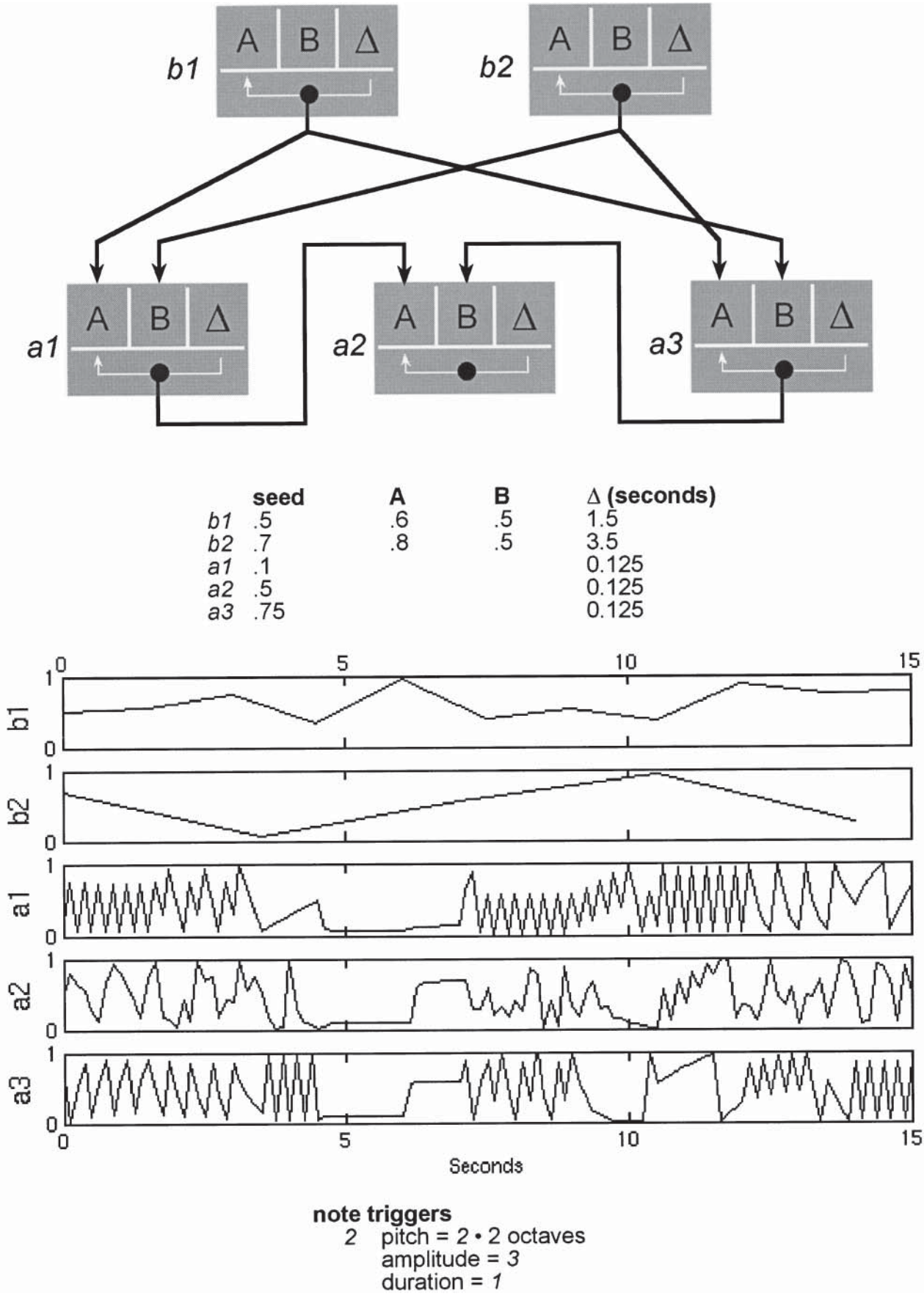
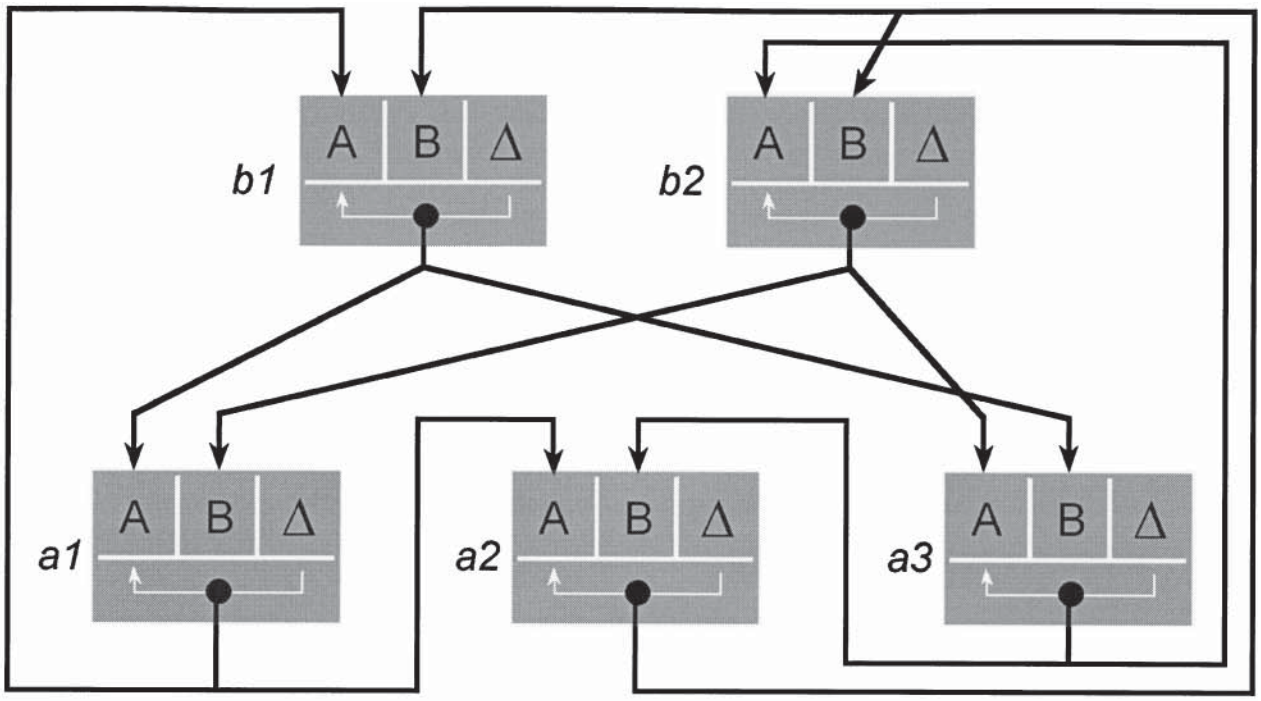
