Muditulib Reference Manual

Muditulib for Pure Data version

Funs Seelen

May 9, 2014

Contents

1 About 1

2 Installation 1

3 Classes 2

4 Abstractions 5

5 Examples 5

1 About

This is the reference manual for Muditulib, a multi-dimensional tuning library. This document is updated for version 0.0-test0 of the corresponding software. More information and articles about the tuning theory behind it can be found at `http://muditulib.eu`.

2 Installation

To install the software first download the package and unpack the `tar`-archive. A command line example is given below. So, open a terminal, navigate to your downloaded file and type:

```
$ tar -xvzf pd-muditulib-0.0test0.tar.gz
```

Then change directory and compile the source for your system.

```
$ cd pd-muditulib-0.0test0
$ make muditulib
```

The sources will now compile into one binary file for the whole library. The last step is to install the library. Because by default files will be copied to your file system you need superuser priviliges to do this (e.g. prepend `sudo` to the next command and type your superuser password).

```
$ make single_install
```

The last thing you need to do is to tell Pd to load your library. The most common way to do this is to add `muditulib` to your libraries to load on startup, within Pd. On linux go to `Edit/Preferences/Startup`. Then quit and restart Pd. Muditulib should now be loaded.

3 Classes

- **midi2ts**

  translates MIDI notes into $Ts()$ values.

  $Ts()$ values are mapped to MIDI notes. How this mapping is done is defined by the *modulation* parameter. The default key mapping at 'mod 0' is equal to the standard baroque keyboard, shown in table 1. If 'mod' is (re-)set, one MIDI note is enharmonically changed for every integer - up or down - in the circle of fifths, calculated from the reference at 'mod 0'. Every modulation up replaces one note in the circle of fifths by adding $(1, -2)$ to its assigned Ts value, starting at MIDI note 3 (*E-flat* to *D-sharp*). In the opposite direction it starts at MIDI note 8 (*G-sharp* to *A-flat*). This means that for example at 'mod 12' all MIDI notes are enharmonically changed once ($0 \rightarrow 11$: from *B-sharp* to *A-double-sharp*). The same applies to 'mod -12', but in the other direction ($0 \rightarrow 11$: from *D-double-flat* to *C-flat*).

MIDI	0	1	2	3	4	5	6	7	8	9	10	11
symbol	c	cis	d	es	e	f	fis	g	gis	a	bes	b
T_n	0	1	1	1	2	2	3	3	4	4	4	5
s_n	0	-1	0	1	0	1	0	1	0	1	2	1

Table 1: Default key mapping at 'mod 0'

arguments: 1. *mod*, default: 0.

inlets: 1. *MIDI notes*

2. *mod*

- **ts2freq**

translates $Ts()$ values into frequency in cycles per second, according to equation 1,

where f is the frequency, T_n and s_n are the numbers of whole tones and semitones, respectively, x is the frequency ratio for the octave, r is the semitone to whole tone ratio, and A is the frequency for the reference a' at $Ts(29, 11)$.

NOTE: $r = \frac{\log s}{\log T}$, if $0 \leq r \leq 1$. If $r > 1$, $r = \frac{\log T}{\log s}$. r is always set to $|r|$.

$$f = x^{((T_n - 29) + (s_n - 11) \cdot r)/(5 + 2 \cdot r)} \cdot A \tag{1}$$

arguments: 1. r, default: 0.5.

2. x, default: 2.

3. A, default: 440.

inlets: 1. T_n, (T_n, s_n)

2. s_n

3. r

4. x

5. A

- `ts2symbol`

translates $Ts()$ values into note name symbols.

The symbols used are based on the *Lilypond* note name symbols.

arguments: no arguments.

inlets: 1. T_n, (T_n, s_n)

2. s_n

- `midi2tts`

translates MIDI notes into $Tts()$ values.

$Ts()$ values are mapped to MIDI notes. The mapping is based on mode and tonic information. The modes from the reference tonic are shown in table 2.

NOTE: in the current implementation the reference tonic is set as a MIDI note. The corresponding $Tts()$ value is extracted from the chosen mode itself. For example in E minor (*mode:* 0, *tonic:* 4) the minor mode starts at (is transposed to) MIDI note 4, $Tts(1, 1, 0)$. The latter is extracted from the minor mode at index 4. One disadvantage of this reference is that the minor mode on MIDI note 1 is then built on top of a *D-flat*, while the major mode is built on top of a *C-sharp*. Usually, one would prefer the exact opposite. However, such issues can be solved by just adding or subtracting $(1, -2)$ to or from all $Tts()$ values outputted by [midi2tts].

arguments: 1. *mode*, default: 1.

2. *transposition*, default: 0.

inlets: 1. *MIDI notes*

Minor (0)			Major (1)		
Dif.	Total	C.	Dif.	Total	C.
s	-	0	s	-	0
s	s	5	T/s	T/s	4
T/s	T	9	s	T	9
s	Ts	14	s	Ts	14
t/s	Tt	17	t/s	Tt	17
s	Tts	22	s	Tts	22
T/s	TTt	26	T/s	TTt	26
s	TTts	31	s	TTts	31
s	TTtss	36	s	TTtss	36
T/s	TTTts	40	t/s	TTtts	39
s	TTTtss	45	s	TTttss	44
t/s	TTTtts	48	T/s	TTTtts	48

Table 2: Tuning patterns for the MIDI keyboard: modes and modulation options from a reference tonic. Shown are the differences to the previous MIDI note and the total amount of distance to the reference in symbols and in commas.

