Dan Trueman,[*] Aatish Bhatia,[†]
Michael Mulshine,[**]
and Theo Trevisan[‡]
[*]Effron Music Building
Lewis Center for the Arts
Princeton University
Princeton, New Jersey 08544, USA
[†]516 W 162nd Street, New York, New York 10032, USA
[**]Center for Computer Research in Music and Acoustics
Department of Music
Stanford University
Stanford, California 94305, USA
[‡]2455 Frist Center
Princeton, New Jersey 08544, USA
{dtrueman, trt2}@princeton.edu,
aatish@gmail.com,
mulshine@ccrma.stanford.edu

Tuning Playfully: Composed and Adaptive Tunings in bitKlavier

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 system for adaptive tuning. 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.

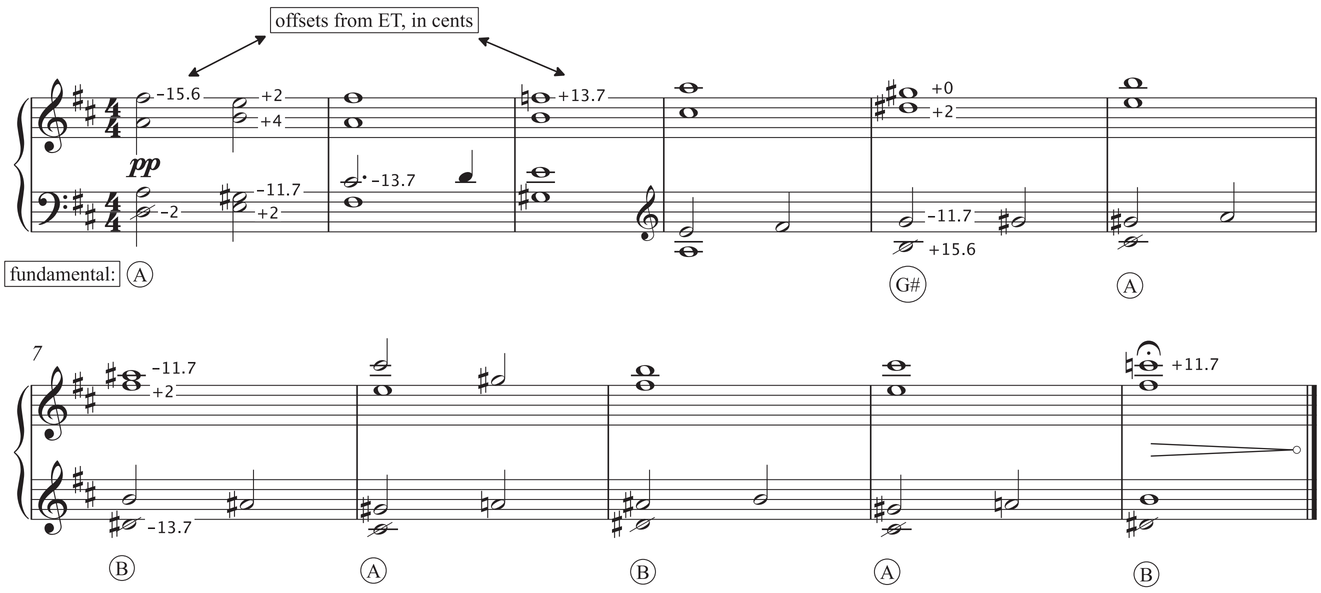
In the companion article in this issue of *Computer Music Journal* (Trueman and Mulshine 2020) we described how John Cage's prepared piano has served as an important inspiration and metaphor for bitKlavier, a new instrument that combines a flexible software tool with the powerful constraints and ubiquity of the MIDI keyboard. 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 taking advantage of the hard-earned embodied feedback loop between player and instrument, and subverting it through an engaged process of preparation.

In the companion article 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. Although 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, dynamic systems that encourage new possibilities for creative expression via a playful, real-time feedback loop between the musician and the instrument.

