

# Stepwise Tempo Changes in BpmDj

Van Belle Werner (werner.van.belle@gmail.com, werner@onlinux.be)

June 5, 2006

## Abstract

kbpm-play provides an algorithm that automatically changes the play-rate from its current speed back to normal speed. A first version of the software relied on a linear change of pace, which turned out to be slightly wrong. In this short article we describe what went wrong and how to do it right.

## 1 Defining Variables

We define a number of variables necessary to investigate the problem.

$a$  the current play speed of the song, expressed in samples/second.

$b$  the target play-speed, expressed in samples per second.

The values placed in these two variables will lie around the sample rate.

$r$  the sample rate of the original soundtrack. In BpmDj this is 44100 Hz.

We further assume a linear change of pace. E.g: every second the play-speed is updated.

$N$  defines the number of steps necessary to reach the final speed  $b$

$i$  the current step. This variable ranges from 0 (at which moment the speed is supposed to be  $a$ ) to  $N$  (at which moment the play-speed should be  $b$ ).

We now want to calculate the lplaying speed we need at step  $i$

$s_i$  the playing speed at step  $i$ .

## 2 Linear change

A first straightforward approach to determine  $s_i$  seems linear interpolation, as follows

$$s_i = \frac{i(b-a)}{N} + a$$

The main problem we observed with this approach was that the music seemed to step up (or down) with an uneven pace. It did *sound* wrong. A closer investigation reveals that simple interpolation will lead to a non linear frequency change. To demonstrate this closer, we will work with a song which has only one waveform: a pure sine with frequency  $f$ .

We can now determine the frequency of that note at step  $i$  (call it  $f_i$ ) through multiplying it by the current speed ( $s_i$ ) and division by its normal speed (the sample-rate  $r$ ).

$$f_i = \frac{f \cdot s_i}{r} \tag{1}$$

Since human beings perceive frequencies in a non equally spaced manner, we need to convert this frequency to a note in a (for this purpose) equi-temporal scale. Rising one octave amounts to doubling of the note frequency. If A4 is 440 Hz, then A5 will be 880 Hz and A3 is 220 Hz. Therefore, to map the frequency  $f_i$  to its note number we need to take  $\log_2$  (which will yield the octave number) and multiply it by 12 (to yield the note number). The perceived note at step  $i$  will be

