5 Parametric morphing algorithm

This chapter will detail the first note sequence morphing algorithm that was developed and applied to the *LEMorpheus* software infrastructure: the parametric morphing algorithm. The parametric morphing algorithm was the first approach explored; both within this research and probably within note-level morphing as a whole (Mathews and Rosler 1969). Parametric morphing involves converting the note sequences of source and target into multiple continuous envelopes and combining the values of source and target envelopes, weighted on the morph index. The combined values are then converted back to note-sequence data.

Another name for this algorithm is 'interpolation' (Oppenheim 1995), however, the term 'parametric morphing' was chosen instead, because interpolation of the morph index occurs in all the morphing algorithms explored in this research. As well as this the important feature of parametric morphing is the fact that the source and target are *parameterised* into continuous envelopes, rather than the fact that the parameters are *interpolated*. That is, the choice of parameters to represent the music is a crucial decision, whereas it is obvious that interpolation of the parameters will proceed simply through weighted combination.

From a compositional standpoint, the parametric morphing algorithm described below is in the realm of mathematical experimentation; the **abstract** approach to composition. This is because the current low-level scheme does not afford the modelling of higher-level music composition approaches such as **heuristic** and **trial**. However, if the note sequences were parameterised into higher level musical constructs, for example, psychological descriptors such as "valence" (happy to sad) and "arousal" (intense to relaxed) (Wundt 1896), some sense of musical intention or musical goal could also be incorporated.

The musical results of the parametric morphing algorithm I developed were adequate when applied to very similar source and targets but unappealing for more difficult examples. This observation was made only through informal tests and, because of the particularly obvious lack in musical coherence, no effort was expended on formal tests. The reader can confirm this by listening to the examples in section 5.3. Despite this, some concepts for future improvement to the current parametric morphing algorithm have been developed and are included in section 5.4.

5.1 Overview

Parametric morphing involves the transformation of discrete note sequences from source and target into continuous parameter envelopes, the combination of these envelopes and the

generation of discrete notes from the combined envelopes. Advantages of the technique are that it operates in realtime, is deterministic and simple to understand. A disadvantage is that parametric morphing is not a common musical technique and the aesthetic results are unusual. Furthermore, it is currently quite difficult to imbue the process with a coherent sense of style. It is also difficult to deal with polyphonic music, as envelopes afford a monophonic sequence.

Section 5.2 describes the detailed workings of the algorithm, including the envelope representation, combination and note generation. Following this, the section on evaluation, 5.3, provides a number of simple musical examples that demonstrate various properties of the parametric morphing algorithm. The evaluation ends with a remake of Mathews and Rosler's original *The British Grenadiers* to *When Johnny Comes Marching Home* (1969) morph using the new parametric morphing algorithm.

5.2 Description

5.2.1 Envelope representation

Envelope information is stored in a list of nodes, each node consisting of a value for the parameter controlled by the envelope and the time at which that value exists. A continuous envelope is formed by the lines in-between each node.

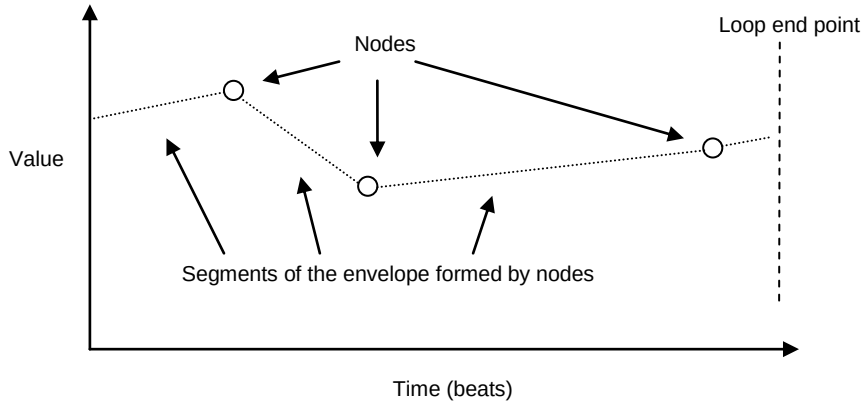

Figure 1 Example envelope representation

Because the note sequence is actually a loop, that is, the first note in the sequence is directly subsequent to the last note in the sequence, the last and first nodes will be connected by a line that wraps around from the end of the loop to the beginning. The key difference between an envelope and a discrete note sequence is that the envelope affords the generation of a value

from *any* point on the timeline, regardless of whether or not a note (or node) exists at that time. In the current system, there is a separate envelope created for the following dimensions of note information: **phase, inter-onset, duration, pitch, dynamic** and **polyphony**.[1]

Inter-onset envelope

The inter-onset envelope records the distance between the start times of notes. When generating the list of inter-onset nodes, each one is made to occur at the same point on the timeline as a particular note and the inter-onset value for that node is the distance from that note to the next. A zero gradient linear function used to connect each node. When two or more nodes with the same inter-onset value are in sequence, as with a regular beat, the latter nodes are redundant and are removed.

