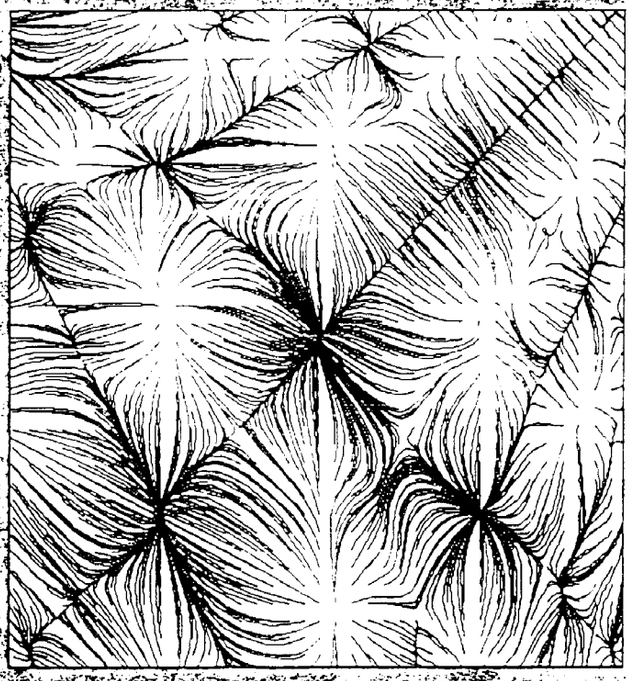
Music, Mind and Machine

Studies in Computer Music, Music Cognition and Artificial Intelligence



Peter Desain and Henkjan Honing

Kennistechnologie

Peter Desain and Henkjan Honing
Centre for Art, Media, and Technology
Utrecht School of the Arts
Lange Viestraat 2b
NL-3511 BK Utrecht
The Netherlands

Music Department
City University
Northampton Square
London EC1V 0HB
United Kingdom

The Quantization of Musical Time: A Connectionist Approach

Introduction

Musical time can be considered to be the product of two time scales: the discrete time intervals of a metrical structure and the continuous time scales of tempo changes and expressive timing (Clarke 1987a). In musical notation both kinds are present, although the notation of continuous time is less developed than that of metric time (often just a word like "rubato" or "accelerando" is notated in the score). In the experimental literature, different ways in which a musician can add continuous timing changes to the metrical score have been identified. There are systematic changes in certain rhythmic forms: for example, shortening triplets (Vos and Handel 1987) and timing differences occurring in voice leading with ensemble playing (Rasch 1979). Deliberate departures from metricality, such as rubato, seem to be used to emphasize musical structure, as exemplified in the phrase-final lengthening principle formalized by Todd (1985). In addition to these effects, which are collectively called *expressive timing*, there are nonvoluntary effects, such as random timing errors caused by the limits in the accuracy of the motor system (Shaffer 1981) and errors in mental time-keeping processes (Vorberg and Hambuch 1978). These effects are generally rather small—in the order of 10–100 msec. To make sense of most musical styles, it is necessary to separate the discrete and continuous components of musical time. We will call this process of separation *quantization*, although the term is generally used to reflect only the extraction of a metrical score from a musical performance.

Computer Music Journal, Vol. 13, No. 3, Fall 1989.

Perception of Musical Time

Human subjects, even without much musical training, can extract, memorize, and reproduce the discrete metrical structure from a performance of a simple piece of music—even when a large continuous timing component is involved. This is surprising, given that the note durations in performance can deviate by up to 50 percent from their metrical values (Povel 1977). Indeed, it seems that the perception of time intervals on a discrete scale is an obligatory, automatic process (Sternberg, Knoll, and Zukofsky 1982; Clarke 1987b). This so-called categorical perception can also be found in speech perception and vision. By contrast, the perception and reproduction of continuous time in musical performance seems to be associated with expert behavior.

Once the discrete and continuous aspects of timing have been separated by a quantization process, each can function as an input to other processes. The induction of an internal clock (Povel and Essens 1985) and the reconstruction of the hierarchical structure of rhythmical patterns (Mont-Reynaud and Goldstein 1985) both rely on the presence of a metrical score, while Todd (1985) has developed a model in which hierarchical structure is recovered from expressive timing alone.

Applications of Quantization

Apart from its importance for cognitive modeling, a good theory of quantization has technical applications. It is one of the bottlenecks in the automatic transcription of performed music, and is also important for compositions with a real-time, interac-

Fig. 1. Example of a performed score and its quantization by a commercial MIDI Package using a resolution of 1/64 note.

tive component where the computer improvises or interacts with a live performer. Last but not least, a quantization tool would make it possible to study the expressive timing of music for which no score exists, as in improvised music.

Known Methods

Few computational models are available in the literature for separating a metrical score from expressive timing in performed music (Desain and Honing 1988). Available methods produce a considerable number of errors when quantizing the data. The traditional approach is to expand and contract note durations according to a metrical grid that is more or less fixed—the grid being adjustable to incorporate different, low-level subdivisions (e.g., for triplets). Commercial MIDI software uses this method, which often gives rise to a musically absurd output, as shown in Fig. 1. Better results are obtained when the system tracks the tempo variations of the performer (Dannenberg and Mont-Reynaud 1978), though the system still returns an error rate of 30 percent. More sophisticated artificial intelligence (AI) methods use knowledge about meter (Longuet-Higgins 1987) and other aspects of musical structure. A particularly elaborate system originated at the CCRMA center at Stanford University in the automatic transcription project (Chowning et al. 1984). This knowledge-based method uses information about different kinds of accent, local context, and other musical clues to guide the search for an optimal quantized description of the data. It is entirely implemented in a symbolic, rule-based paradigm. This approach can be seen as the antithesis of our approach, in which all knowledge in the system is represented implicitly. We took the connectionist approach because knowledge-based approaches seemed to offer no real solution to manifest inadequacies of the simplistic metrical grid method. As with the majority of traditional AI programs, the sophisticated knowledge these AI methods use is extremely domain dependent (depending on a specific musical style), causing the systems to break down rapidly when applied to data foreign to this style.

