

A symbolic-numeric approach to quantization in music

Bernard BEL
CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
French Centre for Social Sciences and the Humanities (CSH)
2, Aurangzeb road, New Delhi 110 011
INDIA — email: bel@csh.delnet.ernet.in

Abstract

Bol Processor BP2 is a software environment for music composition in which text-oriented representations of sound events have been preferred to graphic or numeric descriptions.

Text representation requires several abstraction levels. The highest level ('prescriptive' notation) is completed with descriptions of the basic sound units combining numeric data (e.g. MIDI codes) and symbolic data (e.g. topological properties).

The representation of time is multilayered as well, with 'symbolic time' ('beats' on the score) using rational numbers at the highest level.

Using rational numbers provides absolute accuracy, but it may also yield unnecessarily complex phase-diagrams (tables of events ordered on symbolic time) when large integer ratios are involved. Quantization saves computation time and space by simplifying phase diagrams on the basis of a time mismatch tolerance supplied by the user.

This paper explains how quantization is performed by BP2 at the symbolic level itself rather than through arithmetic approximations.

Background

In the early 1980s, the author developed a computer system called the *Bol Processor* (i.e. 'BP') to study the improvisatory methods of North Indian tabla drum players, in collaboration with ethnomusicologist Jim Kippen. Designed for the portable Apple IIc, BP1 was able to analyse musical input (in the form of tabla *bols*, or onomatopoeic syllables) as well as generate new improvisations that musicians could assess (Kippen & Bel 1992, 1994).

The Bol Processor attracted interest from scholars and musicians alike. It was felt that the formal model embedded in it could be expanded to encompass more general musical structures, and in this form would be of some benefit as a tool for music composition. In the late eighties a new version (BP2*) was implemented on the Apple® Macintosh™ along with a sound-object editor allowing its interaction with MIDI devices. Syntactic extensions in BP2 grammars include remote-context rules, context-sensitive substitutions, dynamic rule weight assignment, programmed grammars, scripts and procedural inference control (Bel & Kippen 1992, Bel 1992b). Later on the program was allowed to run in background and respond to Apple Events, thereby co-operating with client applications such as data-bases.

BP2 allows a complete control of MIDI parameters, including the vectorised representation of continuous changes of pitch, modulation, etc. (Bel 1996a).

A crucial question in text-oriented representations of music is the mapping of *symbolic*, prescriptive (i.e. incomplete) data to the *numeric* and fully descriptive stream of instructions required by sound

* Bol Processor BP2 may be retrieved from <http://www.bp2.org>

processors. The potential of a symbolic-numeric approach such as the one implemented in BP2 will be illustrated in the problematic of time quantization.

Text-oriented symbolic representation of music

Musical scores in BP2 are **text** representations of arbitrarily complex musical structures. This format is attractive in several aspects:

1) Text lends itself to elementary find/replace operations that may be formalised with rewriting systems. For instance, a formal grammar may be used as a generative model for a unique musical piece or a set of related musical items. By combining declarative and procedural representations of the elaboration process (Bel 1992b), BP2 grammars lend themselves to **design-based** and **improvisational rule-based** compositional approaches (Laske 1989:48).

2) In a client-server environment, standard data-bases used for musical documentation may include musical data (scores) and processes (scripts and grammars) stored in text format. This information is transferred (via Apple Events) to BP2 which interprets it in background on an external MIDI device.

The need for a versatile prescriptive representation of music

Western music notation favoured a graphic representation of music which owes its simplicity to conventional quantizations of the main parameters: pitch and time. Lines on a classical musical score presuppose a twelve-tone division of octaves. In the same way, measure bars indicate a time interval equal to an integer number of occurrences of a conventional unit. Indeed, the refinements of microtonal intonation (*portamenti*, quartertones, *shrutis*...) may be represented more or less accurately with additional symbols, and other symbols have been designed for the description of unusual duration patterns. But then musical scores tend to become more **descriptive** than **prescriptive**. For instance, the notation of melodic or rhythmic patterns peculiar to a non-western musical system requires a level of intricacy discouraging musicians to transcribe their own music or manipulate its representation.

The lack of a suitable prescriptive representation of music may refrain musicians from using paper or computer support for composing music, thereby reducing many compositional ideas to arrangements of precomposed fragments and stereotyped musical gestures. Although 'creativity' may be perceived in antagonist ways by individuals belonging to different cultures, it often appears as the outcome of **explorative tasks**. In Laske's words (1993:211), the (modern) composer is committed to **possible**, rather than **existing**, musics.