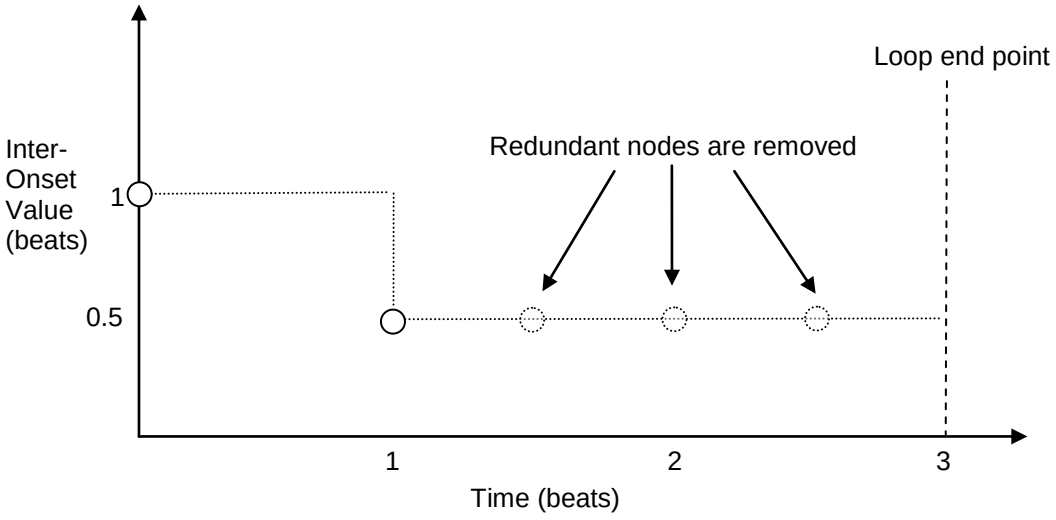

Figure 2 **Example of an inter-onset envelope representing a crotchet (quarter-notes) followed by four quavers (eighth-notes) spanning a three beat loop. The nodes representing the latter three quavers are redundant, as the zero gradient line continues regardless.**

Another alternative linear function is the "self-synchronising" function with gradient of -1 used by Mathews and Rosler (1969). My zero gradient technique requires constant updating during the rendering process (see section 5.2.2) and thus responds faster to realtime changes in the inter-

[1] To be able to generate notes, either inter-onset *or* phase is needed but both is not technically necessary.

onset envelope, but seems no more or less musically appealing than Mathews and Rosler's. For discussion of musical examples, see the section on evaluation (5.3) below.

Pitch envelope

Each of the other envelopes: duration, dynamic and pitch, have a node for each note, with the same value. In a similar way to inter-onset envelope, when a sequence of nodes has the same value, all nodes in the sequence except for the head are removed. The function used to connect these nodes is also a zero gradient linear function. For the pitch envelope this is suitable because, unless using a degree/scale representation such as *DePa* (see chapter four), it does not make sense to interpolate the values between pitches because non-tonal pitches will occur.

The current system can deal with polyphonic music but not in a musically meaningful fashion. An additional "polyphony" envelope stores the degree of polyphony and, if it is greater than one, a copy of the extra polyphonic notes (pitch, duration, dynamic) are added to the node. A note is considered "extra" if it is listed in the note sequence after another note that is on the same start time. There is no specific approach to the ordering of notes that share the same start time. Obvious improvements could be to label which of the two notes is more fundamental to the tonality, or model homophony through chordal tonal representations or model heterophony through multiple parallel streams. However, as most of the tests conducted with interpolation were monophonic, it was not necessary to develop the handling of polyphony beyond this basic level.

Phase envelope

The phase envelope records positive or negative time shifts in beats, relative to the start of the loop and, like inter-onset, uses zero gradient linear functions to connect the nodes. This is analogous to the way musicians play ahead or behind the beat, or the way that the beat in EDM is "turned around" (Butler 2006) by shifting the position of the kick drum to the offbeat.

For example, if the phase shift is 0.5 at a certain point, the data that is read from the other envelopes (such as inter-onset and pitch) at that point will be taken from the point directly previous by 0.5. Conversely, a shift of -0.5 means the data will be read from the other envelopes at a point ahead by 0.5.

If l is the length of the loop, p is the current phase offset, ranging from $-\frac{1}{2}l$ to $\frac{1}{2}l$, and t is the current time that is being shifted, ranging from 0 to l, the phase shift function, ph, is:

$$ph(t) = (t - p) \bmod l$$

Equation 1 Phase shift

The $\bmod l$ is required to wrap $(t - p)$ from the range $-\frac{1}{2}l$ to $1\frac{1}{2}l$, to the range 0 to l. [2]

Currently, the phase envelope is given only a single node during the conversion of notes to envelopes. This is calculated as the time interval of the first node from the start. If the first node occurs more than halfway through the loop, the phase offset is recorded a negative interval from the end of the loop.

Phase was included because of its wide use as a transformational technique in the composition and improvisation of melody; however, considering that only one node is being extracted, phase could be applied to greater effect than at present. Algorithms for estimating phase shifts from surface note sequences would enable multiple phase nodes to be extracted and these are explained in more detail as a possible extension in section 5.4.

Duration and dynamic envelope

For duration and dynamic, zero gradient, linear functions were used; however the capacity for variable gradient linear interpolation also exists. Linear interpolation of duration and dynamic reflects musical practice more accurately, for example, if a musician had to play a sequence between a known quiet note and a loud one, we could expect that, in the standard situation, these notes would gradually go from being soft to loud. Despite this, duration and dynamic are of less fundamental musical significance for most styles than pitch and rhythm (Krumhansl 2000) and the research shifted from parametric morphing to probabilistic morphing (chapter six) before any successful algorithms for parametric morphing of only pitch and inter-onset were invented. Future research in parametric morphing will utilise the capacity for variable gradient linear interpolation that exists in the software.

Converting note sequence into envelopes

The process used to generate the envelopes for inter-onset, pitch, phase, duration and dynamic from note sequence is described in pseudocode below:

---

[2] The *modulo* $(\bmod)$ mathematical function should not be confused with the *remainder* function of programming languages such as Java (where it is %). *Modulo* deals with negative numbers the same way time is wound back on a clock-face, whereas *remainder* deals with negative numbers in the same way as positive numbers, but with a negative valence.

```
// Inputs:
I       An array of notes which have: onset, pitch, duration, dynamic.  For example, the
        pitch of the first note in the array is I[0].pitch.
s       the loop length, or scope, in beats. For example, 4 beats or 8 beats
// Outputs:
E       A collection of envelopes for each dimension. For example, E.pitch is the pitch
        envelope, derived from the note pitches of I.  Each envelope is an array of
        nodes. Each node has time, t, and value, v.  For example, the time that the first
        pitch node occurs would be E.pitch[0].t and the pitch value would be
        E.pitch[0].v. For any envelope, add() appends a node to the end, given position
        and value.
// Global functions:
ADD-ALL
        creates a node for each of the envelopes for pitch, duration and dynamic from a
        given note. The first argument to ADD-ALL is the envelope object (E) that the
        nodes will be added to, the second argument is the position of the nodes, and the
        third argument is the note from which the node values are to be taken.

NOTES-TO-ENVELOPES(note array I, double s) returns E {
        QUICK-SORT(I) // sort according to onset, from first to last

        // phase offset
        IF(I[0].onset <= s/2) {  // if it is less than half-way
                E.phase.add(0, I[0].onset) // phase offset will be positive
        } ELSE-IF (I[0].onset > s/2) { // if it is more than half-way,
                E.phase[0].add(0, s - I[0].onset) // it will be negative
        }

        pno = 0         // previous note onset
        cno = 0         // current note onset
        i = 0           // a counter

        FOR(; i + 1 < LENGTH(I); i++) {
                pno = cno;      // update
                cno = I[i+1].onset


                posi = MODULO(pno - E.phase[0].v, s) // find the current position

                IF(cno == pno) { // if the current note onset is same as previous
                        E.polyphony.add(posi, COPY(I[i]))
                } else { // otherwise, it will be a new onset
                        E.interoffset.add(posi, cno - pno)
                        ADD-ALL(E, posi, I[i])
                }
        }

        pno = I[i].onset
        cno = I[0].onset + sco
        posi = (pno - E.phase[0].v)%s

        E.interoffset.add(posi, cno - pno)
        ADD-ALL(E, posi, I[i]);
}
```

Figure 3 Pseudocode for the algorithm that converts notes into envelopes

5.2.2 Envelope combination

As part of the morphing process, the envelopes from source and target are combined, and weighted on the morph index. This is essentially linear interpolation:

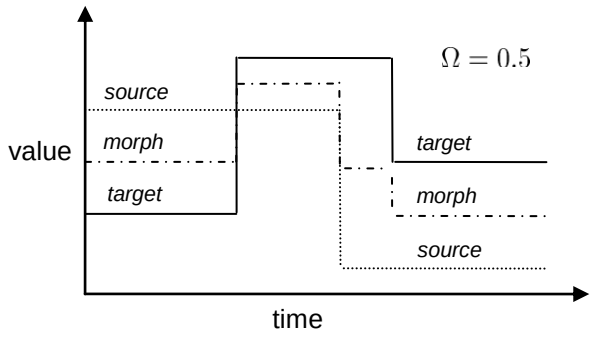

Figure 4 **An example of weighted combination of a source (round dots) and target (solid line) envelope, to produce a morph envelope (dash-dot). The morph index is 0.5, which is why the morph envelope is always exactly half way between the source and target.**

Let Ω be the morph index, which ranges between 0 and 1. Let t be the current time at any beat greater or equal to the start: $t > 0$. Let $S_o(t)$ provide the inter-onset value from the source inter-onset envelope at the given t. Let $T_o(t)$ be the same but for the target inter-onset envelope. Let $M_o(t)$ provide the morphed inter-onset value at t. $M_o(t)$ is derived from a combination of the source and target inter-onset envelopes, weighted on the morph index: $M_o(t) = (1 - \Omega)S_o(t) + \Omega T_o(t)$. The morphing of the other envelopes – pitch, phase, duration and dynamic – operate in exactly the same way as inter-onset.

5.2.3 Envelope playback

Playing notes from the morphed envelopes in realtime requires the playback algorithm to judge when to create a note and what values for pitch, duration and dynamic to give it. In realtime, an envelope playback routine called the **play cycle** is repeated at regularly timed intervals, usually at a quarter of a beat. The current time since playback started, in beats, is updated every cycle and passed to the envelope playback routine of the parametric morphing algorithm. During any particular play cycle, the envelope playback routine may or may not create a note and send it out. This depends primarily on the value of the morphed inter-onset envelope and the morphed phase envelope.

Interpolating loop length for the morphed envelopes

The loop length for the morphed envelopes, let it be l, is interpolated from the source and target loop length, through a combination of the loop length of the source and the loop length of the target, weighted on the morph index. That is, if S_l is the loop length of the source, T_l is the loop length of the target and Ω is the morph index, the loop length is derived thus:

$$l = (\Omega - 1)S_l + \Omega T_l$$

Equation 2 **The length of the loop for the morphed envelopes, derived through weighted combination of source and target loop lengths**

The source and target loop lengths are defined by the user directly.

Phase shifting the current time

As mentioned, the value in the phase envelope represents a shift ahead or behind the beat by a certain amount. Therefore, before any information from the other envelopes (inter-onset, pitch and so on) can be read, the current time must be shifted by the current value in the phase envelope. Let t be the current time, where $t \geq 0$ and let u be the current time shifted by the current value of the phase enveloped.

This is analogous to shifting the play head (u) forward or back while reading from a tape reel (envelopes other than the phase envelope). The precise position of this primary play head (u) is controlled by a separate tape reel (the phase envelope) and separate play head (t) that is not shifted.

Let $M_{ph}(t)$ be the function that returns the morphed phase envelope, that is, the weighted combination of the phase envelopes of source and target. The range of values for phase remains $-\frac{1}{2}l$ to $\frac{1}{2}l$.

Recalling that l is the length of the loop, t is the current time since playback and $M_{ph}(t)$ is the function that returns the morphed phase envelope, the phase shifted time, u, is thus:

$$u = (t - M_{ph}(t)) \bmod l$$

Equation 3 **The current time, phase shifted and bounded within the loop length.**

Determining when to play a note: in theory

It is clear that, optimally, notes should be generated when the average morphed inter-onset envelope value since the most recent note that was played is equal to the time that has lapsed since the most recent note was played. In this way, the actual inter-onset between the most recent note played and the new note would be equal to the average value in the morphed inter-onset envelope over the same period.