Connectionist Methods

Connectionism provides the possibility for new kinds of models with characteristics traditional AI models lack, in particular robustness and flexibility (Rumelhart and McClelland 1986). Connectionist models consist of a large number of simple elements, each of which has its own activation level. These cells are interconnected in a complex network, with the connections serving to excite or inhibit other elements. One broad class of these networks, known as *interactive activation and constraint satisfaction networks*, generally converge towards an equilibrium state given some initial state.

An example of the application of these networks to music perception is given by Bharucha (1987) in the context of tonal harmony. These networks have not yet been used for quantization. The quantization model presented in this paper is a connectionist network designed to converge from nonmetrical performance data to a metrical equilibrium state. This convergence is hard wired into the system, and no learning takes place. The model is thought of as a collection of relatively abstract elements, each of which performs a rather complex function compared to standard connectionist models. While it may be possible to express these functions in terms of one of the formalisms for neural networks, this lies beyond the scope of the present article.

Basic Model

Consider a network with two kinds of cells: the *basic cell*, with an initial state equal to an interonset interval, and the *interaction cell*, which is connected in a bidirectional manner to two basic cells. Figure 2a shows the topology of a network for quantizing a rhythm of four beats, having its three

Fig. 2. Topology of a basic network (a) and a compound network (b).

Fig. 3. Interactive time intervals in a basic network (a) and a compound network (b).

Fig. 4. Interaction function with a peak at 4 and decay equal to −1.

Fig. 2

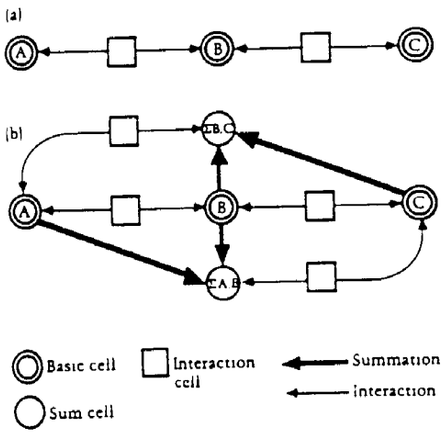


Fig. 3

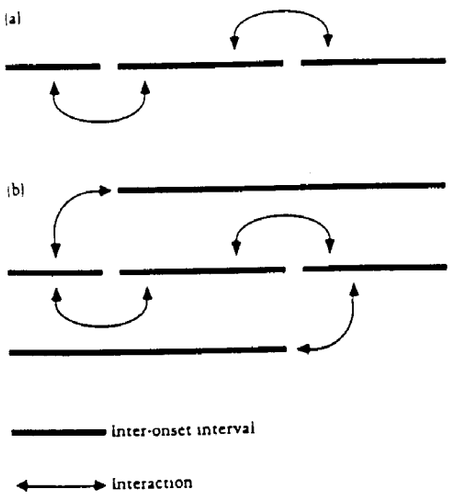
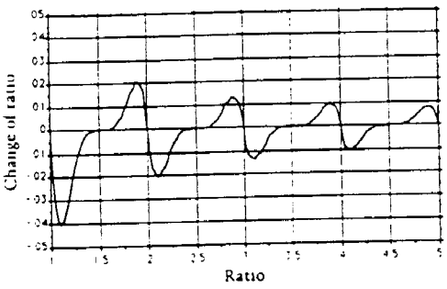


Fig. 4



inter-onset intervals set as states of the three basic cells, labeled A, B, and C. There are two interaction cells connected to the basic cells A and B, and B and C, respectively. Each interaction cell steers the two basic cells to which it is connected toward integer multiples of one another, but only if they are already near this state. It applies the interaction function to the quotient of their states (ratios smaller than 1 are inverted). If this ratio were close to an integer (e.g., 1.9 or 2.1), the interaction function would return a *change of ratio* that would steer the two states toward a perfect integer relation (e.g., 2). Figure 3 illustrates the interactions that are relevant in quantizing the four-beat rhythm. One can see that if the ratio is slightly above an integer, it will be adjusted downward, and vice versa as in Fig. 4.

There are constraints to be taken into account for interaction functions. First, the function and its derivative should be zero in the middle region between two integer ratios. In this region it is not clear if the integer ratio above or below is the proper goal, so no attempt is made to change the ratio. Second, the derivative around integer ratios should be negative to steer the ratio towards the integer, but greater than −1 to prevent overshoot that would result in oscillations. Third, the magnitude of the function should decrease with increasing ratios to diminish the influence of larger ratios. A large class of functions meet these constraints. At present we use a polynomial section around each integer ratio.

Fig. 5. State as a function of iteration count for the rhythm 2, 1, 3 in a basic network (a). State as a function of iteration count for the rhythm 1, 2, 3 in a basic network (b).

The degree of the polynomial, called the *peak* parameter, is typically between 2 and 12. To realize the decreasing magnitude of the interaction function, each section is scaled with a multiplication factor that is a negative power of the integer ratio. This power is called the *decay parameter*, and is typically between −1 and −3. This interaction function is defined as

$$F(r) = (\text{round}(r) - r) * |2(r - \text{entier}(r) - 0.5)|^p * \text{round}(r)^d,$$

in which the first term gives the ideal change of ratio, the second term signifies the speed of change which is at maximum near an integer ratio (with peak parameter p), and the third term scales the change to be lower at higher ratios (with decay parameter d). It is simple to prove that this interaction function satisfies the constraints mentioned.

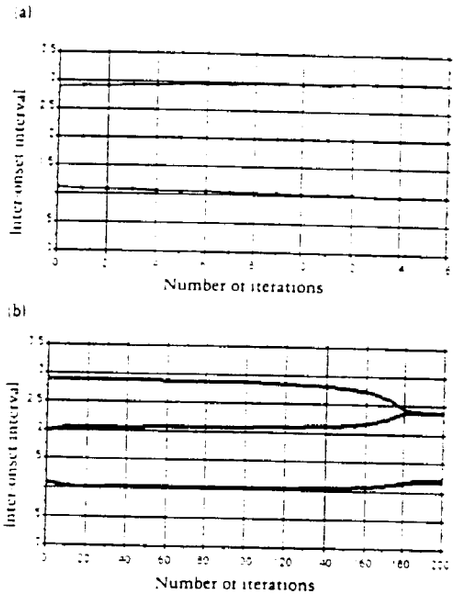
From the change of ratio $F(a/b)$, new intervals $a + \Delta$ and $b - \Delta$ are calculated without altering the sum of both intervals.