In the domain of computer music, trade-offs between descriptive and prescriptive representations are manifold. Programs like Opcode *Finale* are specialised in common music notation completed with performance parameters pertaining to volume, velocity of attack and offsets from metronomic timings. MIDI music sequencers programs like *CuBase* are very versatile thanks to less dependence on conventional musical scores, but their representational model is poor in terms of structural descriptions. Algorithmic approaches in *Compose* (Ames 1989) or Zicarelli & Puckette's *Max* allow the description of musical pieces (in the MIDI environment) as an outcome of arithmetic calculus and logical decisions. In the case of *Max*, real-time MIDI input is also taken into consideration.

At the other end of the scale, programs like *Csound* generate sound tracks on the sole basis of their detailed acoustic description in an *orchestra* file combined with a *score* file containing numerically coded pitch and control information in *standard numeric score* format. This class of software/hardware environments (including also *Kyma*, *Axel*, *Syter*...) make no assumption on musical parameters: it is the responsibility of the composer to apply his/her theoretical constructs to the sound design.

Although *Compose*, *Max*, BP2 and other algorithmic software also make no assumption on music representations, their output must comply with a limited sound format such as MIDI. Further developments of BP2 may explore music descriptions beyond the limitations of MIDI, for instance *Csound* instruments or the MDPL environment (McMillen et al. 1994).

The MIDI format evolved from descriptions of simple actions on electronic keyboard instruments. It may nevertheless be viewed as a plain network exchange protocol in which, for instance, a NoteOn message on key 60, channel 1, will trigger an arbitrary process on a sound device. Users of MIDI samplers or drum machines are familiar with this broad interpretation of MIDI specifications. The output of a MIDI software device may therefore be described as a stream of numbers complying with basic rules, e.g. a NoteOn is expected to conclude on a NoteOff on the same key and channel.

Mapping symbolic descriptions to numeric ones

The discrepancy between a prescriptive (graphic or text) representation and a descriptive numeric output may be dealt with in different ways. *Compose* and *Max* work with numbers and direct designations of MIDI parameters. At the opposite, BP2 makes it possible to assign names and properties to arbitrary streams of MIDI messages and grant them an autonomous existence as ‘**sound-objects**’. Each sound-object name is in turn a ‘**terminal symbol**’ of the representation language (Bel 1996a-b). The alphabet of terminal symbols covers the significant ‘**musical gestures**’ on which the musical piece or a set of related musical items are based. The term ‘gesture’ has been inspired by early applications of the Bol Processor to studies of Indian drum music. Nevertheless it retained its pertinence in the late eighties when BP2 was used to control sound processors, such as *Syter*, in which arbitrary MIDI messages can be mapped to real-time processes.

For the sake of consistency, BP2 accepts ‘simple notes’ as predefined sound-objects containing a pair of NoteOn/NoteOff messages.

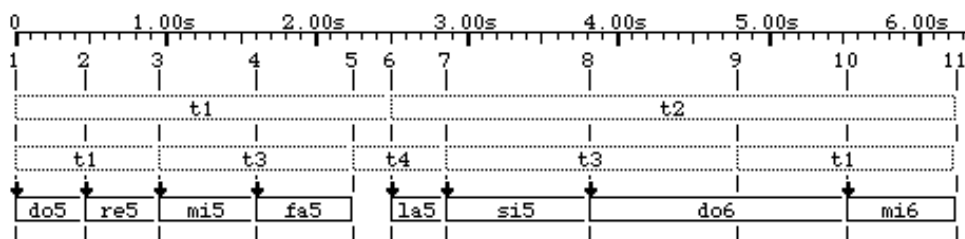
It is easy to figure out that a given sound-object (or musical gesture) may not be performed invariably in different occurrences. For instance, the timings of elementary events may depend on the current metronome setting, or the ‘local tempo’ in the absence of a regular pulse. In addition, the locations of sound-objects along the physical time axis may be adjusted according to the context. These changes are performed by BP2 on the basis of metrical and topological properties of sound-objects (Bel 1992a,1996a).

