ON THE MEANING AND VALUE OF MUSIC

Tigran Aivazian <tigran@quantuminfodynamics.com>

21 October 2019

1 Basic Principles

As a result of discussions on the topic of Music and Temperaments with my friend Boris Rziankin I have formulated in my mind a preliminary understanding of the meaning and potential value of music which I wish to describe in this paper. Specifically, I understood how the making of a *temperament* represents (mathematically) nothing other than the attempt of a "finite to encompass the Infinite". I saw this clearly enough to write a program which generates what could be called "rational temperaments" in accordance with the following three general principles:

- **Principle I: Compactification of Infinity.** The Infinite (i.e. the entire space of all possible sounds) should be squeezed into the finite domain of human hearing.
- **Principle II: Octave Equivalence.** The actual method of squeezing should be based on the psycho-acoustical fact that humans (and some other mammals) consider two sounds to be equivalent if their fundamental frequencies are related as a power of 2.
- **Principle III: Harmonic and Harmonically Even Distribution.** In the final stage of the selection of actual sounds from the huge list of candidates generated by the Principles I and II, one should choose those sounds which are distributed evenly (in harmonic sense, not arithmetically) and also stand in harmonic relation to each other (i.e. their frequencies are related as ratios of small integers).

Mathematically, what this means is that we are seeking a mapping from the set of all natural numbers $\mathbb{N}$ to the set of all rational numbers lying between 1 and 2:

$$f : \mathbb{N} \to [1, 2] \cap \mathbb{Q} \tag{1}$$

To build this map $f(n)$ in accordance with the principles formulated above, we first need to define the following intermediate map $k(n)$:

$$k(n) = \min\{k \in \mathbb{N} | 1 < \frac{n}{2^k} < 2\} \tag{2}$$

The map $k(n)$ reflects the **Principle of Octave Equivalence**. And now the explicit form of the map $f(n)$ can be given:

$$f(n) = \frac{n}{2^{k(n)}} \tag{3}$$

Finally, the actual Temperament as a set of frequencies can be constructed using the map $f(n)$:

$$T[N_{min}, N_{max}] = f(\{N_{min}, N_{min}+1, \ldots, N_{max}\}) \tag{4}$$

Here the natural numbers N_{min} and N_{max} serve as parameters that can be varied. It would seem natural to set $N_{min} = 3$ (as the endpoints 1 and 2 are always included) and vary only the value of N_{max}, thus obtaining a family of temperaments T_N which, in a manner of speaking, "encompass greater and greater part of infinity".

The implementation of the above algorithm in Python is enclosed in the last section of this paper. The results of running this program are as follows:

$$T_{32} = \{\frac{17}{16}, \frac{9}{8}, \frac{19}{16}, \frac{5}{4}, \frac{21}{16}, \frac{23}{16}, \frac{3}{2}, \frac{25}{16}, \frac{27}{16}, \frac{29}{16}, \frac{15}{8}\} \tag{5}$$

$$T_{1000} = \{\frac{271}{256}, \frac{575}{512}, \frac{609}{512}, \frac{645}{512}, \frac{683}{512}, \frac{181}{128}, \frac{767}{512}, \frac{813}{512}, \frac{861}{512}, \frac{57}{32}, \frac{967}{512}\} \tag{6}$$

$$T_{5000} = \{\frac{1085}{1024}, \frac{2299}{2048}, \frac{4871}{4096}, \frac{645}{512}, \frac{1367}{1024}, \frac{181}{128}, \frac{3069}{2048}, \frac{3251}{2048}, \frac{861}{512}, \frac{3649}{2048}, \frac{1933}{1024}\} \tag{7}$$

$$T_{10000} = \{\frac{8679}{8192}, \frac{9195}{8192}, \frac{4871}{4096}, \frac{5161}{4096}, \frac{1367}{1024}, \frac{5793}{4096}, \frac{6137}{4096}, \frac{3251}{2048}, \frac{6889}{4096}, \frac{3649}{2048}, \frac{1933}{1024}\} \tag{8}$$

Note that it took 11 seconds to generate T_{10000} on a Linux machine with a 4.2GHz processor. Listening to a sample music played back using ET as well as T_{1000} and T_{5000} shows a subtle difference, especially in the chords. Out of nine people surveyed, two could not detect any difference, five (of which one is a professional musician) showed preference for T_{5000} and two preferred ET. Listening to the same music played in T_{32} results in a uniform feeling of "mistuned piano". The audio files of this experiment are available at `http://quantuminfodynamics.com/audio/1`