$$\frac{a + \Delta}{b - \Delta} = \frac{a}{b} + F\left(\frac{a}{b}\right)$$

which implies

$$\Delta = \frac{bF\left(\frac{a}{b}\right)}{1 + \frac{a}{b} + F\left(\frac{a}{b}\right)}$$

In simulating the network, each interaction cell updates the states of the two basic cells to which it is connected. This process is repeated, moving the basic cells slowly towards equilibrium. Equilibrium is assumed when no cell changes more than a certain amount between two iterations. For example, let us take a rhythm with inter-onset intervals of 2, 1.1, and 2.9 csec. As the representation of duration is currently unimportant in the model, they are treated as relative values (tempo has no influence on the quantization). This rhythm is represented in a basic network as three cells with the initial states 2.0 : 1.1 : 2.9. Iterating the procedure outlined above

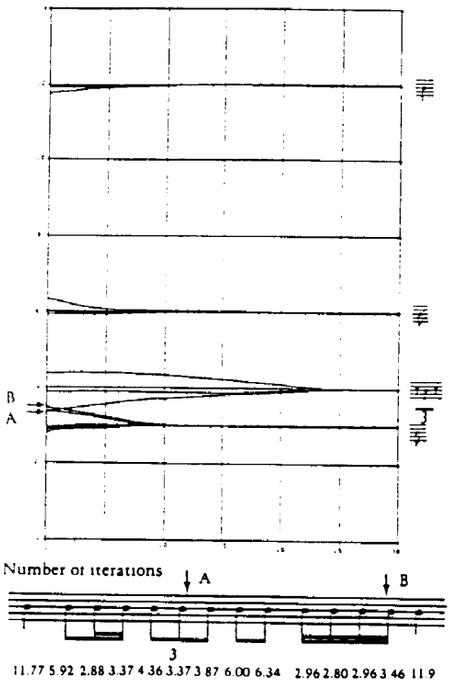


for the interactions between cells labeled A and B, and cells B and C will adjust the durations toward 2 : 1 : 3, where the net reaches an equilibrium. Figure 5a is a graph of the state of each basic cell as a function of the iteration count.

This type of network can of course only quantize very simple rhythms. Consider for instance the rhythm 1.1 : 2.0 : 2.9, which should converge to 1 : 2 : 3. The cell representing 2.9 only interacts with its neighbor 2.0, the resultant ratio 1 : 45 being a long way from an integer. The basic net adjusts these values to 1.2 : 2.4 : 2.4, as seen in Fig. 5b.

What the model fails to take account of is the time interval 3.1, the sum of the first two durations. If this interval were incorporated into the model, it would interact successfully with the third interval (2.9) in such a way that the pair of intervals would gravitate toward the ratio 1. This observation leads to a revised model.

Fig. 6. State as a function of iteration count for a complex rhythm in a compound network.



Compound Model

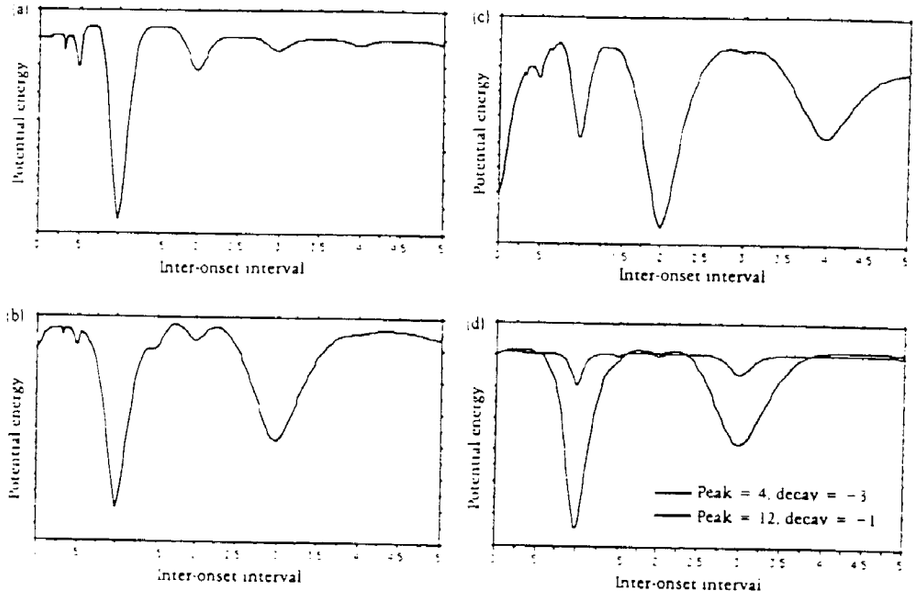
In order to represent the longer time intervals generated by a sequence of notes, *sum cells* are postulated. These cells sum the activation levels of the basic cells to which they are connected. The interaction of a sum cell with its basic cells is bidirectional: if the sum cell changes its value, the basic cells connected to it will all change proportionally. The sum cells are interconnected to cells representing adjacent intervals by the same interaction cells that are used in the basic model. The function of the interaction cells is once again to try to steer the interconnected cells—which may be sum cells, or a mixture of sum cells and basic cells—toward an integer ratio as was shown in Figs. 2b and 3b.

Our earlier example—a duration sequence of 1.1, 2.0, 2.9—is now quantized correctly due to combined effects of interacting sum cells and the interactions between the basic cells. Let us consider a more complex example using the real performance data shown in Fig. 6. In this rhythm the final sixteenth note is played longer than the middle note of the triplet. Nonetheless the local context of the two intervals steers each note towards its correct value as seen in Fig. 6. The compound model produces promising results, even though the network is rather sparse, allowing only adjacent time intervals to interact. A compound network for a rhythm of n intervals consists of n basic cells, $\lfloor (n + 1)(n - 2)/2 \rfloor$ sum cells, and $\lfloor n(n^2 - 1)/6 \rfloor$ interaction cells.