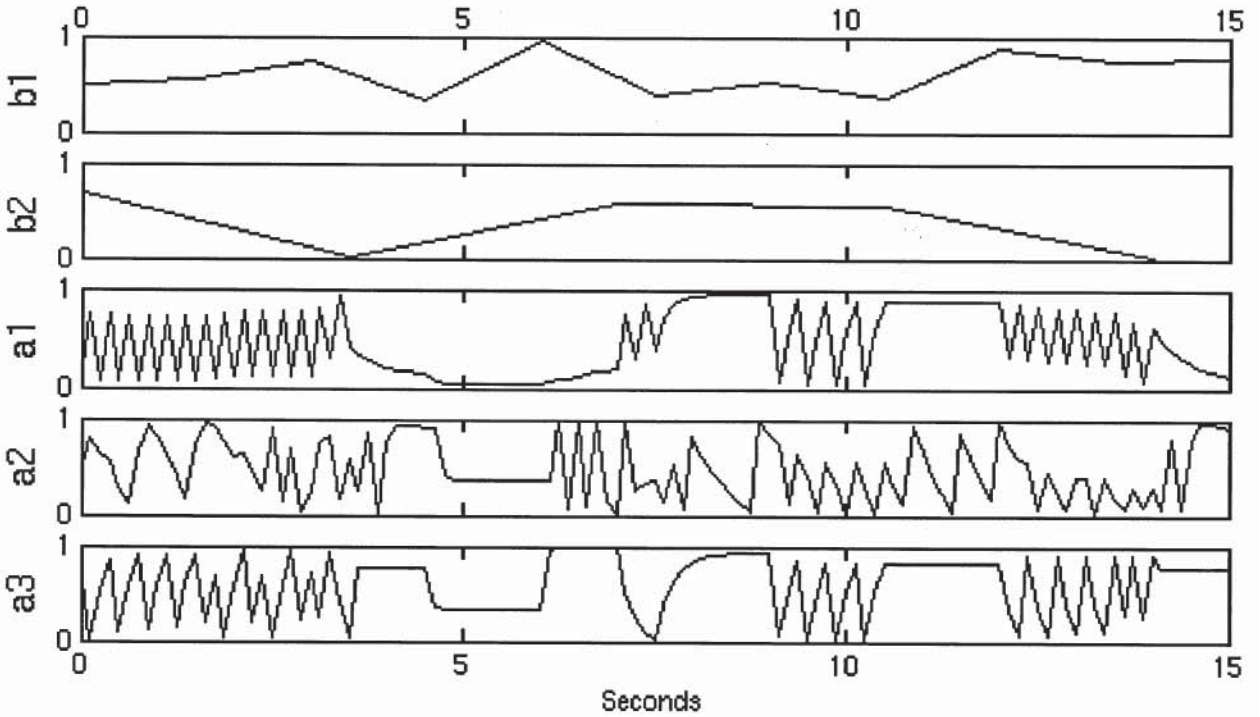