Time patterns

Regularity of metronomic beats is not a prerequisite for the performance of items produced with the help of BP2. Consider for instance the sequence of simple notes performed in ‘**smooth time**’ (no pulse, ‘*temps lisse*’ as per Boulez 1963:107):

do5 re5 mi5 fa5 - la5 si5 do6_ mi6

Symbol ‘-’ indicates a silence and ‘_’ the prolongation of the preceding sound-object. Instead of using a metronome, BP2 is instructed to derive physical durations from a particular arrangement of ‘**time-patterns**’. Each pattern is a preset integer ratio attached to a terminal symbol: ‘t1’, ‘t2’...‘t4’. This arrangement determines irregular ‘**time streaks**’ (numbered 1 to 11) as shown on the graphic score:



(A discussion of this example may be found in BP2 documentation page 48)

In the BP2 environment, vertical lines numbered 1 to 11 are called ‘**time streaks**’ and indicate ‘**symbolic durations**’ which may not be proportional to physical durations. In this example, the symbolic durations of ‘do5’ and ‘mi5’ are both one unit whereas their respective physical durations are roughly 480ms and 700ms.

Text representation in BP2 allows the description of polyphonic items thanks to a set of conventions known as ‘**polymetric representation**’ (Bel 1992a,1996b). Superimposed sequences of sound-objects are listed in an expression between curled brackets. The polymetric expression showing both sound-objects and time-patterns in the above example would be:

{t1 t2, t1 t3 t4 t3 t1, do5 re5 mi5 fa5 - la5 si5 do6_ mi6}

Time quantization

In common MIDI programs, time quantization stands for the reduction of time intervals to multiples of an elementary unit. The unit must be small enough to make roundings unnoticeable. A typical value is 1/64th of a quarter note. This unit may temporarily be set larger while entering music from the MIDI keyboard. For instance, 1/8th of a quarter note discards any interval smaller than 0.125 seconds if the metronom setting is 60.

Programs based on this simple principle yield a good time accuracy so long as time intervals are multiples of the reference unit. Complicated polyrhythmic effects (see example *infra*) may be blurred by this rudimentary quantization.

After introducing ‘period notation’ of sequences in BP2, the problem of accurate timings will be studied from a fresh viewpoint.

‘Period notation’ of sequences, polymetric expressions and phase diagrams

Let ‘a’, ‘b’... ‘i’ be terminal symbols designating sound-objects. The expression “a.b.c.ab.cd.ef.abc.def.ghi” is interpreted as a sequence of nine beats in which each of the first three beats contains a single time-object, the following three beats contain two objects, and the remaining ones three objects. Subdivisions of beats are equal. An almost equivalent representation (valid in BP2) would be:

/1 a b c/2 a b c d e f/3 a b c d e f g h i

Expression ‘/2’ (a **tempo marker**) indicates the beginning of ‘speed 2’. The initial ‘/1’ is the default speed and may therefore be omitted.

The rescaling of beats in a sequence is based on the same algorithm as that of simultaneous sequences in a polymetric expression. For example, the sequence “/1 abcde.fgh” is interpreted:

/3 a__ b__ c__ d__ e__ f____ g____ h____

Similarly, the polymetric expression “{abcde, fgh}” leads to the **‘phase diagram’**:

```
a  _ _ b  _ _ c  _ _ d  _ _ e  _ _
f  _ _ _ _ g  _ _ _ _ h  _ _ _ _
```

The concept of phase diagram is self-explanatory: sound-objects starting at the same **symbolic date** (such as ‘a’ and ‘f’ in the above example) must be placed in the same column. This diagram should not be confused with a graphic score on the physical time axis. Columns are regularly spaced on the diagram, but the resulting spacing of time streaks may be irregular (see the case of smooth time and time-patterns, *supra*). In addition, due to constraints impending to their properties, the locations and physical durations of sound-objects may not coincide with time streaks.

The quantization problem: a numeric approach

In BP2, all time calculations are performed on (long) integer ratios. This guarantees absolute time accuracy in the absence of cumulated rounding errors. Since a rounding may accidentally turn up when numbers are overflowing, the first motivation for implementing time quantization has been to discard complicated ratios. Another outcome of the quantization scheme is the saving of computation time and memory space.

The **quantization value** is the largest acceptable time mismatch in a given musical item. Declared by the user, this value must anyway be larger than the resolution of the MIDI driver (1 to 10 milliseconds). A typical value in the MIDI environment is 50ms.

