Playing with Tuning in bitKlavier

Dan Trueman*, Aatish Bhatia†, Michael Mulshine*, Theo Trevisan*

* Department of Music, Princeton University

† Council on Science and Technology, Princeton University

Abstract

Combining a new software system with the familiar interface of the MIDI keyboard, bitKlavier is a versatile instrument for exploring the nature of tuning and temperament. We describe a number of approaches it facilitates, including composed tunings, moving fundamental systems, and a novel adaptive tuning system. All of these are characterized by the overarching design priority for bitKlavier to be a context for musical play and exploration, as opposed to finding singular, "correct" solutions to particular tuning "problems," as has often been the case historically.

Introduction

Previously, we described how John Cage's prepared piano has served as an important inspiration and metaphor for bitKlavier, a new instrument that combines a flexible software tool with the powerful constraints and ubiquity of the MIDI keyboard (Trueman and Mulshine, 2019). This inspiration has nothing to do with the *sound* of the prepared piano and everything to do with the kind of creative process that it inspires. Specifically, bitKlavier aims to provoke a practice that is both playful and generative, by leveraging the hard-earned embodied feedback loop between player and instrument, and subverting it through an engaged process of preparation.

In our earlier paper, we focused on the development history of the instrument and its time-domain machine-inspired virtual preparations; here we explore the tuning possibilities of the instrument, with an emphasis on *composed tunings* and *dynamic adaptive systems.* We can think of the act of tuning as the original "preparation" for keyboard instruments: it requires design, planning, and significant time to implement, without requiring rebuilding the instrument itself. While the actual retuning can happen instantaneously in bitKlavier, the tuning systems themselves require preparation, and these preparations in turn profoundly shape the creative process. As such, the goal of experimenting with tuning systems in bitKlavier is not primarily about finding exact solutions to tuning problems. Rather, it is more about creating interactive, responsive, and dynamic systems that encourage new possibilities for creative expression via a playful, real-time feedback loop between the musician and the instrument.

Tuning poses long-standing and deeply consequential challenges to musicians and instrument designers, particularly in the case of keyboard-centric instruments (like bitKlavier) with their discrete keys—the division of the octave into twelve static steps creates intractable conflicts with the physics of the overtone series, forcing us to choose from innumerable imperfect (if fascinating) possibilities. Given that bitKlavier is at heart a keyboard-conceived instrument, we will be focusing on this particular 12-key case, and while this is related to the history of temperament, we are freed from some of the constraints that acoustic instruments impose (it is impossible, for instance, to instantaneously retune an acoustic piano). There are many resources for learning more about this history (Keislar, 1987; Duffin, 2008; Barbour, 2004; Isacoff, 2003), as well as learning about just intonation (Doty, 1999) and tuning from around the world (Forster, 2010). A recent article by Stange et al. (2018) provides an overview of the challenges of tuning, while Polansky et al. (Polansky, L., D. Rockmore, M.K. Johnson, D. Repetto, and W. Pan., 2009.) presents a rigorous mathematical approach to tuning and temperament. Finally, Wooley (2018) is a fascinating collection of articles about tuning and the various ways a number of composers and performers are wrestling with it today.

Composed Tunings

What we call *composed tunings* are in some ways primitive, but nonetheless compositionally powerful. In the chorale at letter D in *Nostalgic Synchronic* Etude #5 ("Wallumrød") by Dan Trueman (Figure 1; see also the collection of audio examples available online for this paper), the fundamental for a simple just-intonation system changes periodically throughout, ensuring that each sonority is just tuned while still liberating the progression to move through sonorities that would sound strongly out of tune otherwise. In this example, the slashed notes are played ever so slightly before the other notes, setting the tuning fundamental as indicated by the circled letters (via a *modification* in bitKlavier).

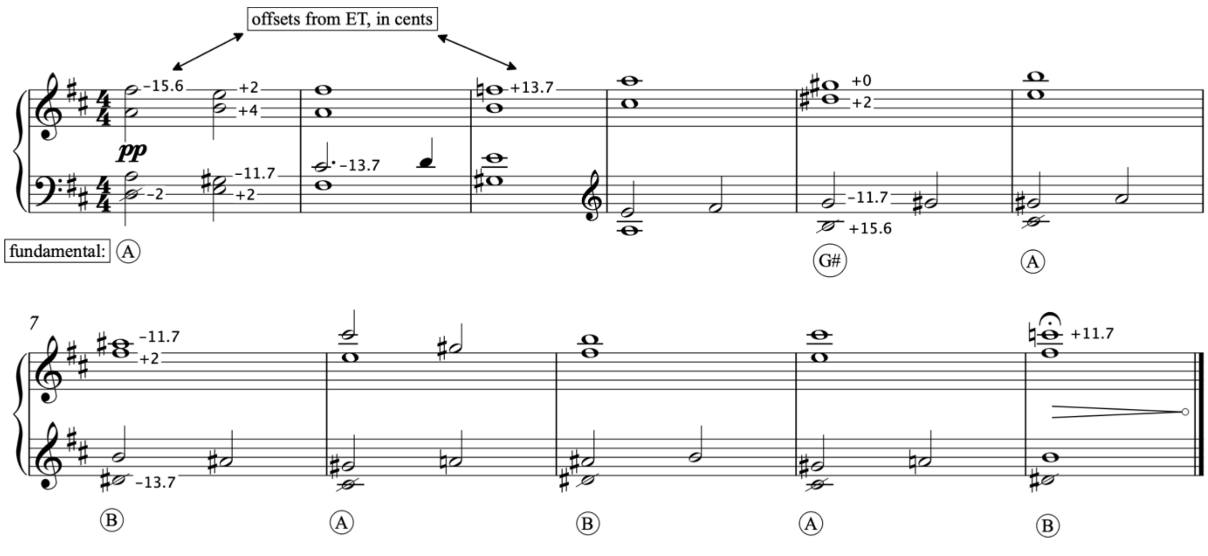


Figure 1. Moving fundamentals (circled). Offsets from ET are indicated only for the first appearance of individual notes, or when they change (for instance, the D# in m7, changing from the D# in m5). Slashed notes are played just before other "simultaneous" notes, ensuring (in this case) that they all use the same new fundamental.

Consider measure 5, for instance. Here, the B has a slash through it, indicating that it should be played slightly before the other notes in that chord. A *keymap* (with this B enabled) is connected to a modification object that in turn changes the fundamental of the tuning to G# when this B is played (see the companion paper to this one, Trueman and Mulshine (2019), for further information about how this sort of signal flow works in bitKlavier), so this entire chord will now be just-tuned relative to G#, resulting in a B that is 15.6 cents sharp to equal-temperament (ET) (a 6:5 minor-third relative to G#), a D# that is 2 cents sharp (a 3:2 perfect fifth), and a G-natural that is 11.7 cents flat (a 15:8 major seventh). From there on out, the fundamental changes in each measure. Note that some pitch classes end up being tuned differently over the course of the chorale (G# in m.1 vs m.5, for instance). It is this ability to manually configure tuning, in composition-specific ways, that motivates that term "composed tuning."

In *Nostalgic Synchronic* Etude #2 ("Undertow"), we experience a horizontal, melodic kind of composed tuning (Figure 2). Here, the D in the left-hand is what changes tuning, so the slashed-F in the right-hand is played just before the D (to be clear: the slash is a performance indicator, not a preparation indicator; it tells the performer to play that note slightly before the others at that moment, but bitKlavier may be prepared so that any of those notes cause the fundamental change), and therefore is tuned according to the prior tuning (C-just). Then, a reverse note (the *nostalgic* preparation type in bitKlavier—backwards piano notes with durations dependent, in this case, on the length of preceding sustained note, triggered on note release) which begins when the F is released (notated with a parenthetical note and dashed swell) will be tuned according to the new system, which is an overtone tuning of A, so the F is treated like the 13th-partial, 40.5c sharp to ET (13/8). We end up with a kind of *micro-voice leading*, where the F moves up by almost equal increments of ~40c before reaching a just tuned F#.

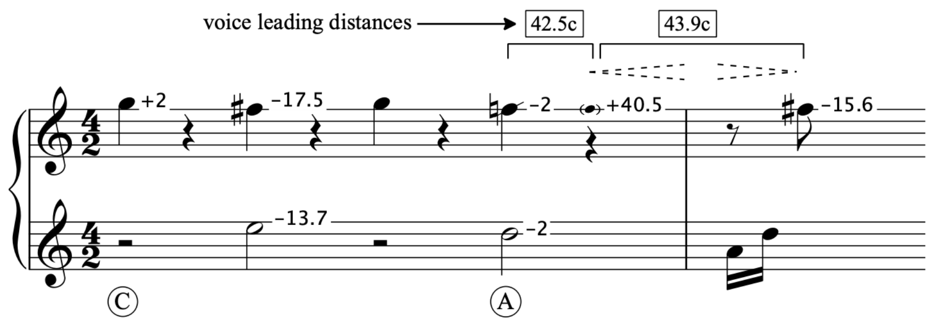


Figure 2. Micro-voice leading, resulting from moving fundamentals, from Trueman's Etude #2, "Undertow." The tuning of first F on beat 4 is set by the C-fundamental, so is 2c flat to ET (4:3 over C), and the A-fundamental (triggered by the D in the left-hand) sets the tuning of last reverse F (13:8 over A, or 40.5c sharp to ET), which then continues to the last F# (5:3 over A, or 15.6c flat to ET). Horizontally, then, the first F shifts up 42.5c (40.5c + 2c) from beat 4 to the offbeat of beat 4, which then shifts up another 43.9c (100c - 15.6c - 40.5c) to the F#.