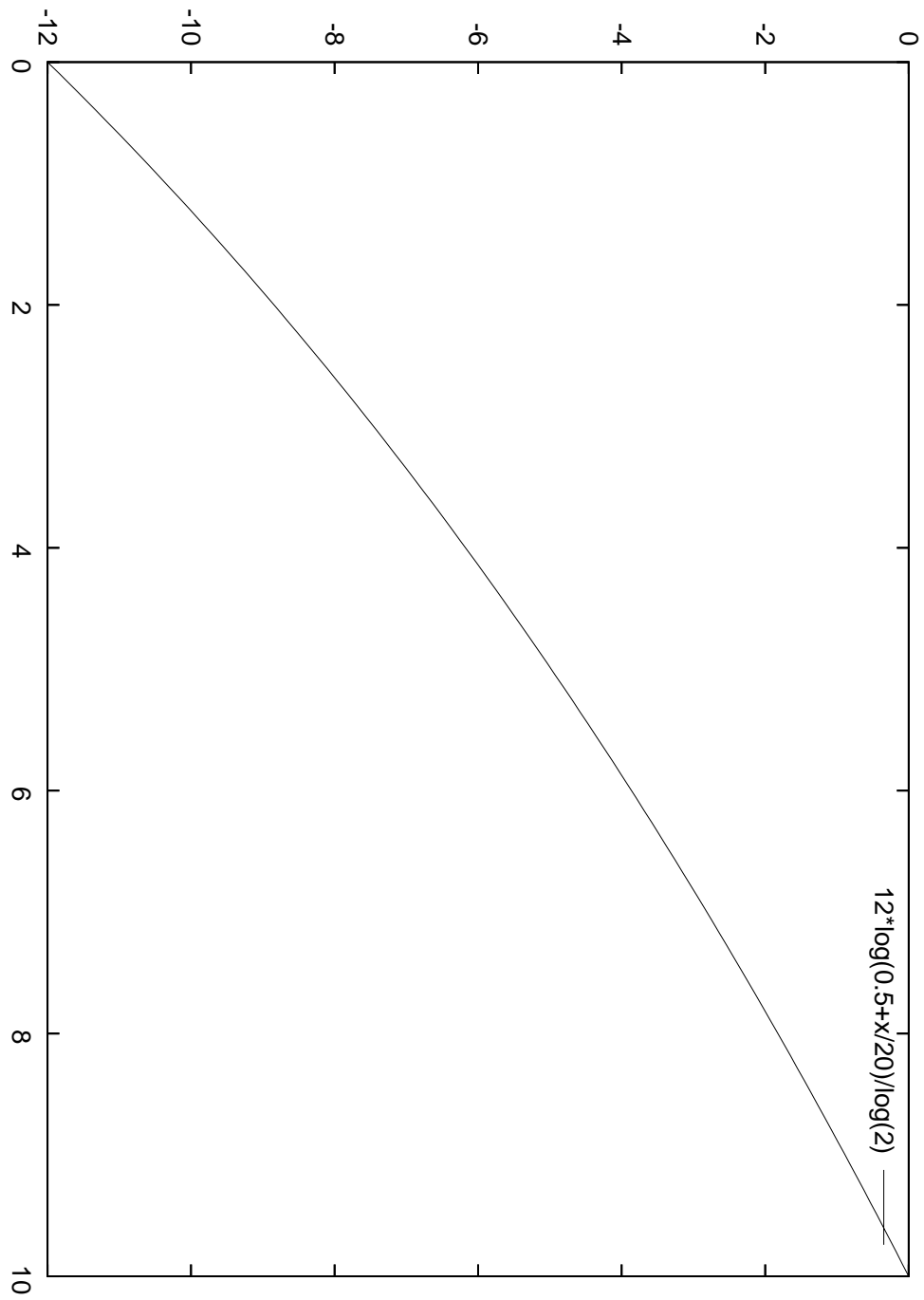


Figure 1: Perceived note during change. Horizontal: time step. Vertical the transposition.

$$n_i = 12 * \log_2 f_i \quad (2)$$

Clearly, we will hear a non-linear scale. To illustrate this, let us assume that we would like to change the tempo of a song from half the playing speed ( $= 0.5r$ ) to its normal playing speed ( $= 1r$ ) in 10 steps. Figure 1 lays out the transposition at a given time step.

The slight bend we can observe does not look as much, but when looking at the local effect by measuring the transposition at every step then the problem becomes more obvious (see figure 2)

### 3 Non linear change

To resolve this problem we must compensate for the logarithm and express our requirements on the note scale instead of the speed-scale. We actually want the transposition at every step to be constant (called  $C$ ). Formalized:

$$n_i - n_{i-1} \equiv C$$

According to 2 this becomes  $12.\log_2(f_i) - 12.\log_2(f_{i-1}) \equiv C$ . According to 1 this further gives

$$\frac{C}{12} \equiv \log_2\left(\frac{f}{r}s_i\right) - \log_2\left(\frac{f}{r}s_{i-1}\right)$$

Merging various constant factors (12,  $f$  and  $r$ ) into  $C$  gives

$$C \equiv \log_2 s_i - \log_2 s_{i-1} = \log_2 \frac{s_i}{s_{i-1}}$$

and thus  $2^C \equiv \frac{s_i}{s_{i-1}}$ , in which  $2^C$  will be a constant itself.

$$C = \frac{s_i}{s_{i-1}}$$

In other words we need a constant multiplier for the playing speeds instead of a constant difference. By induction we get  $s_1 = C s_0$ ,  $s_2 = C s_1$ ,  $s_i = C s_{i-1}$  which leads to

$$s_i = C^i s_0 \quad (3)$$

We further know that  $s_0 = a$  and that  $s_N = b$ , so

$$b = C^N a$$

Isolating  $C$  gives

$$C = \log_N \frac{b}{a} \quad (4)$$

Practically, filling 4 into 3 gives

$$s_i = \left( \log_N \frac{b}{a} \right)^i a$$

To calculate the logarithm using a specific base  $B$  (10 or  $e$  for instance) use the following