Let the average level of the inter-onset envelope since the most recent note be a. Let the time since the most recent note was played be v. The condition for playing a note in any given play cycle is therefore:

$$a = v$$

Equation 4 **Condition for playing a note during any given play cycle. a is the average value in the inter-onset envelope since the most recent note played. v is the actual interval between the most recent note played and the current position.**

This begs the question: how is a calculated? Let t be the current time, bounded within the loop length, l, such that $0 \leq t \geq l$. Let the function that returns the value of the morphed inter-onset envelope at any particular time be $M_o(t)$. Because this is a continuous function, rather than a set of discrete points, the average, a, must be calculated through the integral of the function (the area underneath it) from the most recent note to the current time, divided by the period of that integral, v. This way, the length of time spent at any particular value in the continuous function is weighted appropriately into the average.

For the case where the onset of the most recent note played requires no bounding, that is, when $0 < t - v$, we have:

$$a = \frac{\int_{t-v}^{t} M_o(t)dt}{v} \quad \text{for } 0 \leq t - v$$

Equation 5 **Determining a, the average value in the inter-onset envelope since the last note. $M_o(t)$ is a function that returns the value in the inter-onset envelope at particular points in time. t is the current time, bounded within the loop length. v is the time interval between the current position and the most recent note that has been played. This is only for when $0 < t - v$.**

For the situations where $0 > t - v$, two spans will be needed, one for 0 to t and another for $(t - v) \bmod l$ to l, recalling that l is the loop length. That is:

$$a = \frac{\int_0^t M_o(t)dt + \int_{(t-v) \bmod l}^{l} M_o(t)dt}{v} \quad \text{for } 0 > t - v$$

Equation 6 **Determining a, as per the previous equation, but in situations where $0 > t - v$. l is the length of the loop.**

Determining when to play a note: in practice

An example of how note generation relates to the inter-onset envelope can be shown diagrammatically (Figure 5). Each **play cycle**, the area under the inter-onset envelope spanned by that cycle is accumulated and the distance since the last note was played is incremented. When checking to see if a note should be generated, the accumulated area is divided by the time that has lapsed since the last note was generated in order to find the average inter-onset value over that period, as per Equation 5 and Equation 6. If the time that has lapsed equals or exceeds this value, a note is generated. The relationship between the area under the inter-onset envelope, the time interval from the previous note and the point at which notes should be generated is made clear in Figure 5 (below).

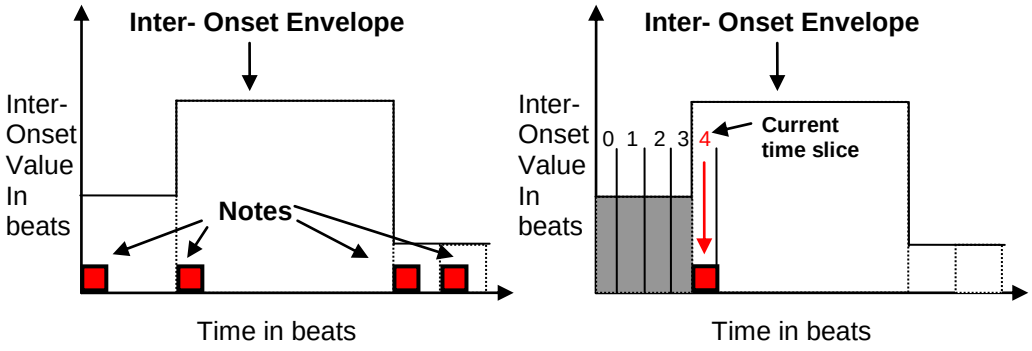

Figure 5 **Top: an inter-onset envelope (dotted line) with the notes (red) that were used to create it. Bottom: Generating a note from the inter-onset envelope – on the fourth play cycle, the area under the inter-onset envelope since the last note was generated (shown by the grey) will be equal to the distance from the previous note to the current position squared, thereby generating a note (shown by the red arrow). That is, combining Equation 4 and 5, $v^2 = \int_{t-v}^{t} M_o(t)\ dt$.**

When a note is generated, it is initialised with values in the pitch, duration and dynamic envelopes at the time of the current play cycle. After it has been generated, a user defined

switch controls whether the accumulated area is reset to 0, or if it has the square of the distance between the previous note and the current position removed, leaving a remainder.

Another user defined parameter controls the incidence of note generation by slightly offsetting the accumulated area on a range from -0.2 to 0.2. Increasing this parameter will increase the number of notes being generated and decreasing it will decrease the number of notes generated.

During the morph, the result of envelope combination might change rapidly, meaning that the accumulated area at one point is not necessarily what it would be if playback had started at that particular morph index. Because of this, the user can specify an alternative option: to recalculate the accumulated area from simulated playback at each morph index value, rather than updating the area and resetting it when notes are generated. This way, a complete note sequence with a constant morph index can be created for each possible position of the morph index. However, because the morph index usually changes over time before any of these complete sequences can be heard, the listener usually only hears a "window" into each one.

Direct interpolation in absolute MIDI pitch space often yields notes that are foreign to the tonality of both the source and the target. In order to counteract this, the user can choose to constrain the interpolated pitch to the set of pitch-classes that are used by source and target. This involves searching the set and choosing the pitch-class that is closest to the interpolated pitch. When the interpolated pitch is halfway in between two pitch-classes from the set, the statistical occurrence of pitch-classes is considered and the more common is selected. When creating the final pitch from the pitch-class and the octave of the original pitch, the pitch can sometimes be shifted by an octave, as when the interpolated pitch is above the highest pitch-class in the set and closer in circular pitch-class space to the lowest pitch-class. This is easily overcome by comparing the distance of the unconstrained interpolated pitch to the new pitch-class at the octave one above, the octave one below, and the original octave, to find the closest.

Often, slight changes in the inter-onset envelope between the source and target will effectively put the music out of phase and adopt a rate of play that is unusual for the time signature. In order to counteract this, another user option is to reset the accumulated area at the start of each loop. This maintains a degree of rhythmic coherence, effectively removing the phase shifts and stopping rate changes short before they begin to imply another time signature.

The pseudo code for the morphing algorithm that combines the envelopes and renders them into notes is in the appendix (9.5.1).