This might seem a complicated setup for a small gesture, but it is one of only five very similar configurations used in the piece, and, for perspective, is actually quite simple when compared to, say, preparing the piano for Cage's *Sonatas and Interludes.* Crucially, this preparation wasn't created in order to facilitate a gesture like this; rather the gesture was discovered through experimentation with the preparation.

"Systerslått," a movement from *Songs That Are Hard To Sing* for string quartet and percussion quartet by Dan Trueman, is based on a traditional Norwegian fiddle tune that is decidedly not equal-tempered. In the example here (Figure 3), we see the passage notated with Helmholtz-Ellis accidentals in the strings (Sabat and Schweinitz, 2004; Nicholson and Sabat, 2018), which efficiently indicates how each interval is just-tuned (the down arrows indicate just-tuned major 3rds (5:4) and 6ths (5:3), the up arrows just-tuned minor 3rds (6:5) and 6ths (8:5); notes with normal or no accidentals are tuned relative to a background grid of perfect 3:2 fifths) while this particular version of the tuning is not meant to claim traditional accuracy, it does reflect a reasonable hearing of how it might be played. The strings are in scordatura, with open C-G-C-G strings in the viola and cello and G-C-G-E strings in the violins; this open E is just-tuned to the G below, and so is flat by a syntonic comma (a ratio of 80/81, or 13.7 cents), notated here by a single down arrow, as in m.2. In order for the viola and cello A's to be tuned most consonantly to that E, they must also be a syntonic comma flat, hence the down arrow for the A's.

For bitKlavier to match this melody, its fundamentals must change in each measure, and note that in m.2 the fundamental changes to A, a syntonic comma low, in order to match the strings as described. Though the spelling of the bitKlavier part is conventional, simply to make it more readable for the player, the actual sounding tuning will match what the strings play—the right hand of the piano is in unison with the strings. This sort of reasonably complex composed tuning enables the strings to retain the crucial tuning aspects of the traditional melody and play with the keyboard, without rounding off the tuning to the more generic equal temperament.

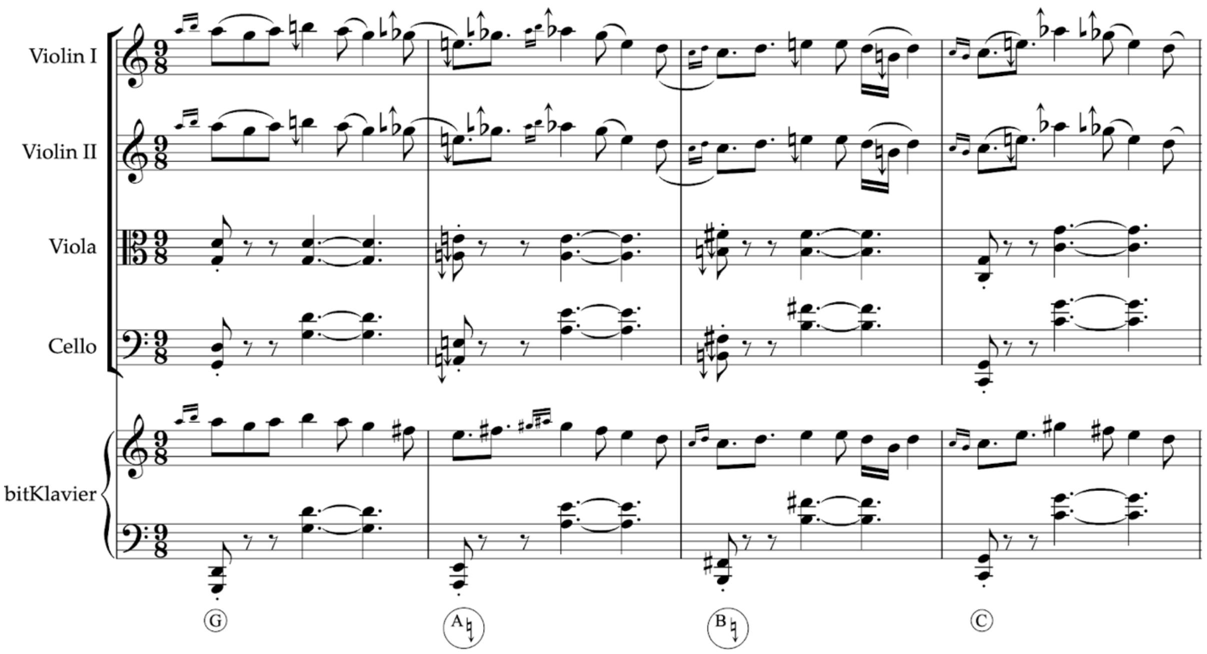


Figure 3. Moving fundamentals to create overtone-tuned melodic lines, approximating the non-ET sound of a traditional Norwegian fiddle tune.

In the final example—*Nostalgic Synchronic* Etude #4 ("Marbles")—the pianist repeats essentially the same pattern in the right hand (mm.1-2, largely repeated in mm. 3-4), but the left-hand D-flat and C change the fundamental of an overtone-based tuning back and forth between D-flat and C (Figure 4).

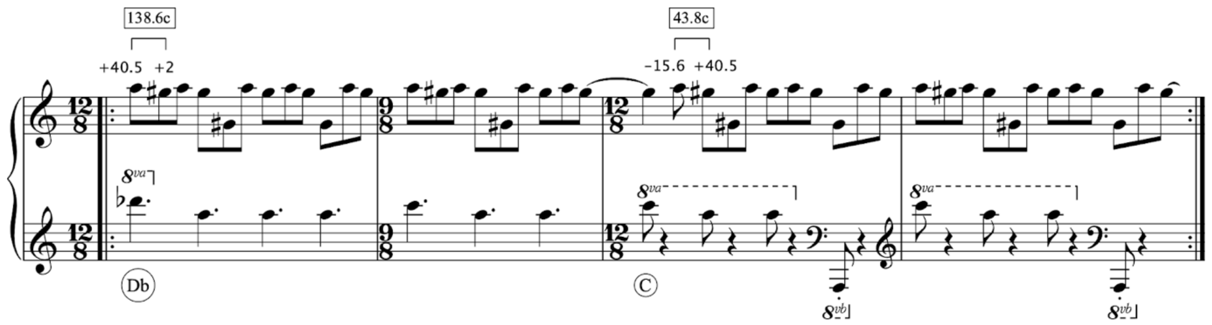


Figure 4. Changing fundamentals under a constant fingering pattern, causing markedly different melodic intervals.

The alternating A-G# interval changes dramatically, from nearly a semitone-and-a-half to less than a quarter-tone, giving a strong sense of expansion and contraction. For the player, this is both galvanizing and disorienting; the piano is changing under the player's hands, and it is unfamiliar to have the same physical pattern generate noticeably different, if still recognizable, sounds.

Adaptive Tunings

The idea that instruments could adaptively tune on the fly as you play goes back at least as far back as Eivind Groven's Pure-tuned Organ in the 1930s (Code, 2002), continuing through Lou Harrison's 1950s "free style" tuning, or what Polansky calls "paratactical" systems (Polansky, 2018; Polansky, 1987). Paratactical systems essentially choose between a set of options for tuning a particular note, depending on the context; Groven's was the first known implementation of such a system, initially created with telephone switchboard technologies, and subsequently recreated digitally by David Code (Code, 2002). In the 1970s, Harold Waage created a hardware system, "the intelligent keyboard," that adaptively tuned while played

(Waage, 1985). Computers also invite the possibility of dynamic continuous systems that aren't limited by a catalog of discrete options; again, Polansky both proposed and implemented such systems (Polansky, 1987; Polansky, 1994). The Hermode system (http://www.hermode.com) is available in Apple Logic Audio and Cubase and works reasonably transparently, optimizing the tuning of sonorities and intervals in constrained ways. It is, however, static in that you have to specify that the entire Logic project session be in Hermode tuning, with particular settings, and then leave it untouched—it is implemented more as a production tool than a performance instrument. *Pianoteq* also has a range of highly sophisticated ways of tuning their instruments, with a catalog of historical temperaments, while also taking into account inharmonicity and octave stretching (www.pianoteq.com). Again, however, their system is static and bound to the idea that once a keyboard is designed and tuned, it is meant to stay that way.

We have implemented two adaptive tuning systems, the first rudimentary and the second inspired by John de Laubenfels' "spring" tuning, as detailed in Sethares (2005, pp.159-162). In both cases our aim is not to create an "optimal" system that calculates the "right solution," but rather to build a system that can be played with, sometimes exhibiting unanticipated or decidedly "badly tuned" behaviors; as Douglas Repetto argues in "Doing It Wrong, "creative acts require deviations from the norm and [...] creative progress is born not of optimization but of variance." (Repetto, 2011)

Moving Fundamental Systems

The two types of "moving fundamental systems" shift the fundamental of a chosen tuning system based on what is played. The fundamental moves either after a certain number of notes have been played, or after a set time has passed. The two flavors of this system differ in where this fundamental then moves; in the first system, the new fundamental is simply the note last played (as tuned), while in the second "anchored" system the new fundamental is tuned to a given background scale.

Consider a system that uses a common just scale to tune consecutive intervals, essentially resetting the "fundamental" of the scale with each note, and the simple melodic pattern in Figure 5. Above each interval are the just-tuned ratios, and below are cents-deviation from ET. Note that the second C is a syntonic comma sharper (81:80) than the first C, because a just-tuned M6th (5:3) is actually smaller than a just-tuned sequence of a P5 (3:2) and M2 (9:8)—3/2 * 9/8 = 27/16 = 5/3 * 81/80. This phenomenon, where sequences of different just-tuned intervals can give rise to naturally rising or falling overall tuning, is sometimes referred to as a "comma pump."