Understanding the Model

In connectionist systems the global behavior emerges from a large number of local interactions. This makes it very difficult to study the behavior of the network at a detailed level. While it may initially seem attractive to use descriptions like "winning cells," "pulling harder," etc., a better understanding of the patterns of change within the network and of the influence of context requires the development of specialized methods. An approach that has proved very useful is what we call the *clamping method*. This entails the clamping, or fixing, of the states of all but one of the cells. The remaining cell is given an activation level in a reasonable range (the independent variable). Then the resulting change that would have taken place—after one iteration—if the cell were free to change its activation level is monitored (the dependent variable). In order to facilitate the interpretation of this measure (the amount of change), the function is negated and integrated to give a curve with local minima at stable points. The state of the experimentally varied cell will tend to move towards a minimum, like a rolling ball on an uneven surface. As such, it can be interpreted as a curve of potential energy. These minima and maxima can now be evaluated and judged in light of the context set up

Fig. 7. Clamping curve for a cell with a left context of 1 (a). Clamping curve for a cell with a left context of 2, 1 (b). Clamping curve for a cell with a left context of 2, 1, 1 (c). Clamping curve for a cell with a left context of 2, 1 with different parameters for the interaction function (d).



by the surrounding clamped cells. We call the interval between two neighboring local maxima the *catch range*. A value occurring within this range will move towards the minimum between these two maxima, provided the context does not change. The size of the interval where the potential energy stays close to a minimum is called its *flatness value*. It is a measure of the lack of clarity in the context; simple and clear contexts give rise to sharp minima.

Figure 7a shows the potential energy curve of two cells in a basic network; the first has a state of 1, while the other varies between 0–5. The figure shows prominent local minima at 1, 2, 3, 4 and so on, and at the inverse ratios (.5, .33, and so on). These will be the equilibrium states of the second cell. Note the flatter minima at larger ratios.

A graph of the basic interaction (without sum cells) in a 3 cell net with the first two cells clamped to the values 2 and 1 would yield the same curve, since the first cell does not interact with the varying third cell. Introducing sum cells, however, gives a different curve as can be seen in Fig. 7b. A minimum is shown at 3 caused by the interaction of the sum of the first and second basic cells with the last cell (3 : 3 yielding a ratio of 1). The minimum at 3 being strengthened by the interaction of the first cell with the sum of the second two (2 : 4, yielding a ratio of 2). This interaction also results in a weaker minimum at 1.5 (3 : 1.5, a ratio of 2). With a left context of 2 : 1 : 1 the minimum at 3 almost disappears as in Fig. 7c. There is now a strong minimum at 2 because the sum cell—which combines the durations of the second and third cell—is also 2. The sum of the first three cells give rise to the minimum at 4. This clamping method thus gives a clear picture of the mechanisms involved in the complex interactions through a simplification of the process that assumes fixed values in most of the cells. The

same method can also be used to study the influence of the parameters of the interaction function. In Fig. 7d, which uses the same context as in Fig. 7c, the peak and decay parameters have been changed, showing the effect on the catch range.

If we now return to the more elaborate example shown in Fig. 6, we can study the behavior of the net using the clamping method. Fig. 8a shows the potential energy curves resulting from applying the clamping method to the middle note of the triplet and the final sixteenth note. It shows clearly that the different contexts in which they appear result in different curves and that both will be directed towards the appropriate values. Note the wide catch ranges that allow rather large deviations to be quantized correctly and the smoothness of the curves. This smoothness (the lack of small local minima in the curve) is a result of the large number of interactions (364 and 91 for the triplet and sixteenth notes, respectively), which combine additively to yield each point on the curve. When the clamping experiment is rerun with performance data as context, more complex curves result, with a smaller catch range and a greater flatness, which is shown in Fig. 8b. Nonetheless, the durations still converge towards the correct metrical values.

The position of local maxima in the energy curves constitute the boundaries between the categories into which the data will be quantized. As a result, precise predictions can now be made about the perceptual interpretation of rhythmical sequences with a range of experimentally adjusted durations. It is our intention to compare these predictions with the results of empirical studies.

Implementation

In simulating a connectionist network, the calculated change in the state of one cell can be effectuated immediately (*asynchronous update*), or can be delayed, effectuating the change of all interactions at once (*synchronous update*). For asynchronous updates, a random order of visiting cells is generally preferred. In Table 1, a simplified implementation of the quantization model is given in Common Lisp (Steele 1984), based on synchronous updates.

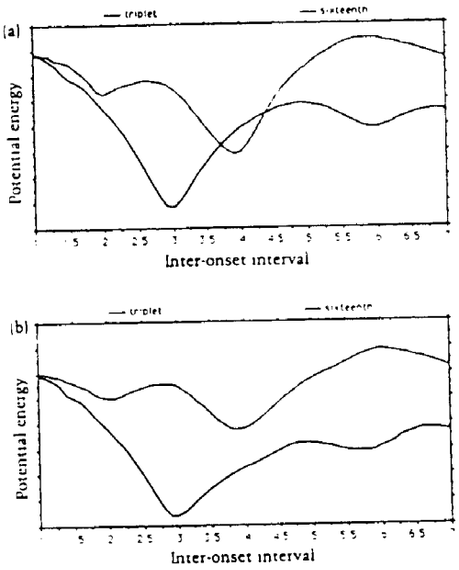


Fig. 8. Clamping curves of two notes in the context of an idealized complex rhythm (a). Clamping curves of two notes in the context of a performed complex rhythm (b).

The basic cells are represented as a vector of inter-onset intervals. The sum cells are not represented explicitly, but are recalculated, summing the represented interval of basic cells for each interaction. A macro is provided that implements the iteration over adjacent sum intervals. The described interaction function is the one we used for the Figs. 5 and 6. This simplified version requires the minimum inter-onset interval to be around 1. More elaborate versions run in Common Lisp and in C on stock hardware (Macintosh II and Atari ST series machines).