Figure 7. A two-tier system of descending control.



	seed	A	B	Δ (seconds)
b1	.5			1.5
b2	.7			3.5
a1	.1			0.125
a2	.5			0.125
a3	.75			0.125



note triggers
2 pitch = *2* • 2 octaves
amplitude = *3*
duration = *1*

Figure 8. The figure 7 network elaborated to be fully driven with both descending and ascending control flow.

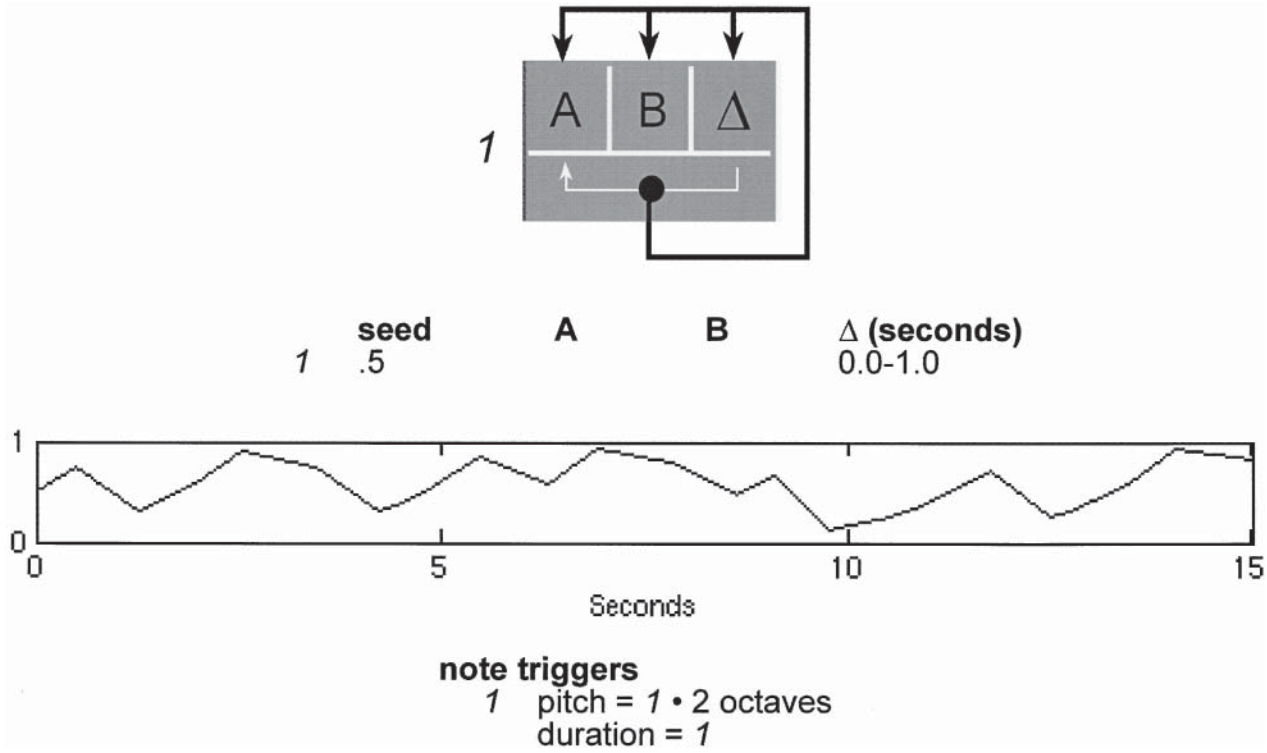


Figure 9. A self-driving node that also drives its own delay time.