Now, imagine that we would like to reset the fundamental every time we play C4. One way to do that would be to create a *reset* preparation, attach it to the *tuning* preparation and to a *keymap* that has only C4 activated (see the introduction to bitKlavier for further details, Trueman and Mulshine, 2019). The same melody would be tuned a bit differently now, as shown in Figure 6. This time, the C's are tuned the same, and the A's take on a different tuning (also by a syntonic comma).

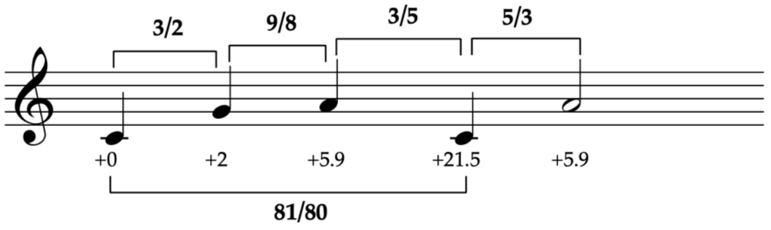


Figure 5. Simple adaptive tuning. Each new note becomes the fundamental for the tuning of the next note, resulting in tuning drift.

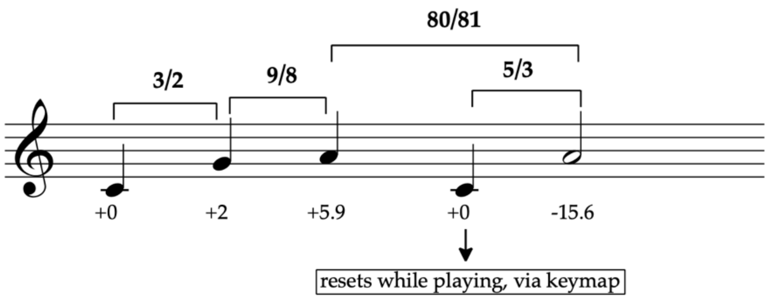


Figure 6. Simple adaptive tuning with prepared resets.

Now, let's consider the second flavor of bitKlavier's adaptive tuning, the "adaptive anchored tuning." Here, the fundamental is reset to a specified anchor scale, rather than the tuning of the last note, after either: 1) a prescribed set of notes have been played, or 2) a prescribed period of time has passed. For example, if the system is prepared to reset after four notes have been played, those first four notes will all be tuned according to the same fundamental, and then notes after that will be tuned according to the fundamental as set by the next note played; we see how this plays out in Figure 7, where the first four notes are all tuned relative to the first C4, and then the 5th note (A4) is simply set to an equal-tempered A, the new fundamental; the next three notes would then be tuned relative to that A, and so on.

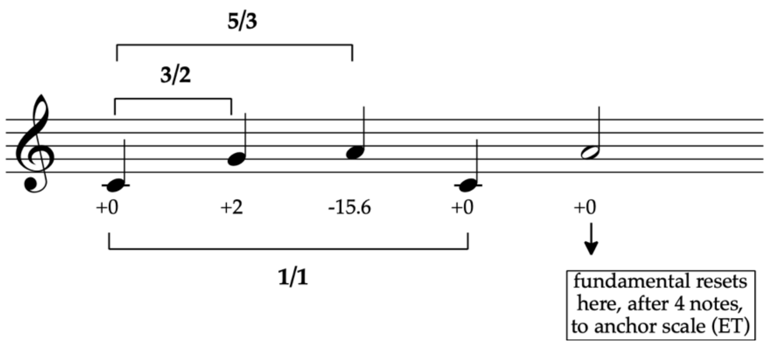

5/3
3/2
+0
+2
-15.6
+0
+0
1/1
fundamental resets here, after 4 notes, to anchor scale (ET)

Figure 7. Adaptive anchored tuning.

These systems are hardly invisible and not nearly as effective as, say, the Hermode system in terms of tuning intervals and chords as "just" as possible without drawing attention to itself. However, they have personality and can be quite provocative to play with. Consider the example in Figure 8, using unanchored adaptive tuning with the fundamental changing with each note (as in Figure 5).

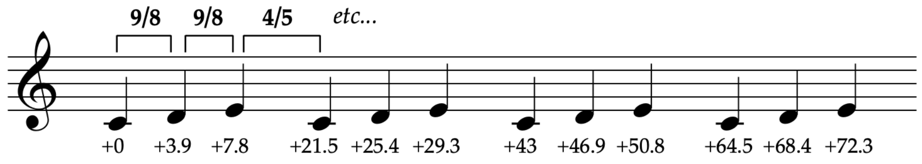


Figure 8. Tuning drift from adaptive tuning. Two ascending 9:8 minor-seconds are larger than one descending 4:5 major-third, causing this pattern to drift upwards by a syntonic comma.

Two 9:8 whole-steps are larger than a 5:4 major-third, so this simple figure drifts upwards by a syntonic comma with every repetition; needless to say, this sort of gesture has compositional promise! Consider for example this passage from "Sinking Song" in *Songs That Are Hard To Sing* (Figure 9).

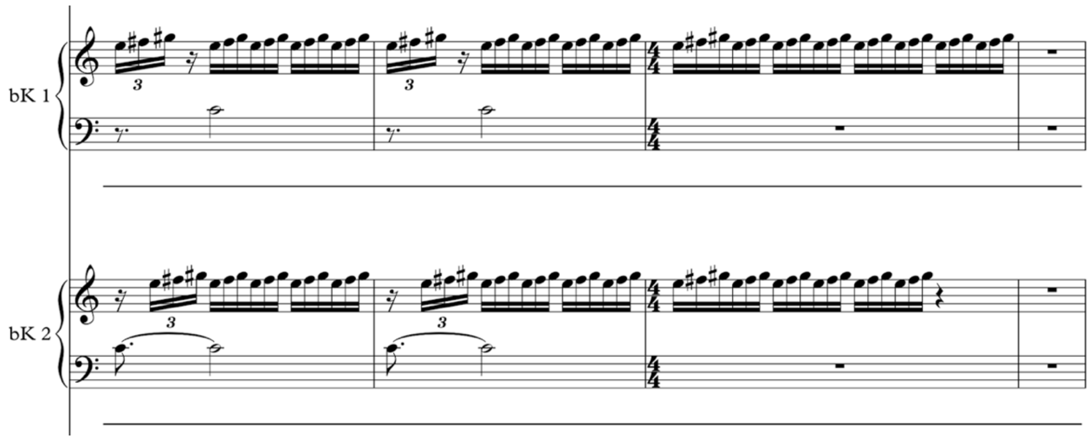


Figure 9. Example of adaptive tuning drift in Trueman's 'Songs That Are Hard To Sing'

Here, the two bitKlaviers gradually "pump" upwards, out of phase with one another, being periodically reset by the middle C's; this is followed by the reverse, where downward figures "pump" down, and in both cases they do this not quite together, creating a disorienting smear.

Spring Tuning

The second adaptive tuning system is entirely different. de Laubenfels proposed a system that (as far as we can tell) has never been implemented in usable form (see Sethares, 2005), perhaps in part because of computational issues at the time. Very recently, and concurrent with our own developments here, Stange et al. (2018) created a powerful adaptive system very similar to de Laubenfels, and it is one we plan to incorporate in bitKlavier in the future. Our own system has some fundamental differences in intent and implementation from both Stange and de Laubenfels which we will explore here.

The basic idea is that a virtual damped spring is connected between every pair of sounding notes, and the spring has a resting position equivalent to a given just-tuned ratio. As the system of notes oscillates, it will eventually settle into an equilibrium configuration that minimizes the overall stretching/compression of the springs—i.e. the energy of the spring system is minimized. In the context of tuning, this translates to minimizing 'out-of-tuneness' (or what de Laubenfels refers to as 'pain'). Specifically, this system will eventually land at a compromise which minimizes the overall deviation of the intervals from their just-tuned counterparts.

For some sonorities, like a major triad, this will result in all three springs at their default length, i.e. all intervals just-tuned. In Figure 10, we see a major triad with all three pitches connected by springs with rest lengths set by just ratios, all of which are compatible.

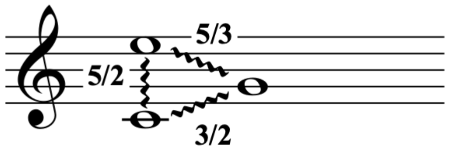


*Figure 10. Springs in a major triad, all of compatible lengths, (3/2 * 5/3 = 5/2)*

But for other sonorities, it's mathematically impossible for all the intervals to be just-tuned. In this case, the springs would either stretch or compress, in real-time, to "temper" the intervals automatically. Eventually, the spring system will settle into the 'least-out-of-tune' compromise (i.e. one that minimizes the spring energy, or the sum of the squares of the deviation of each interval from just tuning).

For example, in Figure 11 the C-E M10th spring would have to stretch, while the m6th springs would have to compress (note that since this is a keyboard-centric system, we don't distinguish between enharmonic intervals, like the m6th and augmented 5th). Some spring strengths might be set stronger than others, so if the M10th spring is stronger than the m6th spring, the tempering will favor the M10th, and so on.