Further Research

The model we have presented needs high peak values to stabilize accurately. Because this results in smaller catch ranges, we are currently studying the automatic increasing of the peak parameter while

Table 1. Micro version of the connectionist quantizer in CommonLISP

```
;;; MICRO CONNECTIONIST QUANTIZER
;;; 1988 P.Desain and H.Honing

;;; Utilities

(defmacro for ((var &key (from 0) to) &body body)
  "Iterate body with var bound to successive values"
  (let ((to-var (gensym)))
    `(let ((,var ,from)(,to-var ,to))
       (loop ,(when to `(when (> ,var ,to-var) (return)))
             ,@body
             (incf ,var)))))

(defmacro max-index (vector)
  "Return index of last element in a vector"
  `(- (array-dimension ,vector 0) 1))

(defmacro zero-vector! (vector)
  "Set elements of a vector to zero"
  `(for (index :from 0 :to (max-index ,vector))
     (setf (aref ,vector index) 0.0)))

(defmacro incf-vector-scalar! (a b from to)
  "Increment elements in a range of a vector"
  `(for (index :from ,from :to ,to)
     (incf (aref ,a index) ,b)))

(defmacro incf-relative-vector-vector! (a b)
  "Increment elements of a vector proportionally"
  `(for (index :from 0 :to (max-index ,a))
     (incf (aref ,a index) (* (aref ,a index) (aref ,b index)))))

(defun print-vector (times vector &optional (stream t))
  "Print all elements of vector"
  (format stream "~%~3d: " times)
    (for (index :from 0 :to (max-index vector))
    (format stream "~2,1,5$ " (float (aref vector index)))))

;;; control structure for iteration over intervals

(defmacro with-all-intervals (vector (begin end sum) (start
finish) &body body)
  "Iterating over all intervals contained in [start,finish]"
  `(let (,sum)
     (for (,begin :from ,start :to ,finish)
       (setf ,sum 0.0)
       (for (,end :from ,begin :to ,finish)
         (incf ,sum (aref ,vector ,end))
         ,@body))))
```

(cont'd)

```
(defmacro with-intervals (vector (begin end sum) (start
finish) &body body)
  "Iterating over intervals"
  '(let ((,sum 0.0)(,begin ,start))
    (for (,end :from ,start :to ,finish)
      (incf ,sum (aref ,vector ,end))
      ,@body)))

(defmacro with-adjacent-intervals
          (vector (a-begin a-end b-begin b-end a-sum
          b-sum) &body body)
  "Iterating over interval pairs"
  '(let ((max-index (max-index ,vector)))
    (with-all-intervals ,vector (,a-begin ,a-end
,a-sum) (0 (1- max-index))
      (with-intervals ,vector (,b-begin ,b-end ,b-sum)
      ((1+ ,a-end) max-index)
        ,@body))))

;;; Main quantization procedures

(defun quantize! (durations &optional (peak 4)(decay -1))
  "Quantize data in durations vector"
  (let ((changes (make-array (length durations) :initial-
element 0.0)))
    (for (times :from 0)
      (print-vector times durations)
      (update! durations changes peak decay))))

(defun update! (durations changes peak decay)
  "Update all durations synchronously"
  (zero-vector! changes)
  (with-adjacent-intervals durations
    (a-begin a-end b-begin b-end a-sum b-sum)
    (let ((delta (if (> a-sum b-sum)
                     (delta (/ a-sum b-sum) peak decay)
                     (- (delta (/ b-sum a-sum) peak decay)))))
      (incf-vector-scalar! changes (/ delta a-sum) a-begin
      a-end)
      (incf-vector-scalar! changes (- (/ delta b-sum) b-begin
       b-end)))
  (incf-relative-vector-vector! durations changes))

(defun delta (ratio peak decay)
  "Return change of time interval"
  (let ((delta-ratio (interaction ratio peak decay)))
    (/ delta-ratio (+ 1 ratio delta-ratio))))
```

(cont'd)

```
(defun interaction (ratio peak decay)
  "Return change of ratio"
  (let ((position (1- (* 2 (- ratio (floor ratio)))))
        (goal (round ratio)))
    (* (- goal ratio)
       (abs (expt position peak))
       (expt goal decay))))
;;; usage examples
;;; minimum element in data should be larger than 1
;(quantize! (vector 1.1 2.0 2.9))
;(quantize! (vector 11.77 5.92 2.88 3.37 4.36 3.37 3.87
                   6.00 6.34 2.96 2.80 2.96 3.46 11.93))
```

For an updated version of this code, see the following addendum.

the network comes to rest. The dependency of the model on absolute time and absolute tempi is still an open question. The most difficult rhythmic cases for this model are: (1) those that involve additive durations that emerge when rests and tied notes occur in the data and (2) divisive rhythms, such as when a quintuplet is adjacent to a triplet. Our aim is to be able to characterize exactly the limits of the model and to evaluate the computational requirements and the psychological plausibility of the results. A further aim is to develop a robust technical tool for real-time quantization using a process model. Tempo tracking is then an absolute necessity.

Conclusion

We consider the compound model presented here to be promising. In difficult cases the system undergoes a graceful degradation instead of a sudden breakdown: that is, the range in which rhythms are caught and quantized correctly becomes more and more limited. However, it is a paradoxical problem with connectionist models that their adaptability means that even a rough first implementation, with obvious bugs, may exhibit appropriate behavior. In order to increase an understanding of the process involved, it is necessary to develop specialized tools for diagnosis and investigation. The clamping method described here seems to have considerable potential, and we are confident that further tools of a similar sort will develop as connectionist modeling gathers momentum.

Acknowledgments

We would like to thank Dirk-Jan Povel, Steve McAdams, Marco Stroppa, the reviewers of *Computer Music Journal*, and especially Eric Clarke and Klaus de Rijk for their help in this research and their comments on the first version of this paper.

References

Bharucha, J. J. 1987. "Music Cognition and Perceptual Facilitation: A Connectionist Framework." *Music Perception* 5(1): 1–30.

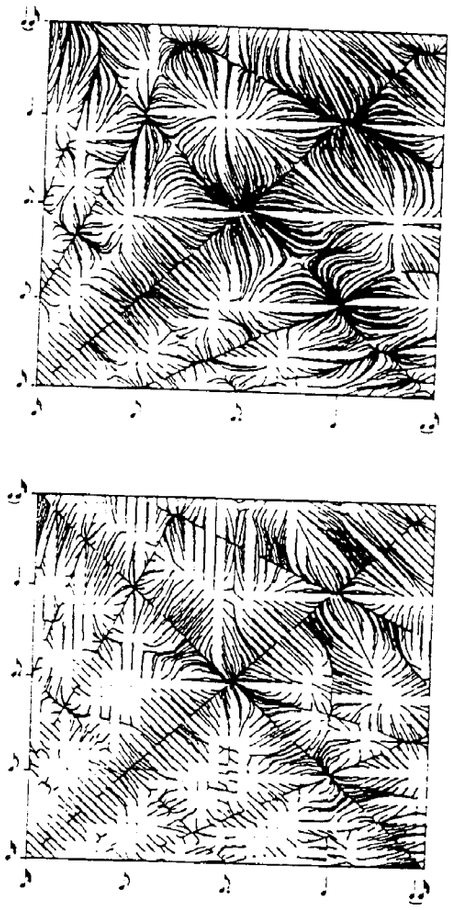
Chowning, J., et al. 1984. "Intelligent Systems for the Analysis of Digitized Acoustical Signals." *CCRMA Report* STAN-M-15.