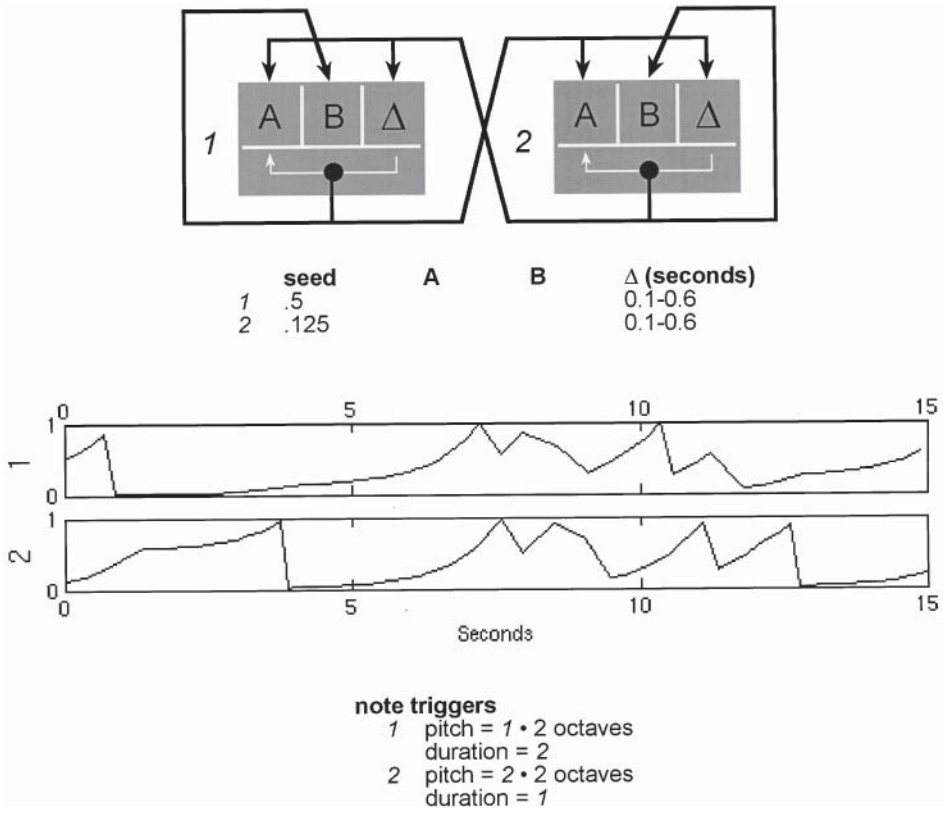


Figure 10. A fully driven network of two nodes, including time-modulation.

7. SYSTEMS WITH TIME DELAY MODULATION

As we have seen, even with a constant-delay network, modulation of amplitude and duration of note events can produce a rich rhythmic vocabulary. However, modulation of the delay times of individual nodes will clearly be of value towards generating greater rhythmic diversity. If we approach this by allowing one node to change the timing characteristic of other nodes, it also adds a profound dimension to the interaction of the nodes, adding considerably to the complexity and richness of the behaviour. One of the most striking results of this approach is the network's capacity to make complex but gesturally coherent rhythmic sequences on a completely unquantised time continuum.

In figure 9 and audio example 7, the figure 4 configuration is presented with one connection added: a link from the output of the node to its own time-delta input. The time-delta input is a scalar between 0 and 1, and it is mapped to the time-delay range of the node. The time-delay range, defined by a minimum delay time and a maximum delay time, is defined at node initialisation. In the figure 9 example, the minimum delay time is 0. This obviously could be problematic in some contexts, since a very large number of notes could be generated, overflowing the system. Although this particular example works with a delay time of 0, the following examples are more typical in that they use a non-zero minimum delay. In the case of figure 9, the resulting behaviour is simple: as the node's state rises, its delay time rises. In the given musical mapping, there is a direct correspondence between rise in pitch and an increase in duration.

In figure 10 and audio example 8, two nodes are arranged in a fully driven system. In contrast to the early discussions of dual-node networks, here each node modulates the other's delay time. Each node triggers its own sequence of notes with the duration set by the opposite node. The two nodes exhibit similar behaviour, but not in unison or parallel.