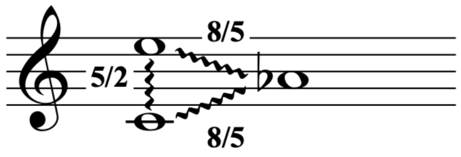


*Figure 11. Incompatible spring lengths in an augmented triad, (8/5 * 8/5 != 5/2)*

Spring Tuning Implementation

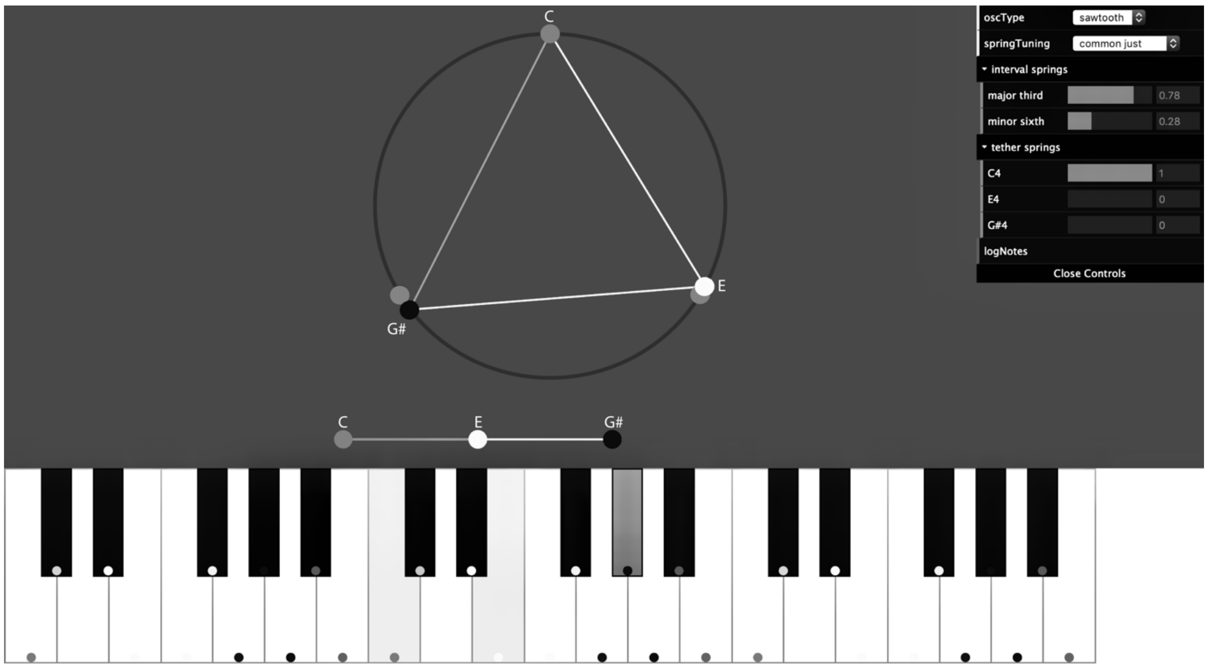
Our implementation of this model utilizes *verlet springs* (Jakobsen, 2003), which are computationally efficient and easy to parameterize with intuitive features like spring strength, stiffness, and drag/friction; this allows us to finely control the manner and rate at which the springs approach the optimal tuning. Our first implementation (https://aatishb.com/springtuning/) was created as a web-based application in the JavaScript creative coding library p5.js (https://p5js.org), using the *Toxiclibs.js* (http://haptic-data.com/toxiclibsjs) physics engine to implement verlet springs. Even in a relatively computationally inefficient language like JavaScript, we can implement in real-time hundreds of springs connecting together dozens of notes, for complex sonorities. In bitKlavier, this system is implemented and expanded in C++. Often used for computer animation, verlet springs move in a naturalistic way that captures the intuitive behavior of more complex spring models, dispersing "energy" over time as they oscillate, and they eventually settle into an equilibrium that minimizes the overall energy (or spring stretching/shrinking) in the system. In the context of spring tuning, this translates to the final tuning of the virtual notes minimizing deviations from the desired target tuning.

As with other adaptive systems, this one can drift, and in fact entire sonorities can gradually drift together, becoming completely unmoored, if they have some residual "momentum" in the springs after adding or removing a note. Our solution is similar to that of Stange et al., in that we introduce "tether" or "anchor" springs that connect each note to a reference. While Stange et al. connect each note to a common reference (which becomes the average 'concert pitch'), we connect every sounding note to a separate reference from a background anchor scale. For example, every note could have a fairly weak tether-spring, or one note could have an immovable spring, and so on, allowing the system to tune in various ways, while keeping parts of it constrained.

In the mathematical Appendix to this paper, we minimize the spring energy equation in the simplified case of equal-strength springs, and demonstrate how equal temperament arises as a limiting case in three different spring tethering scenarios. Our verlet-based simulation matches these theoretical results to high accuracy, verifying that the simulation does indeed converge towards minimizing the spring energy. However, the verlet spring simulation is more general than the results in this Appendix, as the simulation can accommodate springs of unequal strengths (i.e., unequal weights given to different musical intervals), and provides a real-time, dynamic tuning solution rather than an instantaneous optimal solution.

Each verlet spring is controlled by a strength which determines the relative weight of that interval, and how quickly they converge (less stiff springs will spend more time oscillating, stiffer springs will converge quickly). Rather than "optimize" these settings so that the spring system always converges on an ideal tuning in as inaudible a way as possible, we prefer to keep these parameters exposed so the user can prepare the system and play with it in various configurations. For instance, if the virtual springs are all relatively weak, the pitches will gently "vibrate" as they oscillate and converge on the optimal tuning. This pitch oscillation emerges naturally out a dynamic spring-based tuning system, and rather than hide or explicitly prevent this behavior, we think of it as a musically interesting phenomenon worthy of exploration. Indeed in the JavaScript simulation, one can drag the notes with your mouse cursor and release them to create and listen to spring oscillations.

In Figure 12 we see the JavaScript simulation of the system, which illustrates the multiple springs in two ways, circular (above) and linear (below). In the circular representation, pitch classes are placed on the circumference of the circle like a clockface, starting with C at 12 o'clock and moving clockwise by half-step (so, C#/Db would be at 1 o'clock, D at 2 o'clock, and so on); in this example, we have only three pitches active (C, E, G#), which nearly evenly divide the circle into three parts (in the same way that an augmented triad divides the octave evenly). These pitches are connected with lines that represent the three active interval springs (two M3 springs, and one m6 spring). We can see where the ET tunings would be in the shadowed notes (and these ET locations precisely divide the circle into three parts); C (top) has an immovable tether spring, so it is locked into ET, whereas the other two notes diverge significantly from ET. We can also see that the M3rd spring is set to be stronger than the m6th spring, so this solution favors the 3rd over the 6th. Modifying the spring strength causes the system to oscillate and respond audibly in real-time. Again, our intent is to make a system that is engaging to play with and does not suppress "unwanted" behaviors, leaving them available for exploration.



oscType: sawtooth
springTuning: common just
interval springs
major third 0.78
minor sixth 0.28
tether springs
C4 1
E4 0
G#4 0
logNotes
Close Controls

Figure 12. Prototype of spring tuning, in p5.js.

One choice available is the particular scale or table of intervals sizes; for instance, we might choose a flavor of just tuning that includes a 7:4 m7th and a 7:5 a4th/d5th (sometimes referred to as "septimal" intervals, for the inclusion of the prime number 7 in their ratios), as in Table 1. The default unit for spring length is cents, and so, for instance, any two notes a major-2nd apart will have a spring with rest length of 204 cents (and, by extension, 1404 cents for a major-9th, and so on; as of this writing, bitKlavier does not treat compound intervals differently than simple intervals).

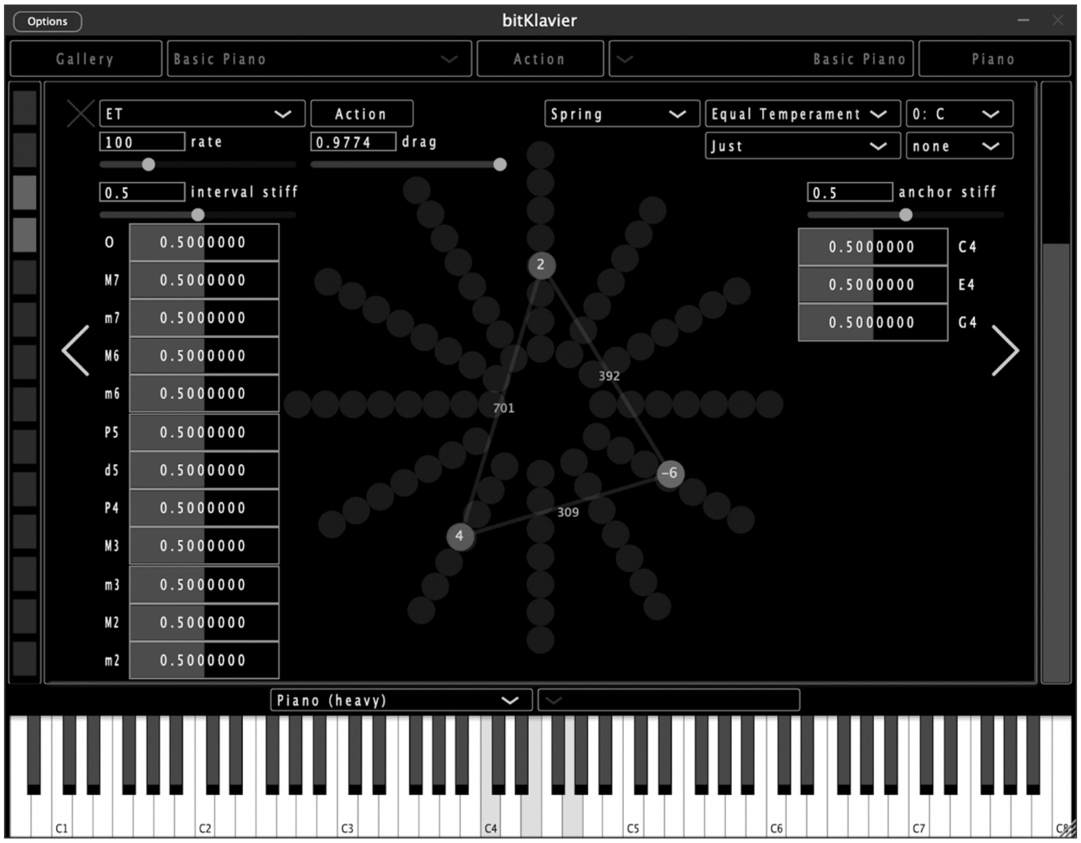
Interval	U	m2	M2	m3	M3	P4	a4/d5	P5	m6	M6	m7	M7
Ratio	**1/1**	**16/15**	**9/8**	**6/5**	**5/4**	**4/3**	**7/5**	**3/2**	**8/5**	**5/3**	**7/4**	**15/8**
Length (cents)	**0**	**111**	**204**	**316**	**386**	**498**	**582**	**702**	**813**	**884**	**968**	**1088**