Clarke, E. 1987a. "Levels of Structure in the Organization of Musical Time." *Contemporary Music Review* 2: 212–238.

Clarke, E. 1987b. "Categorical Rhythm Perception: An Ecological Perspective." In A. Gabrielsson, ed. *Action and Perception in Rhythm and Music*. Stockholm: Royal Swedish Academy of Music. No. 55: 19–33.

Dannenberg, R. B., and B. Mont-Reynaud. 1987. "An On-

Fig. 9. Trajectories in state space of a rhythm of three notes adding up to 3/4. The peak parameter is set to 2 for the upper plot and 6 for the lower.



Addendum

Peter Desain, Henkjan Honing, and Klaus de Rijk

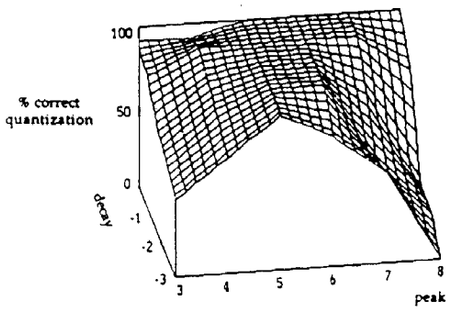
The design of special tools and methods to study the time-quantization network is of great importance, allowing us to explain and predict behavior for particular data, to examine the influence of the parameters on network performance, etc. The clamping method described earlier is one of these tools. A second method visualizes the state space of the system by only taking rhythms of three inter-onset intervals into account. The three degrees of freedom are mapped to two dimensions by normalizing the total length of the rhythm. Each point (x, y) represents a rhythm of three inter-onset intervals x; y; $1 - x - y$ in a net of interacting cells. Drawing the rhythm after each interation yields a trajectory toward a stable point in this space: the quantized version of the three intervals.

Plotting the trajectories of different rhythms exhibits the behavior of the network and the stable attractor points in this two dimensional space. They are positioned on straight lines that represent rhythms with an integer ratio of two durations or their sums ($x = y$; $x + y = z$, where z is the third interval length; $2x = y$; etc.). Fig. 9 shows this state space diagram for three intervals adding up to 3/4s with a variety of trajectories traced on it. One can see relatively large areas of attraction around the simple rhythms and relatively small areas around more complex rhythms. These so-called basins of attraction depend on the parameters of the interaction function; when the peak parameter is set to a higher value (see Fig. 9b), more basins of attraction around complex rhythms appear.

Diagrams such as Fig. 9 can form the basis for experiments to test the validity of the connectionist quantizing method as a cognitive model for rhythm perception. For example, we can plot the analogous diagram for human listeners performing a categorical perception experiment on part of the rhythm space and compare it with the output of the quantizer method. The results can be used to adjust the interval-interaction function of the model to more closely match human performance.

A third method amounts to a systematic explora-

Fig. 10. Mapping of the parameter space to the number of correct quantizations of a set of 50 rhythms.



tion of the space of all possible parameter settings. A mapping can be made from this space to the number of correct quantizations of a set of performances. Fig. 10 shows this mapping for a set of about 50 relatively simple rhythms, varying in length from 3 to 14 inter-onset intervals, performed by a musical expert. In this way, we defined implicitly what a "correct" quantization is. The vertical axis shows the percentage of correct quantizations of the system, and the other axes show the parameters' peak and decay. This visualization brings out specific characteristics of the model. First, it shows the model's sensitivity to its parameters. Often connectionist models behave badly in this respect, needing specific parameter settings for different problems. But Fig. 10 shows the system behaves quite well with respect to parameter sensitivity. The surface between a peak value of 4 and 6 and a decay value between 0 and -2 is almost flat. Second, it shows that the two parameters are more or less independent. A decay value between 0 and -1 is most successful, fairly independent of the peak parameter.

Furthermore, families of rhythms with particular characteristics (e.g., rhythms that change meter, syncopated rhythms, rhythms with swing, sloppy performances of rhythms) could be made and tested, yielding insights into the limitations of the model for these specific types of rhythms and the musical and cognitive interpretations of the parameters. We will explore these issues in future research.

Still, the best understanding of such a complex system arises from a mathematical description through which one can search for analytical solutions, prove convergence and stability properties, etc. The present state of progress on a mathematical description is given below, but much remains to be done.

Mathematical Model

Suppose a rhythm is given by a vector x of durations x_i with $1 \le i \le N$. At each update, a new duration vector is computed by

$$x^* = x + D(x)$$

where D in this case is a kind of update function. With a certain initial vector x, we can compute a set of vectors, $x^*, x^{**}, \ldots$, hopefully approaching equilibrium—the quantized rhythm. To characterize D, we begin by decomposing it into an update of individual basic cells

$$x_i^* = x_i + D_i(x).$$

An interaction cell connected to cells with values a and b should perform an increment of their ratio given by the interaction function

$$\frac{a^*}{b^*} = \frac{a}{b} + F\left(\frac{a}{b}\right).$$

We convert this change of ratio to a change of time interval $\Delta(a,b)$ under the constraint that the sum of the intervals stays the same:

$$a^* + b^* = a + b$$

$$a^* = a + \Delta(a,b)$$

$$b^* = b - \Delta(a,b).$$

This results in the definition of the change effectuated by an interaction cell:

$$\Delta(a,b) = b\frac{F\left(\frac{a}{b}\right)}{1 + \frac{a}{b} + F\left(\frac{a}{b}\right)}.$$