Tuning poses longstanding 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

Computer Music Journal, 43:2/3, pp. 67–88, Summer/Fall 2019
doi:10.1162/COMJ_a_00519

Figure 1. Moving fundamentals (circled). Offsets from equal temperament (ET) are indicated only for the first appearance of individual notes, or when they change (for instance, the D♯ in measure 7, changing from the D♯ in measure 5). Slashed notes are played just before other notes in the same chord, ensuring (in this case) that they all use the same new fundamental (cf. Audio Example 1).



choose from innumerable imperfect (if fascinating) possibilities. Given that bitKlavier is at heart a keyboard-conceived instrument, we will be focusing on this particular twelve-key case, and although 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; Isacoff 2003; Barbour 2004; Duffin 2008), as well as learning about just intonation (Doty 1993) and tuning from around the world (Forster 2010). A recent article by Stange, Wick, and Hinrichsen (2018) provides an overview of the challenges of tuning, and a rigorous mathematical approach to tuning and temperament has been developed by Polansky et al. (2009). Finally, Wooley (2018) assembled 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 Etude No. 5 ("Wallumrød") from Dan Trueman's *Nostalgic Synchronic* the fundamental for a simple just-intonation system changes periodically throughout, ensuring that each sonority is in just intonation while still liberating the progression to move through sonorities that would otherwise sound strongly out of tune (see Figure 1; audio realizations of the musical examples in the figures can be viewed at https://www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00519). In this example, the slashed notes are played just slightly before the other notes, setting the tuning fundamental as indicated by the circled letters (via a "modification" in bitKlavier).

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 article, in this issue of *CMJ*, "Preparing the Digital Piano: Introducing bitKlavier," for further information about how this sort of signal flow works in bitKlavier), so this entire chord will now be in just intonation relative to G♯, resulting in a B that is 15.6 cents sharp to equal

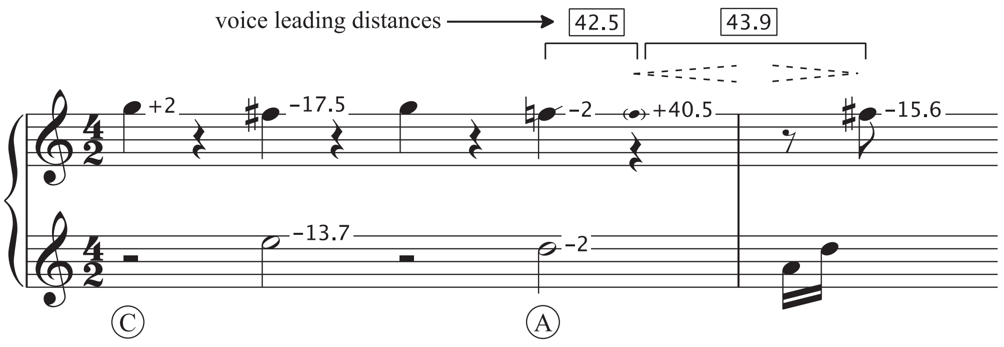


Figure 2. "Micro–voice leading," resulting from moving fundamentals, as used in Etude No. 2 "Undertow" from Trueman's Nostalgic Synchronic. *The tuning of the first F on beat 4 is set by the C fundamental, so it is 2 cents flat to ET (4:3 over C), and the A fundamental (triggered by the D in the left hand) sets the tuning of last reverse F (13:8 over A, or 40.5 cents sharp to ET), which then continues to the last F♯ (5:3 over A, or 15.6 cents flat to ET). Horizontally, then, the first F shifts up 42.5 cents (40.5 cents + 2 cents) from beat 4 to the offbeat of beat 4, which then shifts up another 43.9 cents (100 − 15.6 − 40.5 cents) to the F♯. (Audio Example 2.)*