Table 1. Ratios and rest spring lengths for a septimal just tuning system.

In Figure 13 we see the bitKlavier interface, which locates the pitches on an expanding spiral to allow for different octaves, with the lowest pitches on the inside of the spiral (for instance, all the C's can be found on the 12 o'clock "spoke," with the lowest C near the center of the circle, the highest at the top end of the spoke). The 88 faded circles in the background indicate equal-tempered "anchor" locations, for reference, and numbers within the circles indicate cents offset from ET. Finally, the actual current spring length (not the rest length) is shown on the line connecting pairs of notes. To the left are sliders for setting the relative strengths of the various interval springs, and to the right are sliders for setting the strength of the active tether or anchor springs.

In Figure 13, we have a simple major triad, and a comparison with Table 1 will reveal that none of these springs are at their rest lengths. Why, given that (as described earlier) a major triad consists of compatible just intervals? Shouldn't we have spring lengths of 386, 316, and 702? Because we also have anchor springs, which are all trying to keep these pitches near their equal-tempered tunings, the tuning becomes slightly distorted, though still closer to just-intonation than ET. If we adjust the anchor springs so that only C has any strength and the others have essentially no anchor springs, we get a true just-tuned triad (Figure 14).

In a different vein, we can use the spring system to generate new temperaments, simply by holding down all 12 chromatic notes in an octave and then adjusting the relative strengths of the intervals sliders; strong major-third sliders will bring us closer to ¼-comma meantone temperament, which has pure 5/4 M3rds, whereas strong perfect-fifth sliders will bring us closer to Pythagorean tuning, which has pure 3/2 perfect fifths, and so on. We can even recreate equal temperament by using a symmetric scale for our tuning intervals (where every interval and its complement multiply to 2, e.g. 9/8 * 16/9 for the M2nd and m7th), holding down all 12 notes, and having all the spring strengths equal (see Appendix for a mathematical derivation of this). While these techniques are far from systematic (as in, for instance, Polansky et al. (2009)), they afford, again, a playful approach to the subject, and one that can yield literally an infinite number of temperaments.



100 rate

0.9774 drag

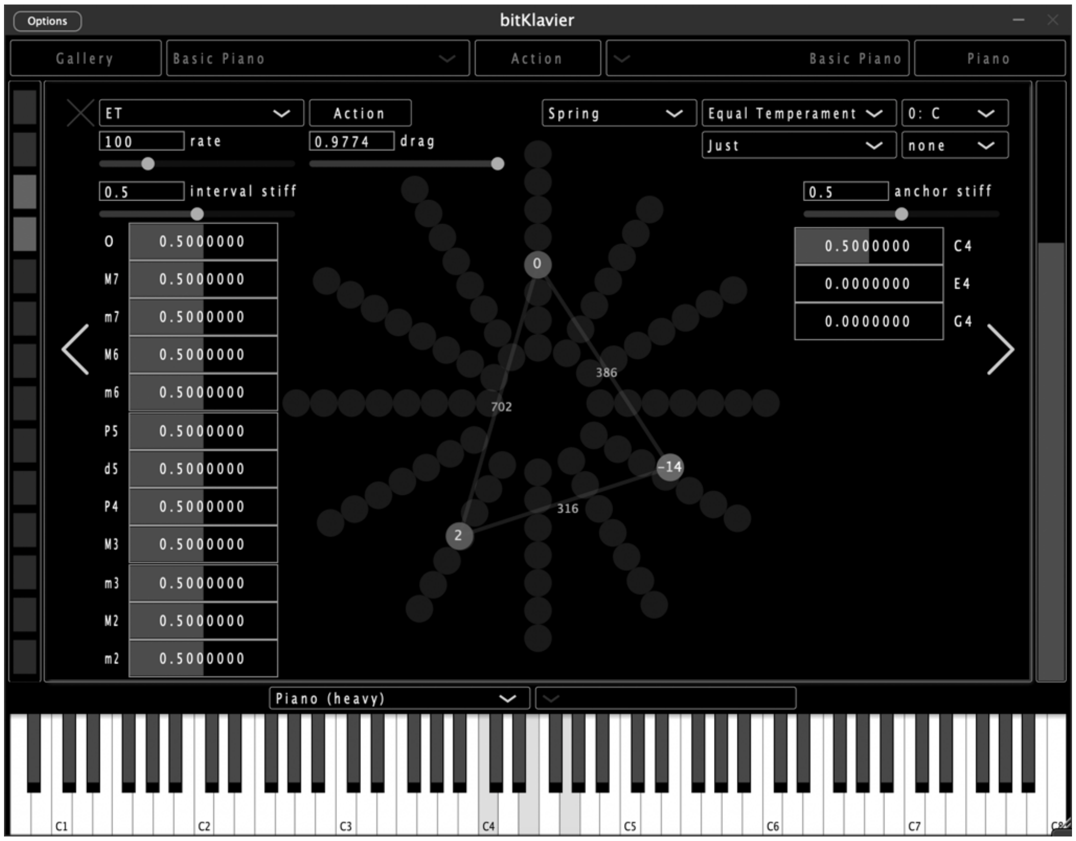
0.5 interval stiff

0.5 anchor stiff

O	0.5000000
M7	0.5000000
m7	0.5000000
M6	0.5000000
m6	0.5000000
P5	0.5000000
d5	0.5000000
P4	0.5000000
M3	0.5000000
m3	0.5000000
M2	0.5000000
m2	0.5000000

0.5000000	C4
0.5000000	E4
0.5000000	G4

Figure 13. spring tuning interface, with anchor sliders distorting a major triad's tuning.



ET
Action
Spring
Equal Temperament
0: C
Just
none
100 rate
0.9774 drag
0.5 interval stiff
0.5 anchor stiff

O	0.5000000
M7	0.5000000
m7	0.5000000
M6	0.5000000
m6	0.5000000
P5	0.5000000
d5	0.5000000
P4	0.5000000
M3	0.5000000
m3	0.5000000
M2	0.5000000
m2	0.5000000

0.5000000	C4
0.0000000	E4
0.0000000	G4

Piano (heavy)

Figure 14. With only one anchor spring active (note the E4 and G4 anchor sliders near the top right are now set to 0), the major triad is now tuned as expected.

The overall *stiffness* of the interval and anchor springs affects the dynamics of the system, causing the springs to vibrate more or less quickly, and the *drag* models a kind of overall friction in the system (imagine the springs submerged in a viscous fluid), damping the spring oscillations. Finally, the *rate* (in Hz) determines how frequently the spring model is updated. All four of these parameters can be adjusted to create a wide range of different responses, from slowly oscillating systems to almost inaudible and instantaneous tunings.

Non-unique Interval Sizes and Moving Fundamentals

One of the more complex components of this system is the *interval scale fundamental*. When set to “none” (as in Figure 14, above the “anchor stiff” slider), the behavior is as described so far, where the resting length for a spring is set locally, only with regards to the interval between its two pitches. So, for example, every M2nd spring will be set to 204c when using the just scale from Table 1. In this case, the D in a C-D-E trichord will be squeezed by its springs on either side, and it will in turn push the C-E M3rd a bit wide. In Figure 15, we see the M2nds are a bit compressed, and the M3rd stretched (these steady-state values are taken directly from the bitKlavier simulation).

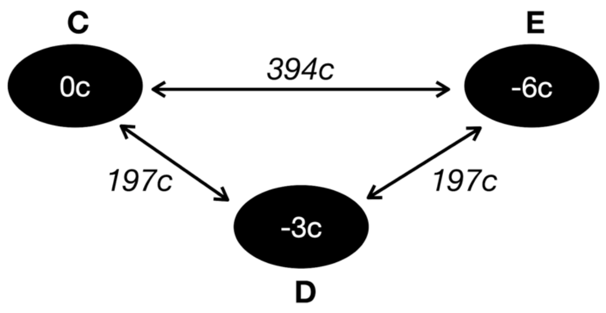


Figure 15. A M3 with a whole-step in between; the M2 springs push the M3 wide, while the M3 pushes the whole-step springs narrow, leaving the springs stressed.