5.3 Informal Evaluation

The evaluation of the parametric morphing algorithm described above consisted only of informal testing and listening. This convinced me that, while the algorithm can generate coherent and smooth morphs when the source and target are similar, it fails with patterns that are very different. I considered this to be self-evident from the musical examples that were created with it and so I expended no effort on empirical tests for this algorithm (subsequent algorithms were tested more rigorously, as described in Chapters six and seven). Some of the musical examples that I generated using the parametric morphing algorithm will be discussed below, showcasing various properties of the algorithm that include phase-offset detection; pitch and duration interpolation; cancellation when morphing inverse melodic contours; monophonic music limitation; and onset resolution limitation. The final examples demonstrate interpolation between source and target music of various styles.

5.3.1 Phase offset

Phase offset is currently calculated from the onset of the first note in the loop. Figure 6 shows the transcript of a simple, four beat long pattern that is morphed into a pattern that is identical except for being shifted behind by one beat. The MIDI example is included (~5.1). Because the first note of the target pattern (bar five), starts on the second beat, the phase shift recorded for this pattern is -1. The source music has no phase shift. During the morph (bars two to four), it is easy to see that the phase is being interpolated from 0, to 0.25, 0.5 and 0.75 because the pattern is shifting behind the beat by 0.25 in each bar.

Figure 6 **Phase shift interpolation, made obvious by morphing from a pattern starting on the first beat to the same pattern starting on the second beat.**

The limitations in the way the phase envelope is currently calculated are made obvious in the example shown in Figure 7, and heard in this example (~5.2). As with the previous example, the target pattern is the same as the source, except it is shifted behind by 1 beat. Unlike the previous example, there are notes within the last beat of the source that wrap around to the beginning when the target is phase-shifted. Because the first note in the target is on the first beat, the value in the phase-offset envelope of the target is zero. As a result, interpolation occurs in pitch rather than phase, even though the two patterns are identical except for a phase-shift.

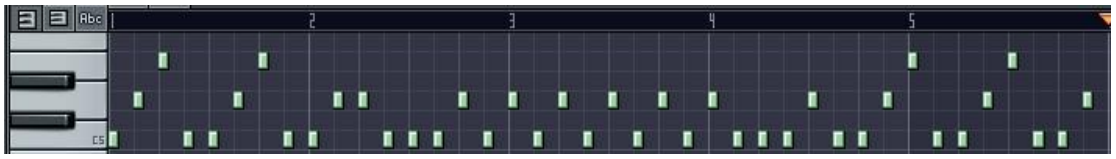

Figure 7 No phase shift detected - pitch interpolation is used.

5.3.2 Inverse melodic contours

The problem with phase offset is made even more apparent when we consider a source and target that are direct inversions of each other (~5.3). In this case, most of the pitches during the morph will be that of the pitch centroid, which is not particularly relevant or interesting to listen to, as shown in Figure 8.

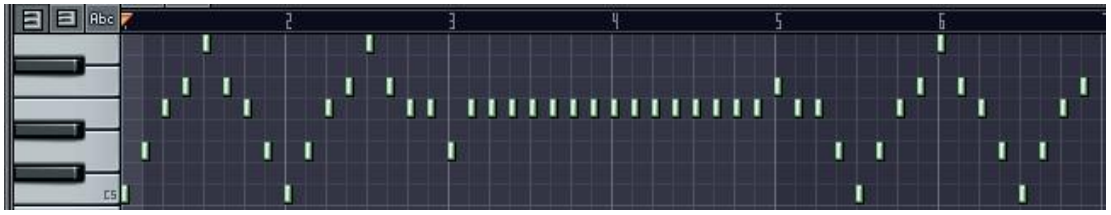

Figure 8 Interpolating between inverse melodic contours creates a mostly flat 'unmelodic' contour.

The details of a much more sophisticated approach to phase detection that might be implemented in the future will be presented below in 5.4.

5.3.3 Pitch and duration interpolation

As opposed to phase and inter-onset, the envelopes for pitch and duration are much more straightforward to generate. The linear interpolation of these two dimensions is clearly shown in the example transcribed in Figure 9 and audible in this example (~5.4). The pitch pattern in the target (bar six) is same as the source (bar one) except that it is transposed up a perfect fifth. The durations of notes in the source are increasingly short, whereas in the target they are increasingly long.

Figure 9 **Pitch and duration interpolation. In the source pattern (bar one), the notes start with a long duration and become shorter, while in the target (bar 6), the notes start short and get longer. The pitches of the target are the same as the source, but transposed up a perfect fifth.**

However, on listening, it becomes obvious that the linear interpolation of pitch is lacking in tonal musicality. To overcome this, tonal constraints have been implemented that shift the interpolated pitch to one of the pitches that occurred in the source and target, as shown in Figure 10 and heard in this example (~5.5). When constraining a pitch that is exactly half way between two of the allowable values, the value that has occurred over a greater time-span of the source and target is preferred. This is evident in Figure 10 where G or B could be equally chosen and B is preferred (see the notes at bar three, beat three and bar five, beat two). All of the B's occupied 2.25 beats, while the G's only occupied around 0.75 in the source and target collectively.

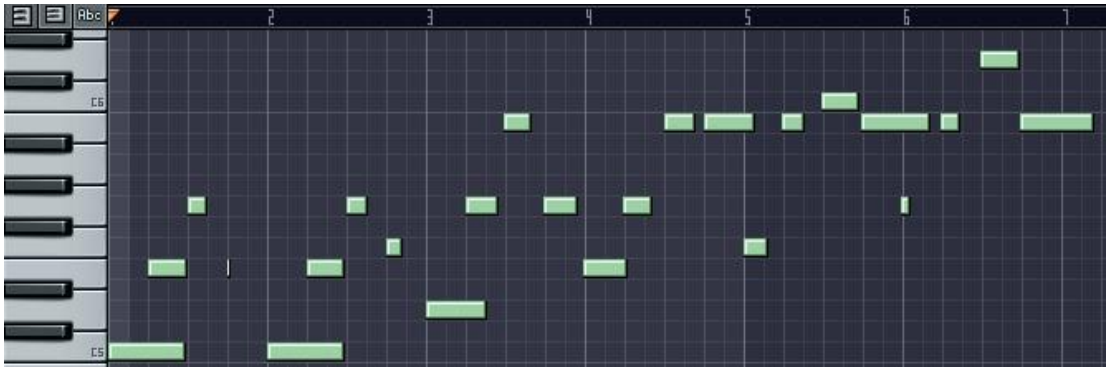

Figure 10 **Constraining the pitch to values that occur in source and target, favouring the more commonly occurring values.**

5.3.4 Music of different styles

When applied to various loops of different styles, parametric morphing had mixed results. If the two patterns share the same tonic and have related scales, it can work quite well, despite other

stylistic differences. To illustrate this, I have included an example that morphs between *Freŕe Jacque* a simple French children's song and a well-known Funk lick from *Chameleon*, by Herbie Hancock. This is transcribed in Figure 11) and is audible in this example (~5.6).