Given sound-objects ‘a’, ‘b’, etc., consider for instance the musical item “a.bcd.efgh.ijklmno” performed with a metronome set at 60 beats per minute and 50ms quantization. To build a phase diagram for this item, BP2 calculates the lowest common multiple of 1, 3, 4 and 7. The result is 84. Sound-object ‘a’ covers the 84 leftmost columns of the phase diagram, then ‘b’, ‘c’ and ‘d’ cover $84/3 = 28$ columns, ‘e’, ‘f’, ‘g’, ‘h’ $84/4 = 21$ columns, and finally ‘i’, ‘j’, ‘k’, ‘l’, ‘m’, ‘n’, ‘o’ $84/7 = 12$ columns.

In the planned phase diagram, the time delay between two columns — or equivalently two time streaks — is $1000/84 = 11.9$ ms, less than the declared time quantization. A simplified phase diagram makes no

difference provided that the time delay between two columns remains smaller or equal to 50ms. The integer part of the ratio $50/11.9 = 4.2$ is called the **'quantization ratio'**. Using a quantization ratio of 4 yields a simplified phase diagram in which sound-object 'a' covers the 21 leftmost columns, 'b', 'c' and 'd' 7 columns, 'e', 'f', 'g' and 'h' 5 columns, and 'i', 'j', 'k', 'l', 'm', 'n', 'o' columns. The durations of 'e', 'f', 'g' and 'h' have become slightly smaller than in the absence of quantization, but the rounding is less than 50ms. Besides, this loss of accuracy does not change the timings and dates of the following seven sound-objects. In other words, rounding errors are not cumulated.

The symbolic-numeric solution for time quantization in BP2

The previous example was solved numerically. BP2 performs the same type of quantization by simplifying the symbolic representation. First, the quantization ratio is evaluated. In most cases, it is the integer part of the ratio between the quantization value and the delay between two time streaks in the expected phase diagram (see *supra*). If the item contains a sequence "going too fast" for the declared quantization, thereby meaning that the delay between two consecutive sound-events is less than the specified quantization, then the smallest delay is taken *in lieu* of the quantization for calculating the ratio. In this way, the sequentiality of sound-objects is preserved even if a large quantization value has been specified.

The quantization process will now be illustrated. Consider the polymetric expression

{do4 {re4 mi4, fa4 sol4 la4}, si4 do5}

which is interpreted as:

/6{do4_____ {re4_____ mi4_____, fa4___ sol4___ la4___ }, si4_____ do5_____ }

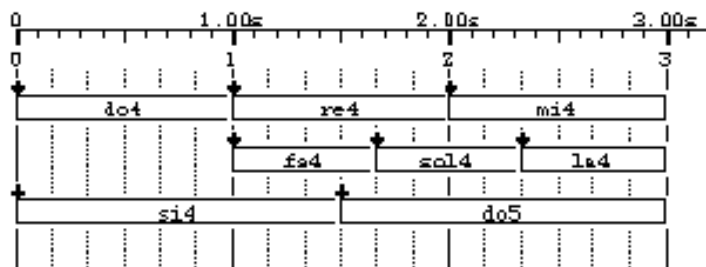
yielding the phase diagram

```

do4_ _ _ _ _ re4_ _ _ _ _ mi4_ _ _ _ _
- - - - - fa4_ _ _ _ _ sol4_ _ _ _ _ la4_ _ _ _ _
si4_ _ _ _ _ do5_ _ _ _ _

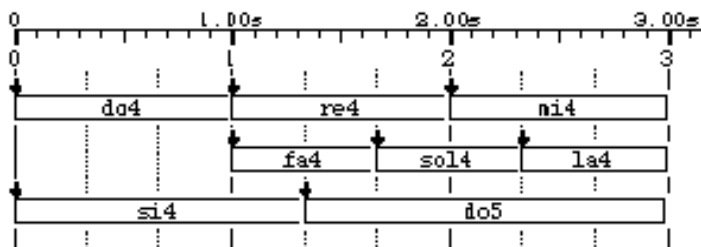
```

and the following graphic score with metronomic setting mm = 60:



The time delay between two consecutive time streaks is $1000/6 = 166$ ms. This is also the delay between the off-setting of "si4" and the on-setting of "sol4".

Let us now set the quantization value to 200ms. The new graphic score is:



The delay between the off-setting of 'si4' and the on-setting of 'sol4' has become 200ms, which makes no significant difference with the previous value (166ms) since both are less or equal to the quantization value. The same remark applies to the unequal durations of 'si4' and 'do5'.

It is easy to reconstruct the simplified phase diagram from this graphic score:

```

do4_ _ _ re4_ _ _ mi4_ _ _
- - - fa4_ _ _ sol4_ _ _ la4_ _ _
si4_ _ _ do5_ _ _ _ _

```