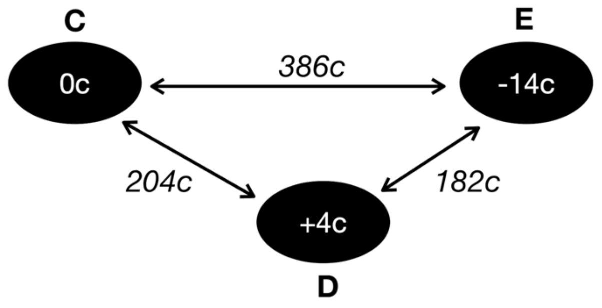


Figure 16. With C defined as the fundamental, the C-E and D-E whole-steps take on different rest lengths, so this system can converge to a tuning with no stress in the springs.

If we set the interval scale fundamental to C (instead of "none", in that same menu), however, the D-E M2nd will use the 10:9 ratio (386 - 204 = 182c, from Table

1) of that just scale (which is the ratio between the major 2nd and 3rd in that scale; 9/8 * 10/9 = 5/4), so all the springs will naturally sit at their rest lengths (Figure 16).

We are now using the scale in a more global way, where the spring lengths depend on the particular pitches, and not just the interval between the pitches; in doing so we highlight the fact that non-ET scales have non-unique interval sizes, and while this is a well-known issue (see Doty, 1993, for instance), it remains a thorny one if we want to fully explore the possibilities of just intonation at the keyboard.

As with many problems, there are multiple ways to approach this; for instance, in the example from Figure 15, we could simply weaken the M2nd spring or strengthen the M3rd spring. That would result in a more just-tuned 3rd, but the D would still be placed halfway between (like a meantone tuning), a different result than that of Figure 16, and a choice that may have other musical consequences. We should add that this domain of playing with the variable spring strengths is rich with possibility, and still largely unexplored.

bitKlavier's movable fundamental allows us to retain some of the richness of the non-unique interval sizes in a coherent fashion. One paradoxical consequence of this is that after notes have been added or subtracted from a sonority and the springs have stopped oscillating, they will all be at their rest lengths; this is essentially tautological, since using the spring length table this way guarantees that all spring lengths will be compatible, but not unique—we might have, for instance, two different spring lengths for the perfect 5th, one for the familiar 3:2 ratio, and another for a "wolf" interval (40:27 between D and A in Table 1, when the fundamental is C), or the two different M2nds as already described.

Why, then, use the springs at all? For one, the springs can change tunings over time, in response to changes in fundamental and the addition of new notes or the subtraction of existing notes. With bitKlavier, in addition to simply setting a fixed fundamental, we can have it change on the fly based on the currently sounding notes via four different modes: *lowest, highest, last,* and *automatic* (all from the same menu where *none* is selected in Figure 14), each indicating which note should be used to set the fundamental. When in *last* mode, for example, the most recent note played is always the fundamental, meaning that the resting spring lengths for existing intervals may have to change every time a new note is added; the springs gracefully adjust for these changes, retuning on the fly dynamically. We also encounter sonorities that are systematically tuned differently depending on how they are played; consider how a dominant 7th chord would be tuned (in *last* mode) when arpeggiated upwards vs. downwards.

Another approach that the springs afford is the mixing and matching of spring lengths based on a fundamental with those determined locally, as in the core spring tuning (when the fundamental is set to *none*). Why might we want to do this? Consider again the case of the 40/27 5th between D-A, when the fundamental is C. Given the central role of the 5th in the overtone series, it would be reasonable to simply ask that all perfect fifths be 3/2, and similarly its complement, the perfect fourths, be 4/3. We might also want to avoid the Pythagorean major-3rds (81/64) that sometimes crop up, replacing them with 5/4 springs. To enable these sorts of combinations, bitKlavier (when in one of the “fundamental” modes) has a toggle next to each interval spring slider that determines whether the fundamental sets its spring length (F) or whether the length is set locally (L; so if all are set to L, then the system will behave identically to if it were in “none” mode). While this can be confusing, and in all likelihood most users will not make use of these options or even understand what they are, in the spirit of play and exploration we want to make this sort of configurability possible; it’s important to remember that we can’t anticipate all the musical situations that might arise in this world of dynamic tuning, and so while we are at risk of having too many features, we don’t want to overly prescribe what we think might be worthwhile. Nonetheless, we have several examples built-in to bitKlavier, all of which can be used as presets without concern for how they work under the hood.

The *highest, lowest,* and *last* modes beg the question: can't bitKlavier just figure out what an ideal fundamental would be to tune whatever sonority is played "as just as possible?" Of course, that is a qualitative question, but for the examples considered so far (a stack of two whole-steps, the dominant 7th chord) it's a reasonable one. In *automatic* fundamental mode, bitKlavier looks at the sounding notes and tries to determine whether they suggest a particular overtone series, inspired in part by the psychoacoustic phenomenon of the "missing fundamental" (see Heller, 2013, and Giordano, 2016, for instance). In Figure 17, we see the overtone series based on C, and then its first few intervals, in complementary octave pairings, so the P5/P4, the M3/m6, and the m3/M6. The *automatic* algorithm works down from the top of whatever chord is sounding looking for these intervals; if it finds any of these intervals, it returns a fundamental of C. It prioritizes 5ths and 4ths over the M3/m6, and similarly the M3/m6 over the m3/M6, in the event that different intervals suggest different fundamentals, and by working from the top down and returning the last fundamental found, it prioritizes lower register voices; imagine a stack of perfect fifths, and the bottom fifth would determine what the fundamental is. If none of these intervals are found, then it leaves the fundamental unchanged.

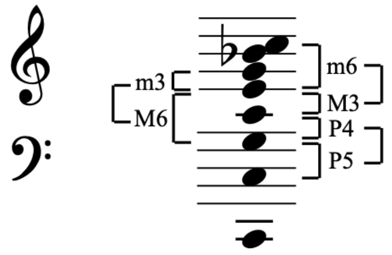
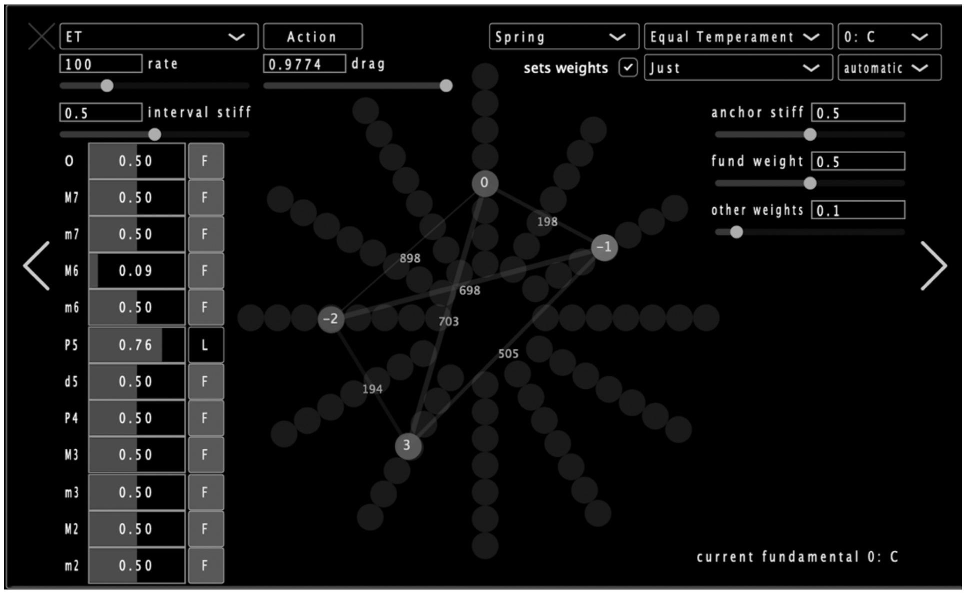


Figure 17. Intervals in the overtone series used to automatically determine a reasonable fundamental for the currently active notes. The algorithm works from the top down, favoring 5ths and 4ths over 3rds and 6ths when they point to different fundamentals.

There are many other possible approaches to the question of automatically calculating an ideal fundamental or optimizing spring lengths more generally, including ones that attempt to minimize stored spring energy, others biased towards undertones, to yet others that work with Sethares' "spectral dissonance" measures (Sethares, 2005), all of which we are looking at. But this simple algorithm is inexpensive to calculate, even when many notes are sounding, and it yields predictable and intuitive results; the C-D-E trichord yields a C fundamental, as does a C dominant 7th chord, and so on—it is a starting point.


ET | Action | Spring | Equal Temperament | 0: C
100 rate | 0.9774 drag | sets weights ✓ | Just | automatic
0.5 interval stiff
anchor stiff 0.5
fund weight 0.5
other weights 0.1

O	0.50	F
M7	0.50	F
m7	0.50	F
M6	0.09	F
m6	0.50	F
P5	0.76	L
d5	0.50	F
P4	0.50	F
M3	0.50	F
m3	0.50	F
M2	0.50	F
m2	0.50	F

current fundamental 0: C

Figure 18. A stack of perfect fifths, C-G-D-A, with automatic fundamental finding and anchor weight setting, and the P5 set to "local" mode and weighted more heavily than the M6. Note that the bitKlavier "gallery" for this figure (available with online materials for this article) has two different "pianos" that the reader can try, one that converges very quickly and smoothly on the stable tuning, and one that oscillates audibly as it converges, due to much lower drag and spring stiffness values. The audio example for this figure has both versions as well.