Figure 11 and audio example 9 present a partially driven system of three nodes in which the delay time for all three is modulated by another node. The network exhibits distinct plateaus and shifts of behaviour even without a clear higher-level control node

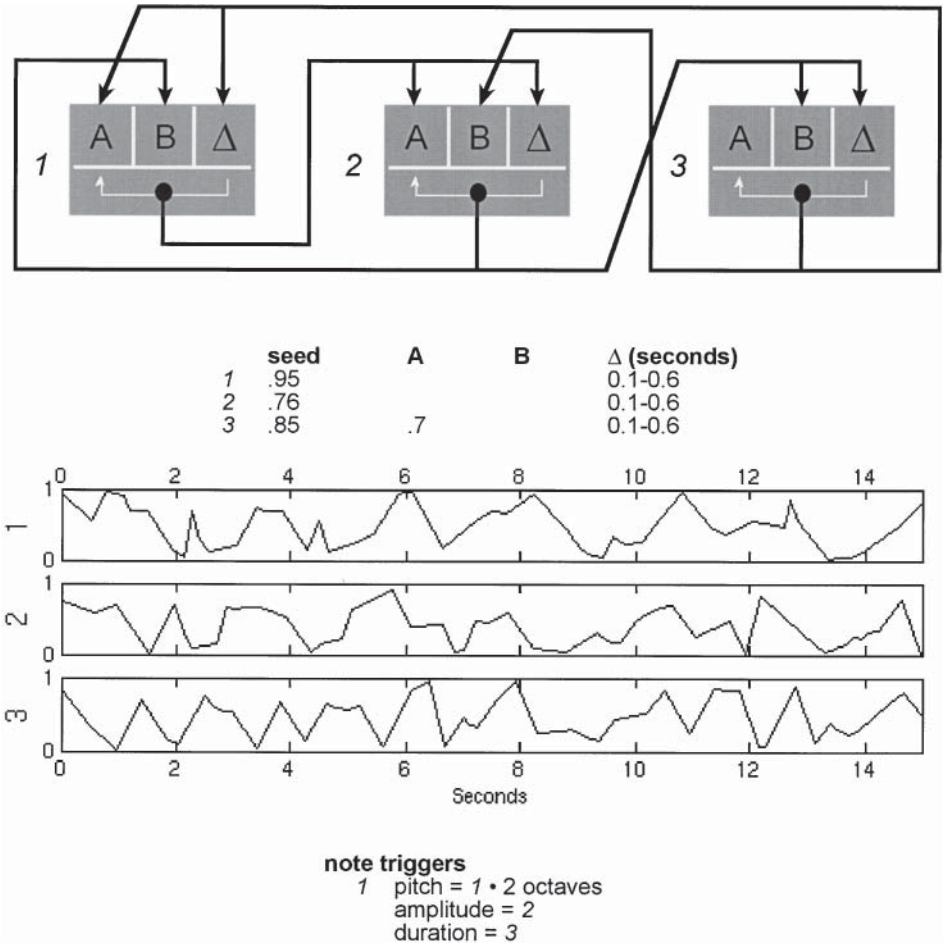
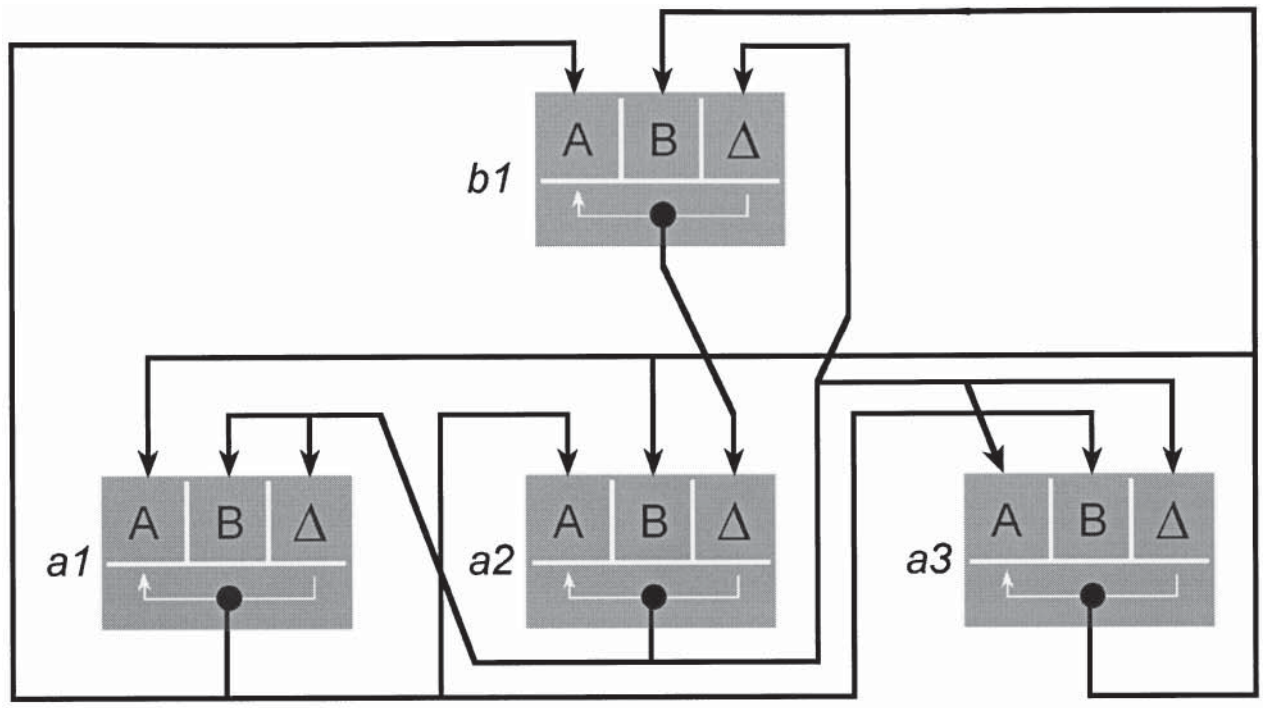
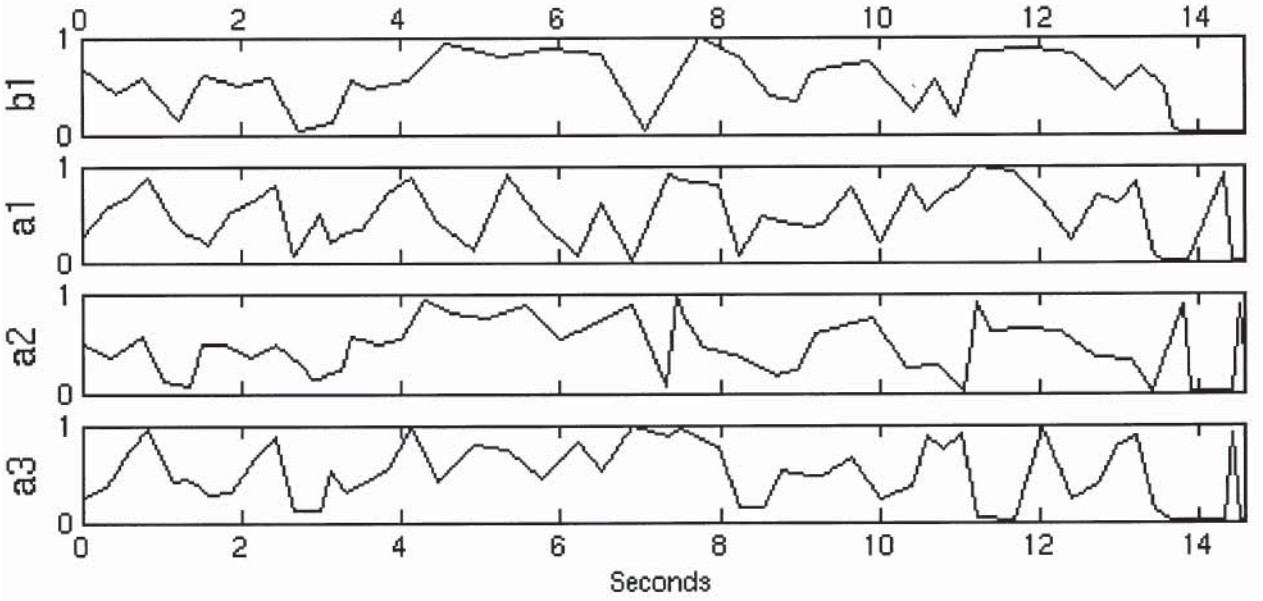