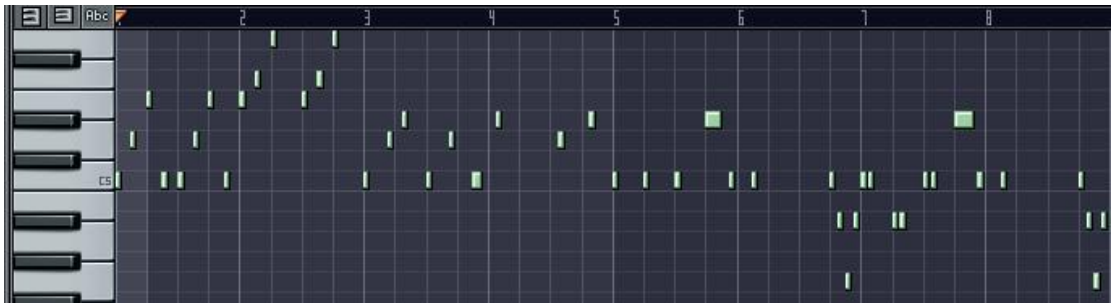

Figure 11 **Morphing between a two bar loop from *Freŕe Jacque* to a two bar loop from *Chameleon* (Herbie Hancock).**

The morphing example above is partially aided by the fact that the two patterns share the same tonic and have fairly similar scales (C Major and C Pentatonic Minor). When the target is transposed to the much more distant tonic of F#, the interpolation tends to make less sense, in terms of tonality, as shown in the example below (see Figure 12) and heard in this example (~5.7)

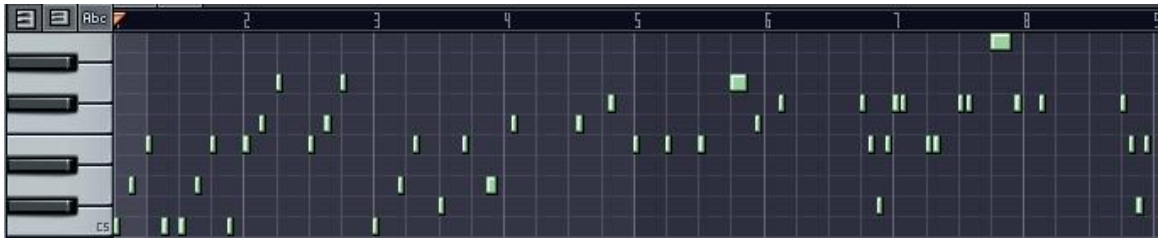

Figure 12 **Interpolation between different styles is harder when they are also in different keys, in this case, C Major and F# Pentatonic Minor. Source and target are two bars long, as in the previous example.**

The first parametric morphing algorithm that was developed by Mathews and Rosler (1969) morphed between *The British Grenadiers* to *When Johnny Comes Marching Home* and back. Although the music is more within the context of 'marching-song' than MEM, I applied the example to my own implementation of interpolation out of curiosity (listen to ~5.8). While it seems no better than the original, it is interesting to note how different they sound. Part of this is a technical issue – the current implementation of interpolation is limited to onset quantisation of 0.25 beats (because of the realtime implementation of quantise and the 0.25 beat play cycle resolution) which makes it difficult to morph to *When Johnny Comes Marching Home*, which is

based on triplets, that is, 0.33. This was overcome by using 0.25 beats as though they were 0.33 beats and slowing the tempo of the target down to 75% of the source. This introduced another problem, in that rather than morphing from one 16 beat loop to another 16 beat loop, the morph was now between a 16 beat loop and a 12 beat loop and similarities of melodic contour in relation to the start of the phrase would be lost.

These informal tests indicated some of the major limitations to a low-level parametric approach. Essentially, it is a suboptimal compromise between source and target. Rather than retaining the compatible musical elements of the source and target and splicing them together in a way that is reminiscent of both, the current implementation of parametric morphing tends to blur the elements together so that the result can hardly be recognised as originating from either, as is particularly apparent when the morph index is 0.5. These results have spurned some ideas for further developments that are outlined below, as well as the investigation of other algorithms that are discussed in Chapters six and seven.

5.4 Extensions

While the current implementation of parametric morphing is fast enough to operate in realtime and displays some elements of a logical musical progression, it lacks musical style and coherence. Some ideas for overcoming this are: alternative and varied envelope representations, interpolation in alternative pitch spaces, higher-level musical representations, rhythmic constraints and more detailed phase offset extraction.

5.4.1 Self-synchronising inter-onset function

The current envelope representation assumes a flat (zero) gradient for each segment in between nodes. This may be a suitable representation for pitch envelopes, however, duration and dynamic would be better served with a straight-line connection (variable gradient, depending on relative position of nodes), for reasons mentioned above. Note-generation could be simplified by using a “self-synchronising” style inter-onset envelope with a gradient of -1, as originally used by Mathews and Rosler (1969) and shown in Figure 14.