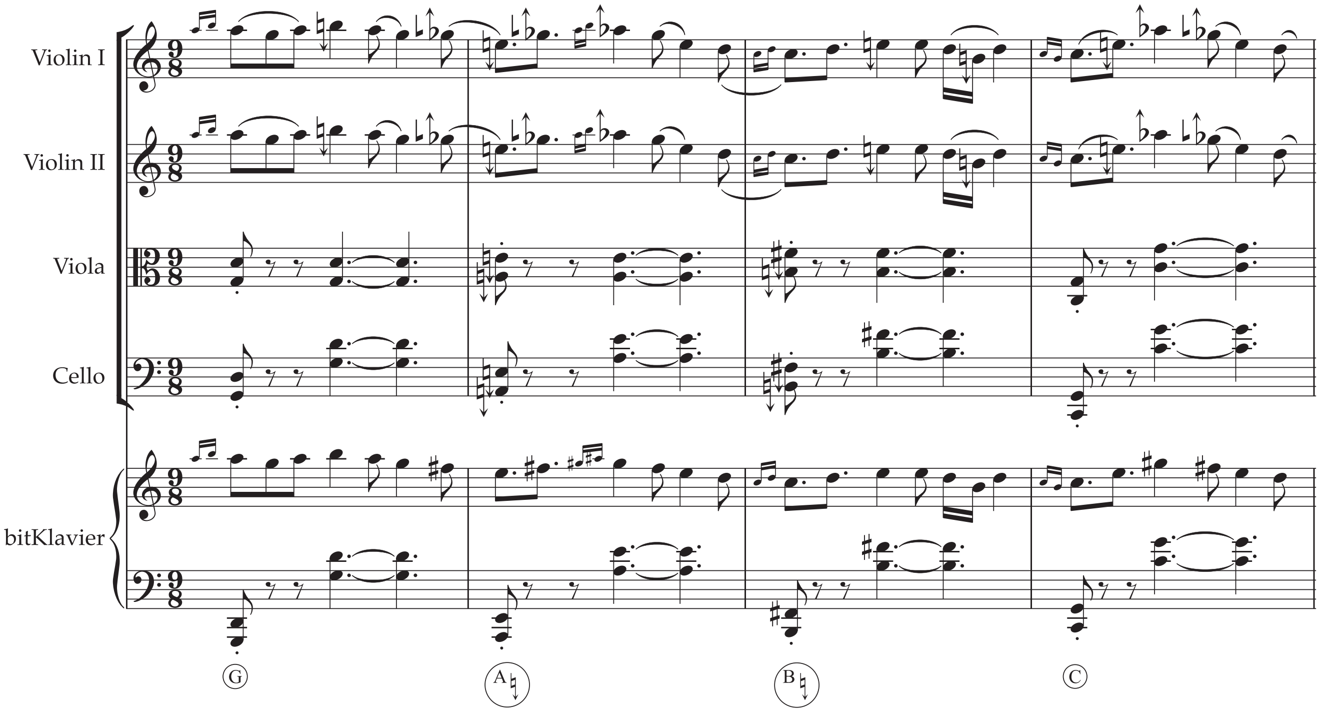
temperament (ET), that is, a 6:5 minor third relative to G♯; a D♯ that is 2 cents sharp (a 3:2 perfect fifth); and a G♮ that is 11.7 cents flat (a 15:8 major seventh). From there on out, the fundamental changes in each measure. Note that some pitch classes end up being tuned differently over the course of the chorale (G♯ in measure 1 versus measure 5, for instance). It is this ability to manually configure tuning, in composition-specific ways, that motivates that term "composed tuning." The settings files for bitKlavier for this figure and many others are included with the online resources for the article at https://www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00519, should the reader be interested in experimenting with these directly.

In Etude No. 2, "Undertow," from *Nostalgic Synchronic* we experience a horizontal, melodic kind of composed tuning (see 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, and therefore is tuned according to the prior tuning, "C-just." (To be clear about the slashed notes: 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.) Following this, a "reverse note" that begins when the F is released 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. (The reverse note is realized with the Nostalgic preparation type in bitKlavier. This plays piano notes backwards with the duration dependent, in this case, on the length of preceding sustained note and is triggered on note release. The sound is indicated in the score by a parenthetical note and dashed crescendo and decrescendo marks.) We end up with a kind of "micro–voice leading," in which the F moves up by almost equal increments of about 40 cents before reaching an F♯ in just intonation.

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 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 (see 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 tuned: The down arrows indicate just intonation for major thirds (5:4) and sixths (5:3); the up arrows indicate just intonation for minor thirds (6:5) and sixths (8:5); and notes with normal or no accidentals are tuned relative to a background grid of perfect fifths (3:2). Although 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.

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



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. The violin open E is in just intonation to the G below, and so is flat by a syntonic comma (the ratio of 81:80, or 13.7 cents), indicated here by a single down arrow, as in measure 2. For the As in viola and cello to be tuned most consonantly to that E, they must also be a syntonic comma flat, hence the down arrow for the As.

For bitKlavier to match this melody, its fundamentals must change in each measure, and note that in measure 2 the fundamental changes to A, a syntonic comma flat, 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.