In a basic net, each basic cell (except the left- and rightmost cell) is connected to two interaction cells (see Fig. 2). Their change is computed by summing the change from each interaction:

$$D_i(x) = \Delta(x_i, x_{i+1}) - \Delta(x_{i-1}, x_i).$$

This describes the complete behavior of the basic network. In the compound network, the value of the sum cells is defined as

$$S_{pq} = \sum_{i=p}^{q} x_i, \qquad 1 \le p \le q \le N.$$

Suppose a sum cell S_{pq} is changed by an update function D_{pq} as follows:

$$S^*_{pq} = S_{pq} + D_{pq}(x).$$

A sum cell S_{pq} is interacting with a number of sum cells on the right ($S_{q+1,\cdot}$) and a number of sum cells on the left ($S_{\cdot,p-1}$), yielding the following definition of D_{pq}:

$$D_{pq}(x) = \sum_{r=q+1}^{N} \Delta(S_{pq}, S_{q+1,r}) - \sum_{l=1}^{p-1} \Delta(S_{l,p-1}, S_{pq}).$$

Here, if $q = N$, the first term vanishes because there are no right neighbors. Likewise, if $p = 1$, the second term vanishes. The change of the sum cells is propagated proportionally to all the basic cells connected to it. In each basic cell the change from all connected sum cells is summed.

$$D_i(x) = \sum_{p=1}^{i} \sum_{q=i}^{N} D_{pq}(x) \frac{x_i}{S_{pq}}.$$

Summarizing the above and taking care of leftmost and rightmost intervals, gives

$$D_i(x) = \sum_{p=1}^{i} \sum_{q=i}^{N-1} \sum_{r=q+1}^{N} \Delta\left(\sum_{k=p}^{q} x_k, \sum_{k=q+1}^{r} x_k\right) \frac{x_i}{\sum_{k=p}^{q} x_k}$$

$$- \sum_{p=2}^{i} \sum_{q=i}^{N} \sum_{r=1}^{p-1} \Delta\left(\sum_{k=r}^{p-1} x_k, \sum_{k=p}^{q} x_k\right) \frac{x_i}{\sum_{k=p}^{q} x_k}$$

This describes the behavior of the compound model.

Until now we have assumed $a > b$ in the definition of $\Delta(a, b)$. We can make a modification to eliminate the need for this assumption, as follows:

$$\Delta(a,b) = h(a,b) \frac{F(g(a,b))}{1 + g(a,b) + F(G(a,b))}$$

where $h(a,b)$ and $g(a,b)$ are defined by

$$h(a,b) = \begin{cases} b & \text{if } a > b \\ -a & \text{otherwise} \end{cases}$$

$$g(a,b) = \begin{cases} \dfrac{a}{b} & \text{if } a > b \\ \dfrac{b}{a} & \text{otherwise.} \end{cases}$$

When we implemented these systems, the results were inaccurate or unstable because the change in large sum cells tended to swamp the influence of smaller, local interactions. Therefore we scaled the interaction with the inverse of the interval b. This gave a precedence to local interactions that worked well. Because we still want to refrain for the moment from modeling the dependence of quantization on absolute global tempo, which was introduced implicitly by this change, we normalized this scaling factor with the overall minimum duration. The factor can be incorporated in the definition of $h(a,b)$:

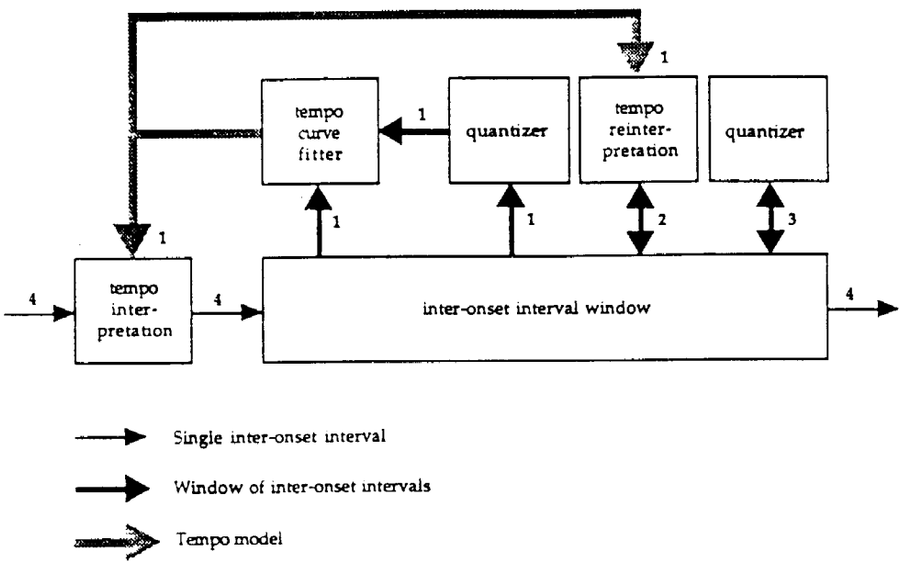
$$h(a,b) = \begin{cases} \min\limits_{1 \le i \le N} x_i & \text{if } a > b \\ -\min\limits_{1 \le i \le N} x_i & \text{otherwise.} \end{cases}$$

We would still like to characterize the final time-quantized equilibrium state for which

$$D_i(x) = 0.$$

In the simplified network, it can be proven that this condition only holds when all $\Delta(x_i, x_{i+1})$ are zero. This implies that the interaction function F has to be zero for all ratios, which in turn means that all ratios are integers or integers plus 0.5. When the sum cells are introduced, the system is much harder to analyze. All equilibrium points of the simplified system are also equilibrium points of the complete system, but there are many additional

Fig. 11. The process model of the connectionist quantizer.



equilibrium points as well. In fact it is not clear yet what exactly are the (stable) equilibrium points of the complete system.

Process Model and Tempo Tracking

A system that takes all of the temporal data available in a piece into consideration is, of course, not feasible when the aim is to develop a robust technical tool for near real-time quantization of longer pieces, nor is such an algorithm plausible as a cognitive model. Luckily, it proved quite simple to design a process version of the quantizer that operates upon a limited window of events. In this system, new inter-onset intervals shift into the window, and metrical durations shift out, being quantized on the way through. With such a model, tempo tracking becomes an absolute necessity since slow global tempo changes spanning a time lapse larger then the window cannot be operated upon nor corrected for.