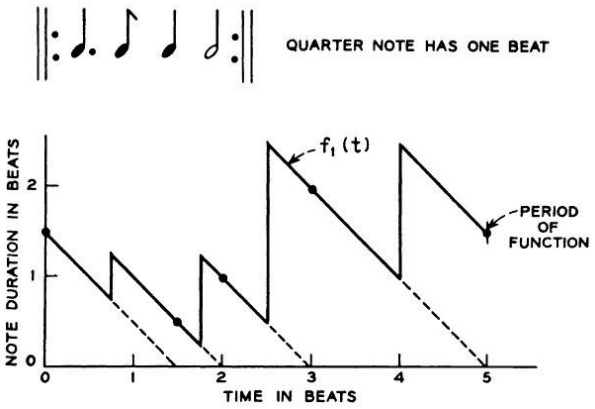

Figure 13 Mathews and Rosler's "self-synchronising function" (reprinted with permission)

However, this technique queries the envelope for a new value only when a new note is being generated which may sometimes leave out important details. For example, during the morph, if one inter-onset distance was particularly large and the values in the interpolated inter-onset envelope suddenly became small, these notes would be missed. An algorithm, such as the current one, that is updated each play cycle makes it possible for these changes to be heard as they occur. However, this may or may not be desirable, depending on the context. Therefore in future developments it would be interesting to compare the "self-synchronising" envelope to the current one in a variety of situations.

5.4.2 Higher-level musical constructs

Incorporating higher-level musical constructs, such as scale degree, scale and metre, could provide a consistent musical framework and be used to define a more coherent musical style. In the same way that interpolated pitches can be locked to known pitch-classes, a set of inter-onsets or onsets could be used as a constraint, increasing the rhythmic coherence by using only known onset intervals or onsets. Despite the gains in coherence, it would be especially difficult to generate music for sequences that have a limited variety of onset-intervals and onsets. As well as this, there is a possibility that too much of this kind of constraint could lead to a mostly static morph that changes suddenly at the halfway point, when the morph index is 0.5. The pitch and rhythm constraints could instead be expanded to include onset-intervals, onsets and pitch-classes that are drawn from music theory, rather than the source and target. For example, relevant modes and scales will have pitches that sound pleasing, even though they did not exist in the source or target.

However, constraints are ultimately limited by the fact that the process of interpolation occurs using a primitive representation, for example absolute pitch space, before the constraints are

applied. In this case, while the musical surface may exhibit certain stylistic features, the underlying processes do not have much musical stylistic influence because the representations being used are void of the bias of musical style or, conceived another way, low-level representations afford stylistically uninformed music that is often difficult to relate to. For example, it would make sense musically to interpolate pitch in a space that is a combination of the circle of fifths, circle of chroma and linear pitch, rather than just linear pitch, because these musical constructs are at the basis of much tonal music, which is a significant part of MEM. Similarly, pitches could be interpolated in scale degree space, although because the scales themselves exist more as discrete symbols than within a continuous space, interpolating deterministically between them is more difficult.

These kinds of representations have been successfully applied to the *TraSe (Transform-Search)* morphing algorithm and, to a lesser extent, the *Markov Morph* algorithm (discussed in chapters six and seven). In future research it would be interesting to see how much of a positive effect they have on interpolation.

5.4.3 Phase offset detection

In a similar way to the higher-level representations mentioned, the phase offset envelope provides an opportunity to extract from the music another dimension which, although not technically necessary, has musical implications. Phase-shifting the beat is a common technique in electronic dance music (Butler 2006) and other styles to renew interest through creation or elimination of syncopation and may be related to the ease with which note sequences can be shifted in sequencers. The current implementation of phase does not utilise the envelope representation effectively, only storing one node, the value of which is calculated from the onset of the first note in the sequence. This single phase value is needed so that the position of the first generated note is known and the interpolation of this value has a noticeable musical impact. However, often rhythmic changes are felt more as a phase-shift, rather than a change in inter-onset rate. Therefore another interesting area for future research of interpolation could be the extraction of a phase-shift envelope and the simultaneous changes to the inter-onset envelope that will be necessary.

Modifying the level of variation in the inter-onset envelope

A user defined parameter can be conceived that controls the relative level of variation in the inter-onset envelope. Let $O(t)$ be the function that takes the time, t, and returns the inter-onset value in the inter-onset envelope at that time. The values of both t and $O(t)$ occur within the range of the loop. That is, if l is the loop length, $0 \leq t, O(t) < l$.

Let h be the value of a horizontal line that runs through the exact centre of the inter-onset function, such that the area under $y = h$ is the same as the area under the inter-onset function:

$$h = \frac{1}{l} \int_0^l O(t)dt$$

Equation 7 The line that goes through the exact centre of the inter-onset envelope

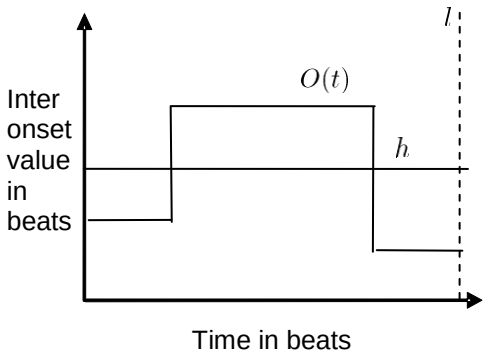

Figure 14 Diagram showing one possible example of the line h that goes through the exact centre of the inter-onset function $O(t)$. The loop length, l, is the period of the function.

Let v be the user defined parameter that controls the level of variation in the inter-onset envelope. The function that reduces or expands the variation in the inter-onset envelope is a:

$$a(O(t)) = h + (O(t) - h)v$$

Equation 8 A function that reduces or expands the variation in the inter-onset envelope. $v = 1$ recovers $O(t)$. $v = 0$, h.

A more natural technique perhaps could be to smooth the waveform and this will need to be examined in the future.

Phase adjustments to suit inter-onset modifications

With variation in the inter-onset envelope reduced or increased, it should be possible to reduce or increase the variation in the phase envelope in such a way that the original note onset pattern is maintained. The phase offset envelope should be created with a minimum number of nodes and a minimum amount of variation. If minimising variation without minimising the number of nodes, the function would include extremely short pulses of phase-shift which are not musically realistic. However, if the number of nodes are minimised in addition to minimising the variation,

the phase function will be smoother and thus possibly reflect the musical applications of phase-shift more naturally. How is such a phase function calculated? The following three diagrams (see Figures 16, 17 & 18) demonstrate what is required through example.