2. don't use for now ...

- **tts2freq**

  Translates $Tts()$ values into frequency in cycles per second.

  The translation is shown in equation 2, where f is the frequency, T_n and s_n are the numbers of whole tones and semitones, respectively, and A is the frequency for the reference a' at $Tts(17, 12, 11)$. The [tts2freq] class also enables slight tuning variations. The syntonic comma can be changed for a Pythagorean or a 53-TET comma (see table 3).

$$f = \left(\frac{9}{8}\right)^{(T_n-17)} \cdot \left(\frac{10}{9}\right)^{(t_n-12)} \cdot \left(\frac{16}{15}\right)^{(s_n-11)} \cdot A \tag{2}$$

  arguments: 1. A, default: 440.

  2. *tuning*, default: 0.

  inlets: 1. T_n, (T_n, t_n, s_n)

  2. t_n

  3. s_n

  4. A

tuning		T	t	s	comma size
0	5-limit	$\frac{9}{8}$	$\frac{10}{9}$	$\frac{16}{15}$	$\frac{81}{80}$ (syntonic)
1	53-TET	$2^{\frac{9}{53}}$	$2^{\frac{8}{53}}$	$2^{\frac{5}{53}}$	$2^{\frac{1}{53}}$
2	3-limit	$\frac{9}{8}$	$\frac{65{,}536}{59{,}049}$	$\frac{2{,}187}{2{,}018}$	$\frac{531{,}441}{524{,}288}$ (Pythagorean)

Table 3: Tuning options

5. *tuning*

4 Abstractions

- **ts2fokker**

  translates a $Ts()$ value to a number of *dieses*[1] and an octave designation. There are some plus and minus adjustments included to fit the needs of the (31-tone) Fokker organ.

- **tts2comma**

  translates a $Tts()$ value to a number of *commas* (53-TET) and an octave designation.

5 Examples

Figure 1 shows an example of the user-defined two-dimensional tuning system. It is set to mean tone temperament with the A at 415 Hz. The *C-sharp* could at this point be the leading tone to D in D minor. The *modulation* index is namely set to 0, which enables the minor keys G, D, and A, along with the major keys *B-flat*, F, C, G, D, and A.

Table 4 shows other tuning possibilities. r (and x if not 2) is given for each tuning. Figure 1 shows how to enter r and x for mean tone temperament.

[1]The *diesis* meant here is one part of an octave that is divided into 31 equal parts. This is very close to the original diesis, the difference between two enharmonically equal tones in mean tone temperament. The latter equals one octave minus three major thirds ($\frac{2}{(\frac{5}{4})^3} = 5^{-3} \cdot 2^7$).

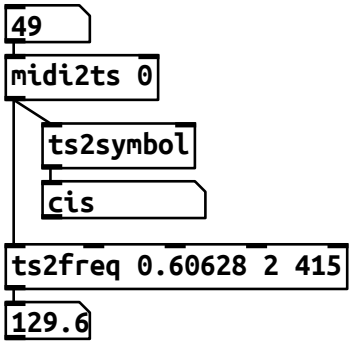

Figure 1: Mean tone temperament, *C-sharp*, A at 415 Hz.

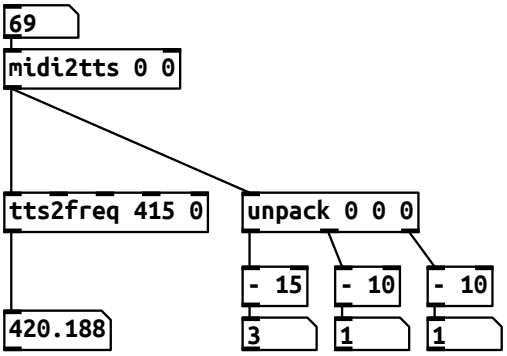

Figure 2: $Tts(18, 11, 11)$: The A in C major is one comma sharp.

Tuning / Temperament	Equation / Calculation	Parameters
Mean tone	$\frac{5}{4} = 2^{2/(5+2r)}$	$r \approx 0.60628$
Pythagorean	$\frac{3}{2} = 2^{(3+r)/(5+2r)}$	$r \approx 0.44247$
	$r = \frac{\log s}{\log T} = \frac{\log \frac{256}{243}}{\log \frac{9}{8}}$	
Tritone temperament	$\frac{7}{5} = 2^{(3/(5+2r))}$	$r \approx 0.59006$
Stretched octave, perfect 5th and 3rds	$\frac{5}{4} = \left(\frac{3}{2}\right)^{2/(3+r)}$	$r \approx 0.63412$
	$x = \left(\frac{5}{4}\right)^{(5+2\cdot r)/2}$	$x \approx 2.01246$
19-TET	$s = 2^{\frac{2}{19}}$, $T = 2^{\frac{3}{19}}$	$r = \frac{2}{3}$
31-TET	$s = 2^{\frac{3}{31}}$, $T = 2^{\frac{5}{31}}$	$r = \frac{3}{5}$
53-TET	$s = 2^{\frac{4}{53}}$, $T = 2^{\frac{9}{53}}$	$r = \frac{4}{9}$

Table 4: Tuning examples of the two-dimensional system.