One final component of the "fundamental" mode in spring tuning is that the fundamental can also be used to determine the anchor spring weights; this is particularly useful in the mobile fundamental modes like *last* and *automatic*, enabling the anchor springs to track the changing fundamentals. In Figure 18, we see this and both the automatic fundamental finding and the local/fundamental setting for the interval springs, on a stack of perfect fifths, C-G-D-A.

Here, since C-G is the lowest fifth, the system determines that C is the fundamental, which will then have an anchor stiffness of 0.5, and all the other sounding pitches a stiffness of 0.1. Then, with the P5 set to "local" (L), the D-A P5 spring is set to the typical 702c, rather than the 680c that the "fundamental" (F) mode would define. This creates a tension between the C-A M6 spring and the D-A P5 spring, which is weighted to favor the P5, so the fifths in this sonority will be close to 3:2. Remember that this is a dynamic system, though, so the M6 won't always be compromised in this way; for instance, if we simply release the D, the D-A M6 will shrink down closer to its normal 5:3 884c length.

Comparisons with Other Systems

Stange's adaptive system addresses non-unique interval sizes through referencing a small table of common variations on particular intervals; when confronted with, say, a major-2nd, the algorithm will run its calculations for both flavors and choose the one that causes the least overall stretching of all the active springs from their rest lengths. Since their system is designed to calculate the optimal tuning for any given situation, the approach makes sense, and it bears some similarity to our mix-match spring system. However, it remains unclear how the Stange system determines where to locate non-unique intervals in a particular sonority; for instance, in the C-D-E example of Figures 19–21, it's not clear how their system determines which whole-step to use the alternative ratio with, since either option will result in identical stretching. One advantage of the moving fundamental approach here is that it orders the non-unique intervals according to the structural logic of the tuning systems themselves, which reflect particular priorities.

Both the de Laubenfels and Stange models have "horizontal" springs that constrain the change in pitch of individual notes over time, to suppress what might be undesirable rapid changes in tuning to a particular pitch. Stange builds a model of "intonational memory", a damped decaying function with approximately three second decay factor (based on psychoacoustic studies), and then uses this for "horizontal" adaptive tuning, corresponding roughly to the horizontal de Laubenfels springs, though it is also calculated immediately rather than embedded in an oscillating system. To this point, we have not found a need to introduce such springs, perhaps because the combination of the interval and tether springs constrains the system enough already, or because we are taking a broader view as to what is "desirable" in our system, where "artifacts" become musical features that we are interested in exploiting. We may introduce these horizontal springs in a future version, but for now we prefer not to add that additional layer of complexity.

More broadly, our system differs from the Stange system in intent. Stange's system calculates an optimal tuning directly, and has some time-domain aspects to include a sense of intonational memory, but these are intended to be as sonically invisible as possible. Of course, the springs in our system can be set to be all relatively strong, with ample drag, so that the convergence is nearly instantaneous and inaudible as well, but our aim is to create something that can inspire play and engage us in musically unexpected ways, perhaps enabling us to discover new music in the process, so we want to enable a range of behaviors and expose as many parameters as might be musically useful. The process and results are different enough that we plan to include the Stange system in bitKlavier in the future, as a complementary adaptive system to our spring tuning.

Zooming out, adaptive systems like these treat tuning more like a stretchable fabric than a rigid lattice, and their interactive dynamic quality emphasizes this flexibility (indeed, verlet springs can be used to build simulations of fabrics; see, for instance, https://aatishb.com/drape/). Bob Gilmore (1995) highlights Harry Partch's persistent use of "fabric" as a metaphor for his extended just intonation systems, revealing an urge to make tuning "an active, expanding fabric of relationships capable of informing, shaping, and ordering the pitch domain: a *laying down* [his italics] of the total pitch space." Gilmore goes further, arguing that Partch's approach, along with those of Ben Johnston and James Tenney, is at least as much about *discovery* and ideas as it is about description and explanation, in line with our own thinking here. For a fascinating exploration of this in Tenney, see Michael Winter's detailed article about Tenney's last composition, *Arbor Vitae* (Winter, 2008).

Discussion

"Every tuning system makes an ideology audible, however (un)justly it might strike our ears." (Moseley, p.108)

The long standing challenges of musical tuning are not only of taste, style, or culture; they reflect—in vivid sonic colors—how our world is fundamentally at odds with itself, down to the non-negotiable elements of number theory: $2^n \neq 3^m \; for\ any\ integers\ m\ and\ n$. This tells us that the two most basic of musical intervals—the first two intervals we encounter in the overtone series, the octave (2:1) and just-tuned fifth (3:2) —are irreconcilable; stack fifth after fifth and it will never equal an octave (Dunne and McConnell, 1999; Heller, 2013; more broadly, this is true for any pair of prime numbers, not just 2 and 3 (Polansky et al., 2009)). There is no perfect solution, there are only compromises, some more interesting than others. ET has been proven a particularly compelling compromise, fudging all of the intervals other than the octave (and even that is fudged on the piano, where inharmonicity rears its head; see Giordano (2010)), thereby enabling the expansive modulating tradition of 19th-century tonal music, as well as the extended triadic harmonies of jazz, among others. But, as has been well established, this was not the tuning of J.S. Bach and his *Well Tempered Clavier* (see Swich, 2011, among others), and is most certainly not the tuning of Indian classical music, Indonesian gamelan music, and countless other musics from around the world (see the many tunings explicated in Forster, 2010, for instance, or Polansky, 1987 and Polansky et al., 2009).

The integers 2 and 3 are also at the heart of how the keyboard itself is laid out, with its alternating pattern of black keys; this may not be a coincidence—stack a few fifths, the pentatonic scale of the black keys emerges, and stack a couple more, the diatonic scale of the white keys follows. Attempts over the centuries to add additional keys so that players can find notes "between the cracks" have yet to catch on, perhaps because their complexity seems daunting for our hands and minds. Given the staying power of the traditional keyboard, we might conclude that its layout is as simple as it should be, without being too simple (imagine there were no black keys at all, just a continuous row of identical keys sounding half-step after half-step) and that this enabled a flourishing performance practice of embodied technique, continuing to this day. Speculation aside, the keyboard as it is, with its alternating sequence of 2s and 3s, remains a powerful interface for musical exploration, one that continues to challenge and inspire musicians today, and one that offers much to the contemporary digital instrument builder.

We are told that J.S. Bach could retune his harpsichord in 15 minutes (Barbour, 2004, p. 191, attributed to C.P.E. Bach; Reinhard, 2009), sometimes even more quickly if he only needed to adjust a string or two to change keys. Tuning and retuning were part of the daily practice of composers and performers at the time, the immutable physical keys under the hands giving rise to tonal keys with variable individual character to the ear, with some thirds smaller than others, some fifths beating faster than others. While digital instruments like bitKlavier are inferior to their analog ancestors in many ways, their software enables retuning orders of magnitude faster than Bach could retune, issuing an invitation to re-engage in the rich world that lies between 2 and 3, in both new and old ways. Invoking William James, Duffin (2008) implores us to embrace the "booming, buzzing confusion" of this in-between music. In order to do so, we need new instruments, new algorithms, and new ways to play and discover.

> "The keyboard is material and ideal, manipulable and manipulative, obfuscatory and revelatory. It both constitutes and represents a field of play on which systematized actions and reactions unfold according to certain rules, but the stakes, the means of regulation, and the interpretation of outcomes are all contingent and contestable." (Moseley, p. 109)

The approaches to tuning in bitKlavier are intentionally messy and open. Composed tunings are clumsy, tying the tuning of the instrument itself directly to the composed notes, allowing the instrument to intervene in the compositional thinking, forcing some notes to be played before others, and yet other notes to be avoided lest they cause an unexpected retune. But this clumsiness offers both constraints and possibilities to the composer, affording new kinds of voice-leading, timbres, and phrase structures (think of the expanding and contracting "half-step" in "Marbles"). Moving fundamentals in our adaptive tuning are similarly clumsy, but they also transform problems (comma pumps, for instance) into features, phenomena with compositional possibility. Spring tuning then accelerates the 15-minute Bach retune into a perpetually oscillating web of converging and diverging intervals; the once static tempered sonority passes before our ears, returning moments later, coming to rest if we let it, and not always where we expect it.

Naturally, all of these systems are works in progress, and we hope they will be indefinitely. This is consistent with the history of instrument building, where the instruments themselves were always considered in-progress, leading to new music, leading to new instruments, and so on.

Acknowledgements

We are grateful to Jeff Snyder for his feedback on the paper and the development of the various tuning systems. Thanks to Larry Polansky for his thorough review and suggestions. Support for bitKlavier has come from the American Council of Learned Societies, the Center for the Digital Humanities, the Council on Science and Technology, and the Music Department at Princeton University.

References

Barbour, J. M. 2004. *Tuning and Temperament: A Historical Survey*. North Chelmsford, Massachusetts: Courier.

Code, D. L. 2002. “Groven.Max: An Adaptive Tuning System for MIDI Pianos.” *Computer Music Journal* 26(2):50–61.

Doty, D. 1993. *The Just Intonation Primer*. San Francisco: The Just Intonation Network.

Duffin, R. W. 2008. *How Equal Temperament Ruined Harmony (and Why You Should Care)*. New York: Norton.

Dunne, E. and M. McConnell. 1999. “Pianos and Continued Fractions.” *Mathematics Magazine*, 72(2), 104-115. doi:10.2307/2690594.

Forster, C. 2010. *Musical Mathematics: On the Art and Science of Acoustic Instruments*. San Francisco: Chronicle Books.