Figure 11. A partially driven network of three nodes.



	seed	A	B	Δ (seconds)
b1	.69			0.1-0.75
a1	.23			0.1-0.5
a2	.5			0.1-0.5
a3	.25			0.1-0.5



note triggers
2 pitch = 2 • 2 octaves
amplitude = *3*
duration = *1*

Figure 12. A fully driven, time-modulated feedback network.

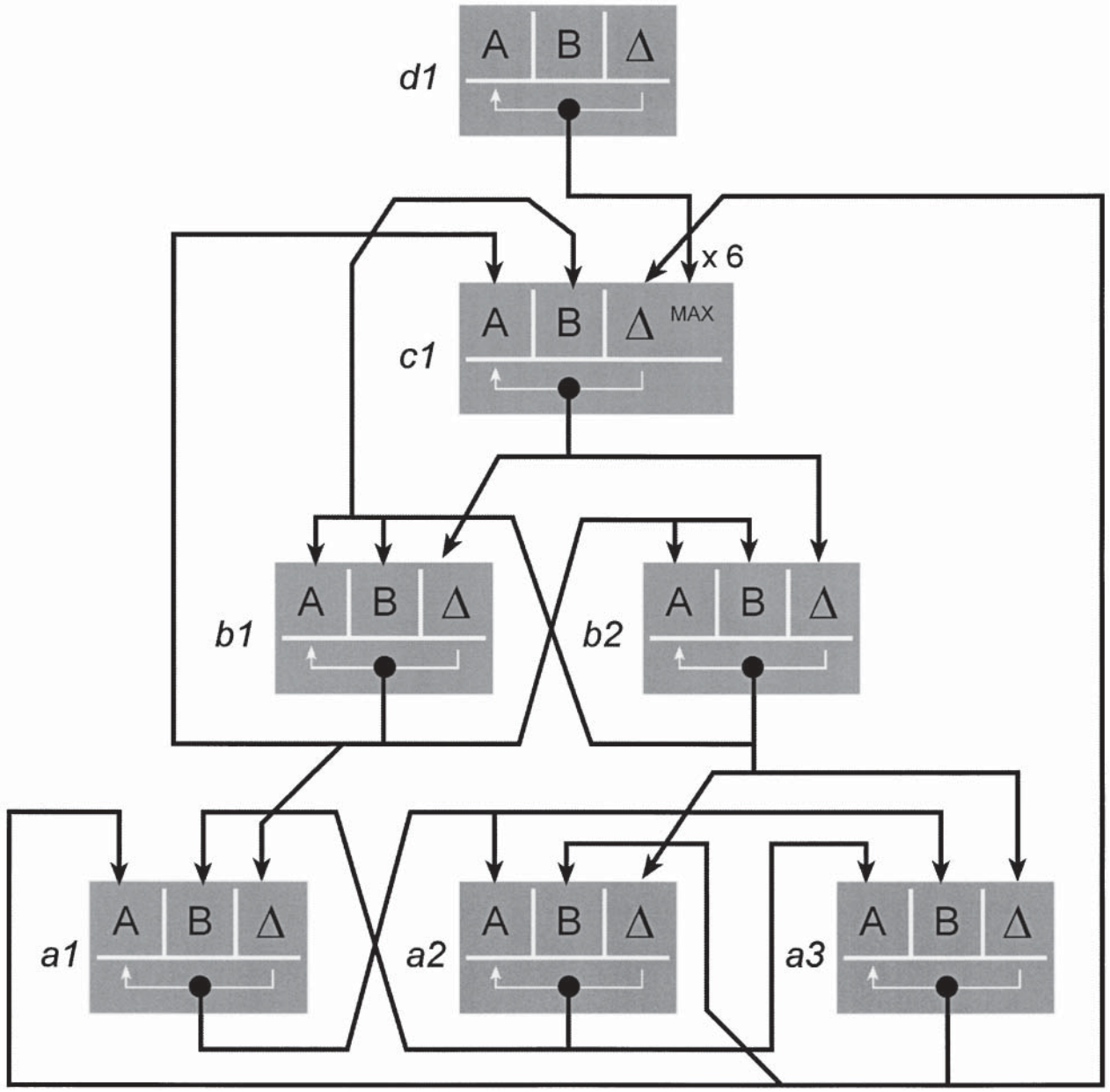


Figure 13. A more complex network consisting of four levels of descending control and limited upward feedback. Node *d1* controls the maximum delay time of node *c1*.

modulating activity. But what happens if one expands the system to include a hierarchical tier? Figure 12 and audio example 10 present a fully driven system with one upper-tier control node and three lower-tier nodes with feedback from the lower to upper tier. This simple system generates a significant diversity of behaviour and rhythmic gesture.

8. MORE COMPLEX NETWORKS

Figure 13 depicts a complex network involving seven nodes organised into four tiers with a strongly descending flow of control, with some upward feedback. The author's implementation files for this network are labelled *HMG*, short for the half-joking label *high-modernism generator*. One may listen to audio example 11 to understand why. At that point the reader can decide for him or her self whether the output of the network suggests that the variable-coupled map network approach has powerful musical potential. Of course, one might argue that the relative ease of this means of generating a potentially never-ending output in this style is a not-so-subtle slight on the depth of the style itself.