Firstly, the note onsets from the original inter-onset function $O(t)$ and the scaled inter-onset function $a(O(t))$ must be generated. In the example shown (Figure 15), the scaled inter-onset function has been scaled down to a completely flat line (the user defined variation, $v = 0$). Let the set of onsets generated by $O(t)$ be A and let the set of onsets generated by $a(O(t))$ be B. In Figure 15, the positions of the grey squares on the x-axis are an example of B while the positions of the white squares on the x-axis are an example of A.

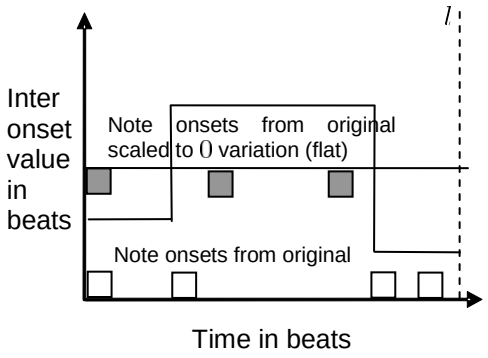

Figure 15 **Example of note onsets generated from a note onset envelope (white squares) compared to the note onsets generated by the same envelope that has been transformed into a flat line with no variation (grey squares).**

Having generated A and B, the second step is to analyze B to see how a phase shift function, let it be $s(t)$, might be applied to t so that $a(O(s(t) + t))$ will generate the original onsets A rather than B. To do this, each onset value from A or B that is absent in the other, $A \bigcup B - A \bigcap B$, will need to be considered as a potential point where a phase-shift node could be added to shift the onsets into synchrony. This is shown diagrammatically in Figure 16.

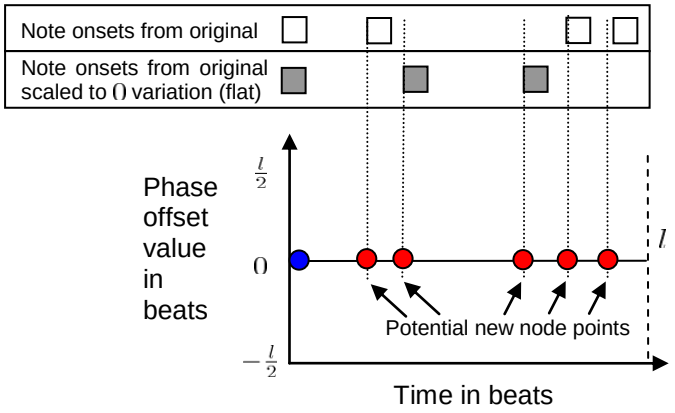

Figure 16 **With the objective of phase shifting the modified note onsets in such a way that results in the original note onsets, this diagram shows all of the potential points at which a new phase offset envelope node should be considered for the example note onsets given.**

Finally, each of the potential points where a node might be added to the phase offset envelope needs to be considered and, if applicable, a phase offset value needs to be assigned. For each potential point, the value of the phase offset will be the interval that "shifts" an onset in B onto the closest onset in A (see Figure 18).

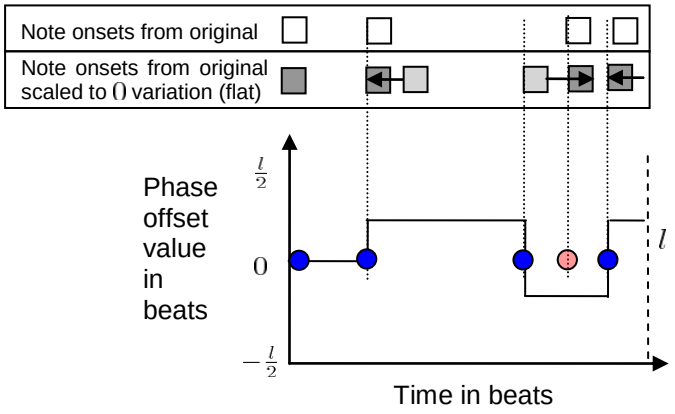

Figure 17 **The final phase offset envelope should be calculated to have the least variation, as with this example.**

5.5 Summary of parametric morphing algorithm

Parametric morphing involves the conversion of note sequences into multiple continuous envelopes, the weighted combination of source and target envelopes and the rendering of multiple continuous envelopes back into notes.

An envelope is used for each dimension of inter-onset, pitch, phase, duration and dynamic. When combining the source and target envelopes, the morph index is used to weight each of them, such that the morphed envelopes match the source initially and become closer to the target as the morph index increases.

For playback of envelopes, a note is created whenever the average value in the inter-onset envelope since the most recent note was played equals the time interval that has lapsed since the most recent note was played.

Informal evaluation of the parametric morphing algorithm has demonstrated the application of particular features, including the interpolation of phase offset, the cancellation of inverted melodic contours as well as interpolation of pitch and duration. Experimenting with music of different styles found that, although reasonable results occur when the source and target share crucial musical elements such as key, source and targets that are even moderately dissimilar, are much more dissatisfactory.

Potential extensions to parametric morphing include: a self-synchronising inter-onset function, in order to allow historical comparisons; higher-level musical constructs, such as scale degree, scale, key and metre; and phase offset detection, where the user could control the relative levels of variation in the inter-onset and phase envelopes.

From the detailed explanation of parametric morphing, the differences with other historical approaches become obvious, justifying the claims to novelty. Despite this, the parametric morphing algorithm described here fell short of an acceptable level of effectiveness in musical output. Although a number of improvements to the current parametric morphing algorithm were conceived, a decision was made to shift the research towards a fundamentally different approach that had not yet been investigated to any great extent: probabilistic morphing. The rationale for shifting the research in this direction was that probabilistic techniques offered random access to the source and target notes, thus increasing the contextual breadth available to the algorithm. As well as this, it appeared more relevant to broaden the approaches to compositional morphing, rather than deepening a single approach (parametric morphing) that was conceived many decades ago.