The architecture we came up with makes use of two main modules, the quantizer and a tempo curve fitter (see Fig. 11). They work in mutual cooperation, communicating via a window of inter-onset intervals. In phase 1 (indicated by the numbered portions of the figure), the quantizer tries to quantize the data in the window. The result is passed, together with the original data, to the tempo curve fitter. This process tries to explain the difference between the quantized and original data as a global tempo change instead of as random fluctuations by fitting a third-order tempo curve to the quantized and original data. With the resulting tempo model, the data window is reinterpreted, and any consistent global change in tempo is removed from the original data in phase 2. The resulting se-

quence is now simpler for the quantizer module to operate upon. In phase 3 it is given a chance to remove the remaining deviations. Finally, in phase 4, a quantized inter-onset interval is shifted out of the window, and a new interval is shifted in, after being interpreted according to the expected tempo. Then the whole process is repeated on the shifted interval window.

As a result a rhythm can be quantized differently depending on the context established by the preceding data. This of course is the same as we would expect from human listeners. For the implementation of the curve fitter, special care was taken to use appropriate numerical methods, since numerical inaccuracies build up because of the feedback architecture used in the method and can result in oscillations.

Polyphony

The system described so far is unable to deal with inter-onset times that approach zero (as in chords or music with multiple voices). Although it may be possible to use other means to "clean" the data before quantizing it, such as rules for recognizing chord chunks, the general connectionist approach used in the quantizer seems a much better alternative. This is because the context can be taken into account when deciding if for example something is to be considered a chord with some spread, or a regular run of notes, or an arpeggio that has its own metrical structure. By introducing note durations, the system can distinguish between sequential and simultaneous inter-onset intervals (i.e., overlapping intervals indicate polyphony). We are currently experimenting with multiple interlocking networks that can handle polyphony. The preliminary results seem to be promising.

Main Characteristics of the System

In summary, the connectionist quantization system has three main characteristics: (1) It is context sensitive, with precedence of local context, as we demonstrated with the example in Fig. 6 and the results of the clamping method. (2) the system has no explicit musical knowledge. There is no preconceived knowledge of metrical or rhythmical structure used to quantize the performance data other than the notion of "integer ratios." All information is derived from the data itself; and (3) the system exhibits graceful degradation. When the quantizer breaks down in a complex situation, it is often able to maintain musical integrity and consistency at higher levels. The resulting error will only generate a local deformation of the score. Furthermore, this deformation will always be a simplification of the rhythm, not a very complex fragment as produced by some traditional systems (see Fig. 1). On the other hand, when more difficult rhythms are fed into the quantizer, they imply a smaller range of deviations than can be accurately captured by the system. Thus, they will be quantized correctly when performed with a higher accuracy or consistency. Such behavior could be another possible link to human cognitive performance.

Finally, a new version of the connectionist quantizer code using the loop macro is shown in Table 2. This version no longer requires the minimum inter-onset interval to be around one.

Reference

Desain, P., and H. Honing. 1991. "The Quantization Problem: Traditional and Connectionist Approaches." In M. Balaban, K. Ebcioglu, and O. Laske, eds. *Musical Intelligence*. Menlo Park, California: AAAI Press.

Table 2. A new micro version of the connectionist quantizer in Common Lisp with the loop macro

```
;;; MICRO CONNECTIONIST QUANTIZER
;;; (C)1990, Desain & Honing
;;; in Common Lisp (uses loop macro)

;;; utilities

(define-modify-macro multf (factor) *)
(define-modify-macro divf (factor) /)
(define-modify-macro zerof () (lambda(x) 0))

(defun print-state (time intervals)
  "Print elements of interval vector"
  (loop initially (format t "~%~2D: " time)
        for index below (length intervals)
        do (format t "~2,1,5$ " (aref intervals index))))

(defmacro with-adjacent-intervals
    (vector (a-begin a-end a-sum b-begin b-end b-sum) &body body)
  "Setup environment for each interaction of (sum-)intervals"
  `(loop with length = (length ,vector)
         for ,a-begin below (1- length)
         do (loop for ,a-end from ,a-begin below (1- length)
                  sum (aref ,vector ,a-end) into ,a-sum
                  do (loop with ,b-begin = (1+ ,a-end)
                           for ,b-end from ,b-begin below length
                           sum (aref ,vector ,b-end) into ,b-sum
                           do ,@body))))

;;; interaction function

(defun delta (a b minimum peak decay)
  "Return change for two time intervals"
  (let* ((inverted? (<= a b))
         (ratio (if inverted? (/ b a)(/ a b)))
         (delta-ratio (interaction ratio peak decay))
         (proportion (/ delta-ratio (+ 1 ratio delta-ratio))))
    (* minimum (if inverted? (- proportion) proportion))))

(defun interaction (ratio peak decay)
  "Return change of time interval ratio"
  (* (- (round ratio) ratio)
     (expt (abs (* 2 (- ratio (floor ratio) 0.5))) peak)
     (expt (round ratio) decay)))
```

```
;;; quantization procedures

(defun quantize (intervals &key (iterations 20) (peak 5) (decay -1))
  "Quantize data of inter-onset intervals"
  (let* ((length (length intervals))
         (changes (make-array length :initial-element 0.0))
         (minimum (loop for index below length
                        minimize (aref intervals index))))
    (loop for count to iterations
          do (print-state count intervals)
          (update intervals minimum changes peak decay))))

(defun update (intervals minimum changes peak decay)
  "Update all intervals synchronously"
  (with-adjacent-intervals intervals
    (a-begin a-end a-sum b-begin b-end b-sum)
    (let ((delta (delta a-sum b-sum minimum peak decay)))
      (propagate changes a-begin a-end (/ delta a-sum))
      (propagate changes b-begin b-end (- (/ delta b-sum)))))
  (enforce changes intervals))

(defun propagate (changes begin end change)
  "Derive changes of basic-intervals from sum-interval change"
  (loop for index from begin to end
        do (incf (aref changes index) change)))

(defun enforce (changes intervals)
  "Effectuate changes to intervals"
  (loop for index below (length intervals)
        do (multf (aref intervals index)
                  (1+ (aref changes index)))
           (zerof (aref changes index))))

;;; examples

;(quantize (vector 1.1 2.0 2.9))
;(quantize (vector 11.77 5.92 2.88 3.37 4.36 3.37 3.87 6.00 6.34
                  2.96 2.80 2.96 3.46 11.93))
```