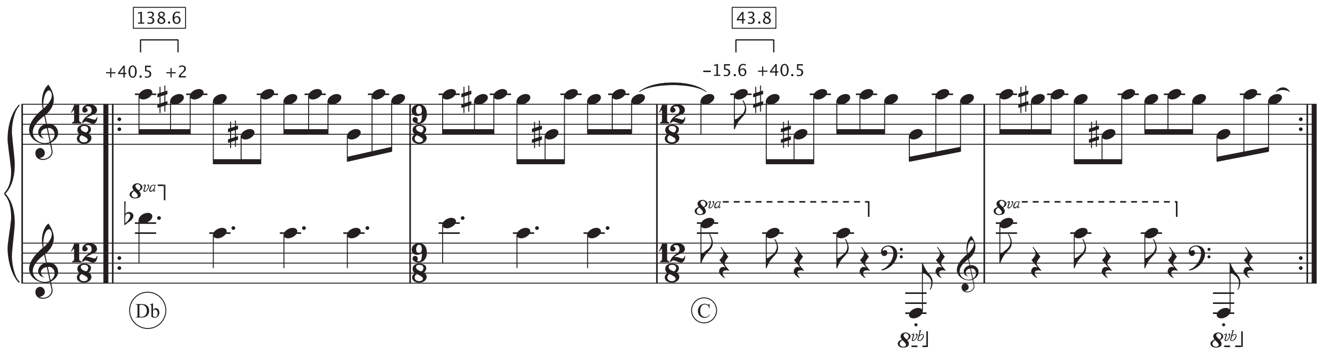
In the final example—Etude No. 4, "Marbles," from *Nostalgic Synchronic*—the pianist repeats essentially the same pattern in the right hand (measures 1–2, largely repeated in measures 3–4), but the left-hand D♭ and C change the fundamental of an overtone-based tuning back and forth between D♭ and C (see Figure 4).

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

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



1950s "free-style" tuning, or what Polansky calls "paratactical" systems (Polansky 1987; 2018). 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 re-created digitally by David Code (Code 2002). In the 1970s, Harold Waage created a hardware system, called the intelligent keyboard, that adaptively tuned while being 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; 1994). The Hermode system (www.hermode.com) is available in Apple Logic Audio and Cubase, and it works reasonably transparently, optimizing the tuning of sonorities and intervals in constrained ways. It is static, however, in that one must 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. Modartt's Pianoteq software 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.modartt.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's "spring" tuning, as detailed by William 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 with which one can play, sometimes exhibiting unanticipated or decidedly "badly tuned" behaviors. As Douglas Repetto (2011) argues in "Doing It Wrong," "creative acts require deviations from the norm and . . . creative progress is born not of optimization but of variance."

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), whereas in the second, *anchored*, system the new fundamental is tuned to a given background scale.

Consider a system that uses a common scale in just intonation 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 is the deviation from ET in cents. Note that the second C is a syntonic comma sharper (81:80) than the first C, because a just-tuned major sixth (5:3) is actually smaller than a just-tuned sequence of perfect fifth (3:2) and major second

Figure 5. Simple adaptive tuning. Each new note becomes the fundamental for the tuning of the next note, resulting in tuning drift between the middle Cs. (Audio Example 5.)

Figure 6. Simple adaptive tuning with prepared resets. In this case the second middle C resets intonation to the first C, stretching the descending major sixth, and causing the second A to be detuned to the first A. (Audio Example 6.)