Gilmore, Bob. 1995. "Changing the Metaphor: Ratio Models of Musical Pitch in the Work of Harry Partch, Ben Johnston, and James Tenney." *Perspectives of New Music* 33:458–503

Giordano, N. 2016. *Physics of the Piano.* Oxford: Oxford University Press.

Heller, E. 2013. *Why You Hear What You Hear.* Princeton: Princeton University Press.

Isaacoff, S. 2003. *Temperament: How Music Became the Battleground for the Great Minds of Western Civilization*. New York: Penguin Random House.

Jakobsen, T. 2003. "Advanced Character Physics." Published online at www.cs.cmu.edu/afs/cs/academic/class/15462-s13/www/lec_slides/Jakobsen.pdf. Accessed April 2019.

Keislar, D. 1987. "History and Principles of Microtonal Keyboards." *Computer Music Journal* 11(1):18–28. Revised 1988 as "History and Principles of Microtonal Keyboard Design," Report STAN-M-45, Stanford University. Available online at ccrma.stanford.edu/STANM/stanms/stanm45. Accessed 31 March 2019.

Moseley, R. 2016. *Keys to Play: Music as a Ludic Medium from Apollo to Nintendo.* Oakland: University of California Press. doi.org/10.1525/luminos.

Nicholson, T. and M. Sabat. 2018. "Fundamental Principles of Just Intonation and Microtonal Composition." Universität der Künste Berlin. www.marcsabat.com/pdfs/JI.pdf. Accessed 3 April 2019.

Polansky, L. 1987. "Paratactical Tuning: An Agenda for the Future Use of Computers in Experimental Intonation," *Computer Music Journal*, 11(1): 61–68

Polansky, L. 1994. "Live Interactive Computer Music in HMSL, 1984-1992." *Computer Music Journal*, 18(2), 59-77. doi:10.2307/3680444.

Polansky, L., D. Rockmore, M.K. Johnson, D. Repetto, and W. Pan. 2009. "A Mathematical Model for Optimal Tuning Systems." *Perspectives of New Music* Vol. 47, No. 1.

Polansky, L. 2018. “A Few Words About Tuning.” *Sound American* 20. soundamerican.org/sa20larrypolanskyafewwords.html#polanskynote8. Accessed 3 April 2019.

Reinhard, J. 2009. “Bach and Tuning.” The Stereo Society, stereosociety.com/wp-content/uploads/2017/02/BACHandTUNING-screen.pdf. Accessed 8 April 2019.

Repetto, D. 2011. “Doing it Wrong” *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2010 Symposium*. Washington, DC: The National Academies Press. doi: 10.17226/13043.

Sabat, M. and W. Schweinitz. 2004. “The Extended Helmholtz-Ellis JI Pitch Notation.” Plainsound. www.marcsabat.com/pdfs/notation.pdf. Accessed 3 April 2019.

Sethares, W. 2005. *Tuning, Timbre, Spectrum, Scale.* London: Springer-Verlag.

Stange, K., C. Wick, and H. Hinrichsen. 2018. “Playing Music in Just Intonation: A Dynamically Adaptive Tuning Scheme.” *Computer Music Journal*, 42:3, pp. 47–62. doi:10.1162/COMJ a 00478.

Swich, L. 2011. “Further thoughts on Bach's 1722 temperament.” *Early Music*, 39:3. doi.org/10.1093/em/car041.

Waage, H. 1985. ‘‘The Intelligent Keyboard.’’ *1/1 (The Quarterly Journal of the Just Intonation Network)* 1(4):1, 12–13.

Winter, M. 2008. “On James Tenney’s Arbor Vitae for String Quartet.” *Contemporary Music Review*, Vol. 27, No. 1, pp. 131 – 150. DOI: 10.1080/07494460701671566

Wooley, N. 2018. “The Just Intonation Issue.” *Sound American* 20. soundamerican.org/sa20index.html. Accessed 3 April 2019.

Appendix—Spring Tuning

The virtual spring system in bitKlavier will eventually converge to a solution that minimizes the energy of the spring system. Here we derive these optimal tuning values for three different tethering scenarios. We have compared these derivations with the actual behavior of the spring-tuning system in bitKlavier and found them to match very closely. Note that this section considers the steady-state convergent values we expect the system to reach, and is not modeling the iterative time-based behavior of the system as it oscillates, which is audible and a key feature of this system. Also, here we assume equal spring strengths, to simplify the derivations; the actual system in bitKlavier does not have this simplification.

The results below omit a few tedious algebraic manipulations. A more detailed derivation can be obtained at https://aatishb.com/springtuning/appendix.pdf.

Minimizing the Energy of a Spring System

Our goal here is to find the set of optimal notes x_i that minimize the energy in the spring system (eq. 1).

$$E = \sum_{\substack{i,j \\ j>i}} \frac{1}{2}\omega(x_j - x_i - I_{ij})^2 \tag{1}$$

Here x_i is the pitch of the ith note (in cents), I_{ij} is the desired interval between the ith and jth note (also in cents), and ω is the spring constant. The i and j indices range from 0 to N-1 (where N is the number of sounding notes, or keys pressed), and to avoid double counting, we only include terms where j exceeds i. To simplify the math, in this appendix we assume that all the interval springs are of equal strength, i.e. $\omega_{ij} = \omega$.

We can minimize the energy by differentiating (1) with respect to each coordinate x_i and setting the result equal to zero, giving a set of N equations. In physics terms, these derivatives correspond to the force acting on each note. In equilibrium, all these forces must be zero. Solving for this condition, we find that:

$$x_k = \frac{\sum_i x_i - \phi_k}{N} \tag{2}$$

where ϕ_k are a set of constants defined in terms of the intervals:

$$\phi_k \equiv -\sum_{i<k} I_{ik} + \sum_{i>k} I_{ki} \tag{3}$$

Scenario 1: Adding a Reference Pitch By Fixing a Note

The set of equations in (2) do not have a unique solution, because we have not yet set a reference pitch. The simplest way to do this is to fix the pitch of a single note. Say we fix x_0. Rearranging (2) for $k = 0$,

$$\sum_i x_i = Nx_0 + \phi_0$$

Substituting this value of $\sum_i x_i$ in (2) yields the solution

$$x_k = x_0 + \frac{\phi_0 - \phi_k}{N} \tag{4}$$

If $I_{ij} = I_{j-i}$, i.e. the intervals depend only on the difference of indices, it can be shown that

$$\phi_0 - \phi_k = \sum_{i=1}^{k} (I_i + I_{N-i}) \tag{5}$$

which allows us to re-express (4) as

$$x_k = x_0 + \frac{1}{N} \sum_{i=1}^{k} (I_i + I_{N-i}) \tag{6}$$

If the intervals are also symmetric, i.e. if complementary intervals add to an octave ($I_i + I_{N-i} = 1200$), the solution simplifies to $x_k = x_0 + \frac{1200k}{N}$. These are equally spaced notes, so we recover equal temperament.

Scenario 2: Attaching Tethers to N Reference Notes

In this section, instead of locking a single note as a constraint, we add a set of 'tether springs' that connect each note to a corresponding reference pitch x_i^{ET} (for example, these could be the equal temperament pitch of each note). This adds a new term to the energy equation

$$E = \sum_{\substack{i,j \\ j>i}} \frac{1}{2}\omega(x_j - x_i - I_{ij})^2 + \sum_i \frac{1}{2}\epsilon(x_i - x_i^{ET})^2 \tag{7}$$

Once again, for mathematical simplicity we assume here that the interval springs all have the same strength ω, and the tether springs have the same strength ϵ. Typically, we use the system in a regime where $\omega \gg \epsilon$. Minimizing the energy as before, we arrive at the solution for equilibrium.

$$x_k = x_k^* - \frac{\phi_k}{N + \frac{\epsilon}{\omega}} \tag{8}$$

where

$$x_k^* = \frac{\sum_i x_i^{ET} + \frac{\epsilon}{\omega} x_k^{ET}}{N + \frac{\epsilon}{\omega}} \tag{9}$$

are a set of constants defined in terms of the reference notes x_i^{ET}. Notice that the ϵ/ω term in the denominator causes the intervals to shrink compared to when tethers are absent.

In the limit of very weak tethers ($\epsilon \ll \omega$ or $lim_{\frac{\epsilon}{\omega} \to 0}$), (8) reduces to

$$x_k = \overline{x^{ET}} - \frac{\phi_k}{N}$$

where $\overline{x^{ET}}$ is the average of the reference notes. If the intervals are symmetric, then by rearranging (5) we find that $\phi_k = \phi_0 - 1200k$. Substituting this in the expression above, we recover equal temperament.

Scenario 3: Attaching Tethers to a Single Reference Note

Instead of tethering each note to a separate reference pitch, one could also tether them all to a single reference pitch x^*. This is similar to the approach followed by Stange et al. and is a special case of the previous configuration (7), with $x_i^{ET} = x^*$. Substituting this into (9) gives

$$x_k^* = \frac{\sum_i x^* + \frac{\epsilon}{\omega} x^*}{N + \frac{\epsilon}{\omega}} = x^*$$

yielding the solution

$$x_k = x^* - \frac{\phi_k}{N + \frac{\epsilon}{\omega}} \tag{10}$$

In the limit of very weak tethers ($\epsilon \ll \omega$ or $lim_{\frac{\epsilon}{\omega} \to 0}$), this reduces to

$$x_k = x^* - \frac{\phi_k}{N}$$

In this limit, we recover the same solution as the previous section. Once again, if the intervals are also symmetric, then $\phi_k = \phi_0 - 1200k$, and once again we recover equal temperament.