$$s_i = \left( \frac{\log_B \frac{b}{a}}{\log_B N} \right)^i a$$

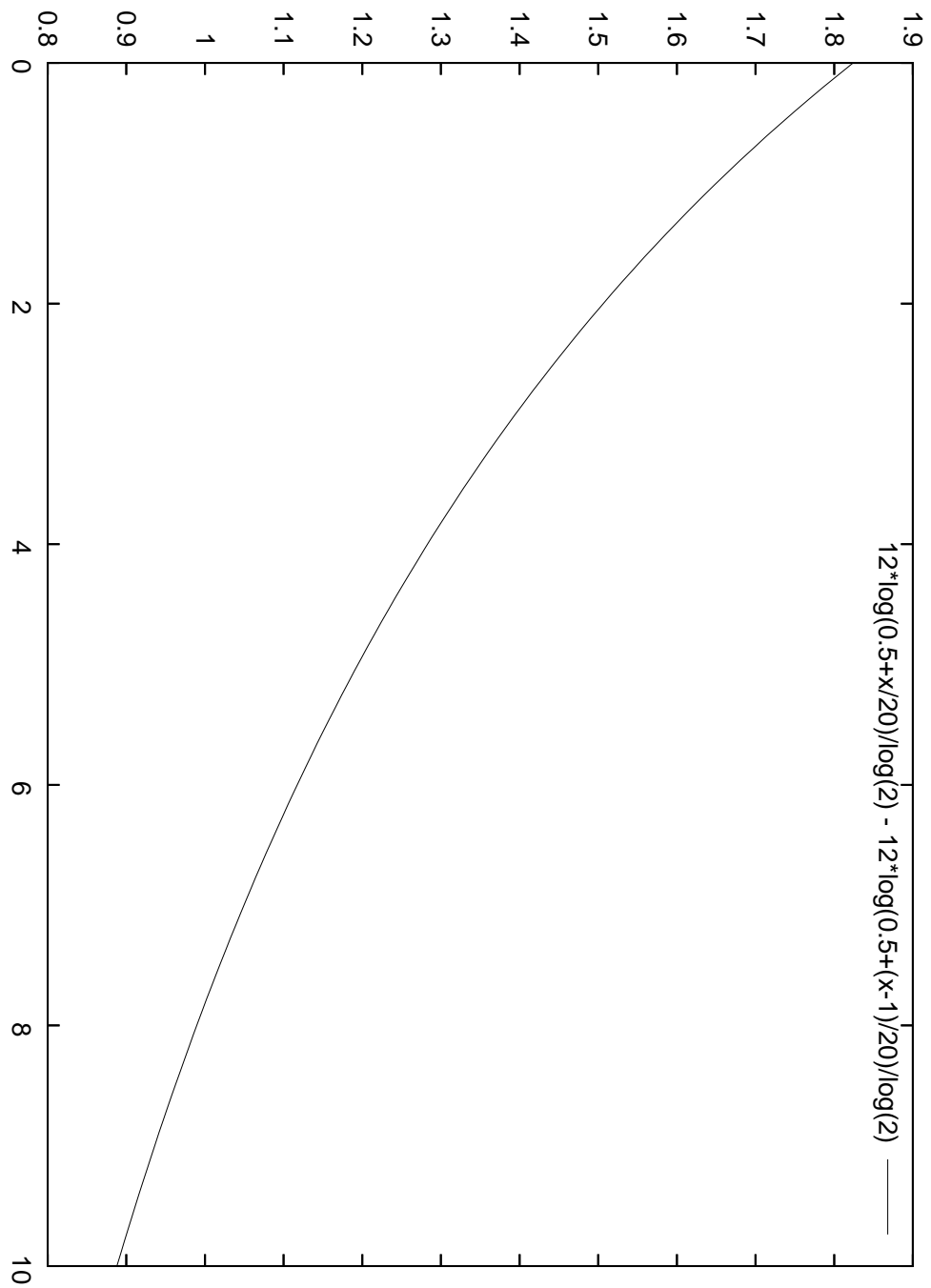


Figure 2: Perceived note change at every step. Horizontal: time step. Vertical the local step transposition.