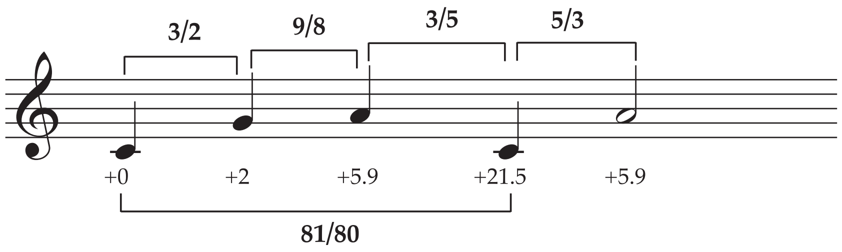


Figure 5

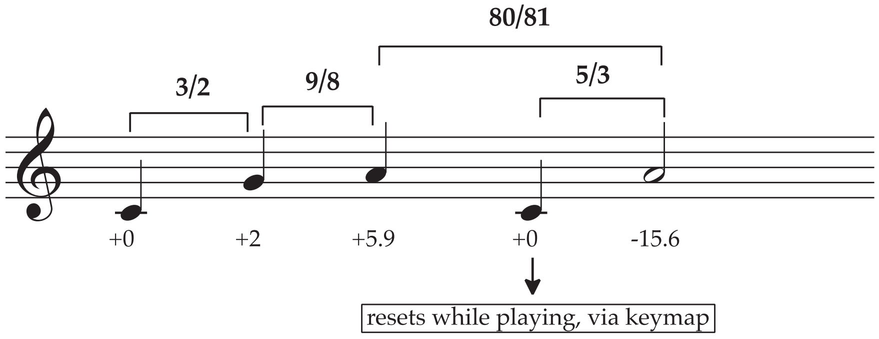


Figure 6

(9:8): 3/2 × 9/8 = 27/16 = 5/3 × 81/80. This phenomenon, where sequences of different intervals in just intonation 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 companion article in this issue for further details). The same melody would be tuned a bit differently now, as shown in Figure 6. This time, the Cs are tuned the same, and the As take on a different tuning (also by a syntonic comma).

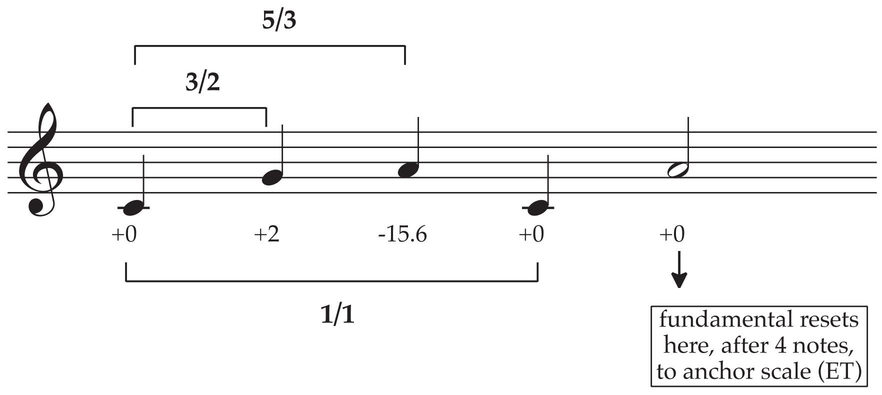
Now, let's consider the second flavor of bitKlavier's adaptive tuning, the "adaptive anchored tuning." Here, the fundamental is reset to a specified anchor scale, rather than the tuning of the last note, after either: (1) a prescribed set of notes have been played, or (2) a prescribed period of time 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.

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).

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!

Figure 7. Adaptive anchored tuning. In this example, after four notes bitKlavier resets the "fundamental" to the ET value of the written note, A. (Audio Example 7.)

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



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

Figure 7

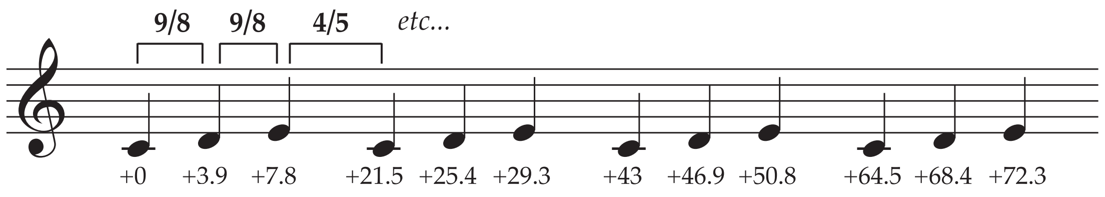


Figure 8

Consider for example this passage from "Sinking Song" in *Songs That Are Hard to Sing* (see Figure 9).

Here, the two bitKlaviers gradually "pump" upwards, out of phase with one another, being periodically reset by the middle Cs; 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 limitations at the time. More recently, and concurrent with our own developments here, Stange, Wick, and Hinrichsen (2018) created a powerful adaptive system akin to that of de Laubenfels, and it is one we plan to incorporate into bitKlavier in the future. Our own system has some fundamental differences in intent and implementation from both Stange et al. and de Laubenfels that 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 or 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 that 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 in just intonation). 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 9. Example of adaptive tuning drift in Trueman's Songs That Are Hard to Sing. *(Audio Example 9.)*

Figure 10. The spring-based model used to effect a "compromise" between conflicting interval distances in just intonation. In this example, the springs in a major triad are all of compatible lengths (3/2 × 5/3 = 5/2) and no further adjustment is necessary.