2 Another Dimension of Music

The above three principles shed light on both the old Pythagorean temperament and on the modern "12 tone Equal Temperament". However, I believe that we should also consider the following:

Principle IV: An Extra Dimension of Music. This dimension can be realised (at least in the first, rough approximation — later we can refine it) by *floating the temperament*, i.e. by varying (very slightly!) the set of frequencies within a given fixed range of human hearing, precisely according to the way the "Infinite" is represented or modelled mathematically. For example, when I say "infinite number of all possible sounds" I really mean all possible frequencies, but surely one cannot exhaust an infinite set on a computer, so it has to be replaced with some large but finite set like a thousand or a million or a billion frequencies. And the way we thus "approximate Infinity" necessarily affects the resulting sequences of temperaments obtained by applying the three above-mentioned principles.

The above elucidates the meaning, but hardly reveals anything concerning the potential value, mentioned in the beginning. This, I believe, consists in the mysterious power of musical form of expression over our ordinary language as a means of bringing out our inner emotional and intellectual processes, bordering on the spiritual, i.e. our personal "holy of holies" and doing so in a non-vulgar and proper way. And if the entire human being is consumed with but one single desire — to seek God and to do his will, to be more like God — then, perforce, the musical expression of his being should become a reflection of this. And if, in addition, this reflection is skilfully and artistically constructed, then, as the sages say, "it has power a whole world to transform."

3 The Python Program

```
#!/usr/bin/env python3.7

#
# scales.py --- Generate Rational Temperaments,
# as described in my paper "On the Meaning and Value of Music" (2019).
#
# Author: Tigran Aivazian <tigran@quantuminfodynamics.com>
# License: GNU General Public License v3.0
#

import sys
from fractions import Fraction

def equivalentk(n):
    """Keep dividing the number n by 2,3,4,... until it is <= Nmin"""
    i = 2
    x = Fraction(n,i)
    while x>=octmul:
        i += 1
        x /= i
    return x

def equivalent2(n):
    """Keep dividing the number n by 2 until it is <= Nmin"""
    x = Fraction(n,2)
    while x>=2: x /= 2
    return x

def harmonize(i, slist):
    """
       Look in the vicinity of i-th element in sound list slist and
       return the 'most consonant' one.
    """
    slen = len(slist)
    dist = 20 # maximum distance to deviate from the element closest to ET
    if i > dist:
        mini = i - dist
    else:
        mini = 0
    if i < slen - dist:
        maxi = i + dist
    else:
        maxi = slen
    #print("Checking for i=%d" % i, slist[mini:maxi], end='\n')
    s = min(slist[mini:maxi],key=lambda x: x.numerator + x.denominator)
    #print("   returning ", s)
    return s

Noct = 12   # the number of notes per octave (12 for ET)
octaves = 1 # the number of the instrument's octaves
octmul = 2  # the octave multiplier (2 for ET)
Nmin = octmul + 1 # the starting number to fit into the octave
Nmax = int(sys.argv[1]) if len(sys.argv) > 1 else 2000 # the last number to fit

et = [octmul**(n/Noct) for n in range(Noct)] # Equal Temperament (ET) of Noct notes

sounds = [1] # candidate sounds
for n in range(Nmin,Nmax):
   x = equivalentk(n) # call equivalent2(n) for strict 'octave equivalence'
```

```
    if x > 1 and x not in sounds: sounds.append(x)

soundslen = len(sounds)

# build the matrix of distances between et and sounds
dist = [[abs(sounds[i]-et[j]) for i in range(soundslen)] for j in range(Noct)]

temp = [1] # list of frequency ratios for the first octave

# choose those which are close to ET and consonant
for i in range(Noct):
    idx = dist[i].index(min(dist[i])) # index of the sound closest to ET
    sound = harmonize(idx, sounds)
    #sound = sounds[idx]
    if sound not in temp: temp.append(sound) # append if not a duplicate

templen = len(temp)
if templen != Noct: # Incomplete temperament
    print("WARNING: Built only %d notes instead of %d" % (templen, Noct))

instr = temp.copy()
if octaves > 1: print("Extending %d notes to %d octaves" % (templen, octaves))
for i in range(octaves-1):
    #print("%d notes, instr=" % len(instr), instr, end='\n\n')
    instr += [octmul*x for x in instr[-len(temp):]]

# Sort and convert to Yoshimi (and human) readable form
instr.sort()
count = 1
for i in instr:
    if i.denominator == 1:
        print(str(i.numerator), end=', ')
    else:
        print(str(i.numerator) + '/' + str(i.denominator), end=', ')
    count += 1
    if count > Noct:
        count = 1
        print('')
```