In contrast to the previous examples, this example was implemented in MAX/MSP. A more complex output triggering mechanism was established in which the state of the triggering node was mapped to a two-octave octatonic scale and the base octave was determined by another node. Both nodes *a1* and *c1* triggered notes, the latter providing the slower, sparser line, since maximum delay time was greater for nodes higher in the hierarchy. It should be noted that a translation of the set-up into the LISP implementation used for the earlier examples did not replicate the MAX/MSP results. This is due in part, no doubt, to the extreme sensitivity of the processes to small changes in state, as described earlier. It may also be due to the differences in the order in which nodes were assessed and data updated in the MAX/MSP environment.

Audio example 12 presents a compositional application of the same network topology in an excerpt from my work *Writing on the Surface* for computer-realised sound and video. The percussive materials in this selection were compiled from fifteen- to twenty-second segments generated by the network in which the network controlled a large ensemble of virtual percussion instruments. The instruments were divided into three groups based on pitch range, and different tiers were responsible for driving different groups. Slow tiers controlled lower pitch instruments. The network controlled pitch inflection, amplitude, and duration of notes, but also the choice of the specific instrument from a group and the location of the sound in quadraphonic space. By using my extensions to Common Music that enabled generation of animation scripts for the 3-D Studio Max animation software (Battey 2001), the same algorithms were also mapped to the creation and animation of a dynamic and complex composition of visual objects.

9. CONCLUSIONS

We have examined a range of configurations of variable-coupled iterated maps using Lehmer's Linear Congruence Formula. Such networks have proven capable of exhibiting complex behaviour, including plateaus and transitions between distinct temporal patterns. The addition of time modulation significantly expands the richness of the behaviours of such networks.

This paper has provided only an initial exploration of the potentials of such networks. LLCF itself generates a certain vocabulary, and that vocabulary is expanded by self-driving. Elements of that vocabulary arise throughout the different configurations of LLCF nodes. However, there are numerous other iterated maps that can be explored, each potentially providing a different base vocabulary. A network could consist of homogenous nodes or a heterogeneous mix of iterated map types. Further, while the systems in this paper were limited to a numerical range from 0 to 1, one can also explore wider data ranges. Quantised duration schemes could also be devised as an alternative to the continuous time expression approach used here. Large networks could be devised in which whole clusters of nodes function together in a particular role, and clusters could be arranged in both hierarchical and non-hierarchical relationships.

Variable-coupled iterated map networks also exhibit some of the disadvantages that arise when working with nonlinear dynamics. As we have seen, the extreme sensitivity of the iterated maps to initial conditions and variations in floating point data handling mean that specific configurations will not necessarily reproduce the same results in different software or operating systems. Further, our tools for analysing nonlinear dynamics are quite limited; in most cases, uses of such systems for generation of musical materials will involve a trial-and-error approach.

ACKNOWLEDGEMENTS

The author would like to thank Gary Nelson, who demonstrated the usefulness of LLCF as a pattern generator while the author was his student at Oberlin Conservatory in the late 1980s.

REFERENCES

Ames, C. 1992. A catalog of sequence generators: accounting for proximity, pattern, exclusion, balance, and/or randomness. *Leonardo Music Journal* **2**(1): 55–70.

Battey, B. 2001. An animation extension to Common Music. *Proc. of the Eighth Biennial Symp. on Arts and Technology at Connecticut College*, pp. 6–11.

Gogins, M. 1991. Iterated functions systems music. *Computer Music Journal* **15**(1): 40–8.

Kaplan, D., and Glass, L. 1995. *Understanding Nonlinear Dynamics*. New York: Springer-Verlag.

Laine, P. 1997. Generating musical patterns using mutually inhibited artificial neurons. *Proc. of the 1997 Int. Computer Music Conf.*, pp. 422–5.

Laine, P. 1999. Motor neuron based virtual drummer. *Proc. of the 1999 Int. Computer Music Conf.*, pp. 194–6.

Leach, J., and Fitch, J. 1995. Nature, music, and algorithmic composition. *Computer Music Journal* **19**(2): 23–33.

Kaneko, K. 1993. *Theory and Application of Coupled Map Lattices*. Chicester, NY: John Wiley & Sons.

Palacios, A. 2001. Cycling chaos in one-dimensional coupled iterative maps. *International Journal of Bifurcation and Chaos* **12**(8): 1,859–68.

Taube, R. 2003. Common Music 2.4.0. http://commonmusic.sourceforge.net/

Wolfram, S. 2002. *A New Kind of Science*. Champaign, IL: Wolfram Media.