Figure 11. Incompatible spring lengths in an augmented triad (8/5 × 8/5 != 5/2). The spring model will find a solution to the incompatibility.*

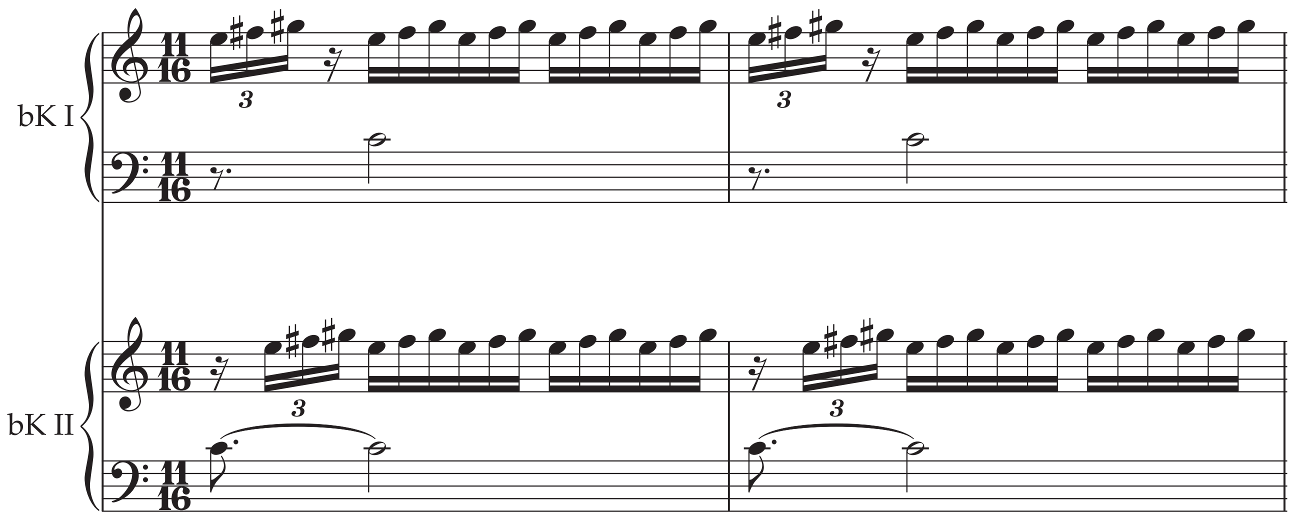


Figure 9

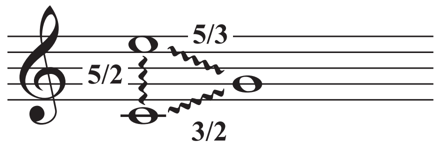


Figure 10

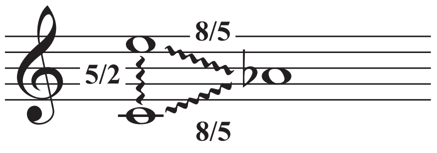


Figure 11

But for other sonorities, it's mathematically impossible for all the intervals to be in just intonation. 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 major-tenth spring would have to stretch, while the minor-sixth springs would have to compress (note that because this is a keyboard-centric system, we don't distinguish between enharmonic intervals, such as minor sixth and augmented fifth). Some spring strengths might be set more strongly than others, so if the major-tenth spring is stronger than the minor-sixth spring, the tempering will favor the major tenth, and so on.

Implementation of Spring Tuning

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 or 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 physics engine (http://haptic-data.com/toxiclibsjs) to implement verlet springs. Even in a relatively computationally inefficient language like JavaScript, we can implement hundreds of springs connecting together dozens of notes, in real time, 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 spring models that are more complex, dispersing "energy" over time as they oscillate, and they eventually settle into an equilibrium that minimizes the overall energy (or spring stretching and 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 and coworkers, in that we introduce "tether" or "anchor" springs that connect each note to a reference. In contrast to Stange and colleagues, who 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.

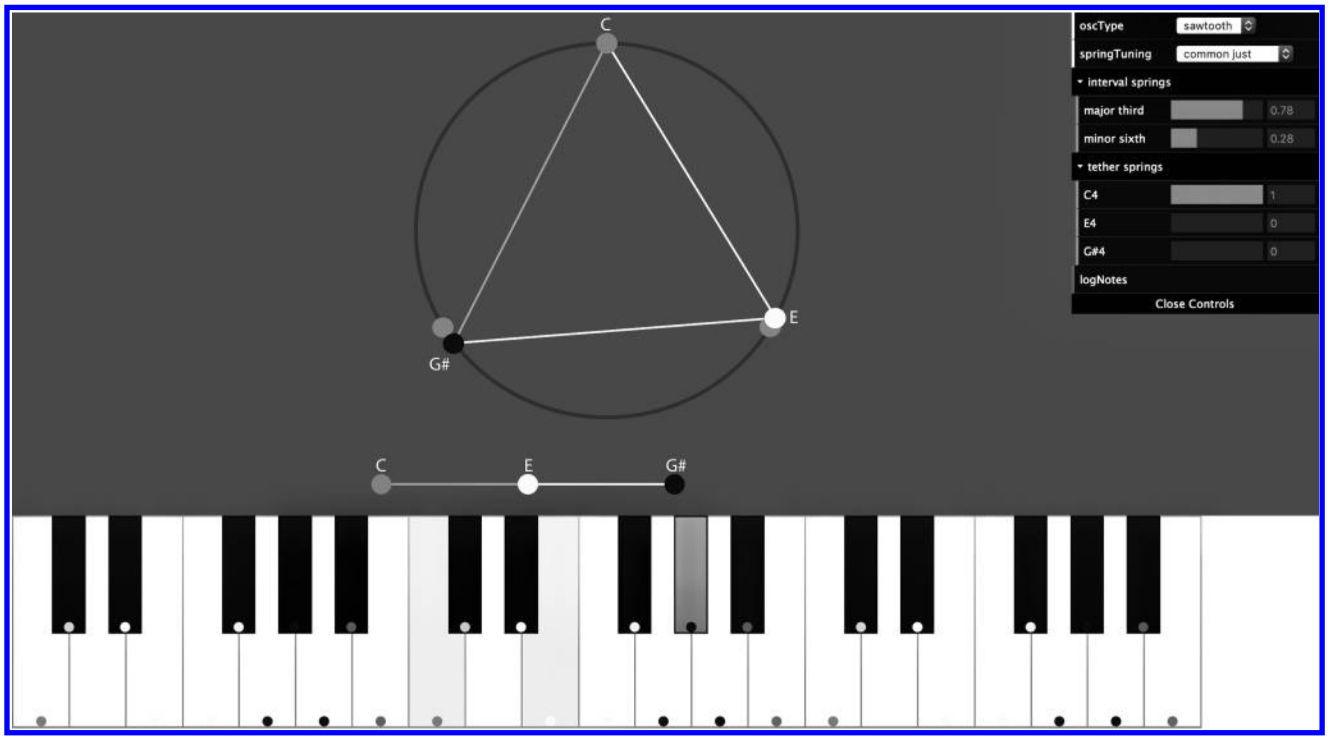
In the Appendix to this article, we minimize the spring energy equation in the simplified case of equal-strength springs, and demonstrate how equal temperament arises as a limiting case in three different spring tethering scenarios. Our verlet-based simulation matches these theoretical results to high accuracy, verifying that the simulation does indeed converge towards minimizing the spring energy. The verlet spring simulation is more general than the results in this Appendix, however, 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 that determines the relative weight of that interval, and how quickly the springs converge (less stiff springs will spend more time oscillating, stiffer springs will converge quickly). Rather than "optimizing" 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. In fact, in the JavaScript simulation, one can drag the notes with the 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♯/D♭ 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 springs for major thirds and one for a minor sixth). 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 (at the 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 major-third spring is set to be stronger than the minor-sixth spring, so this solution favors the third over the sixth. Modifying the spring strength causes the system to oscillate and respond audibly in real time. Again, our intent is to make a system with which it is engaging to play and that does not suppress "unwanted" behaviors, leaving them available for exploration.

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 4:7 minor seventh and a 5:7 augmented fourth (or diminished fifth). These intervals are sometimes referred to as "septimal," because of the inclusion of the prime number 7 in their ratios. (See Table 1 for

Figure 12. Prototype of spring tuning, in p5.js. Active pitches are displayed chromatically, clockwise on the circle, with C at the top. Gray circles indicate where the nearest equal-tempered pitches would be placed, and the black and the white circles show where the pitches are actually tuned. The sides of the triangle represent springs connecting the active pitches. Pitches on the straight line below can be dragged manually and then released, to see and hear how the springs respond. The piano keyboard at bottom can be used to add or subtract pitches from the system.



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

Table 1. Ratios and Rest Spring Lengths for Septimal Just Intonation

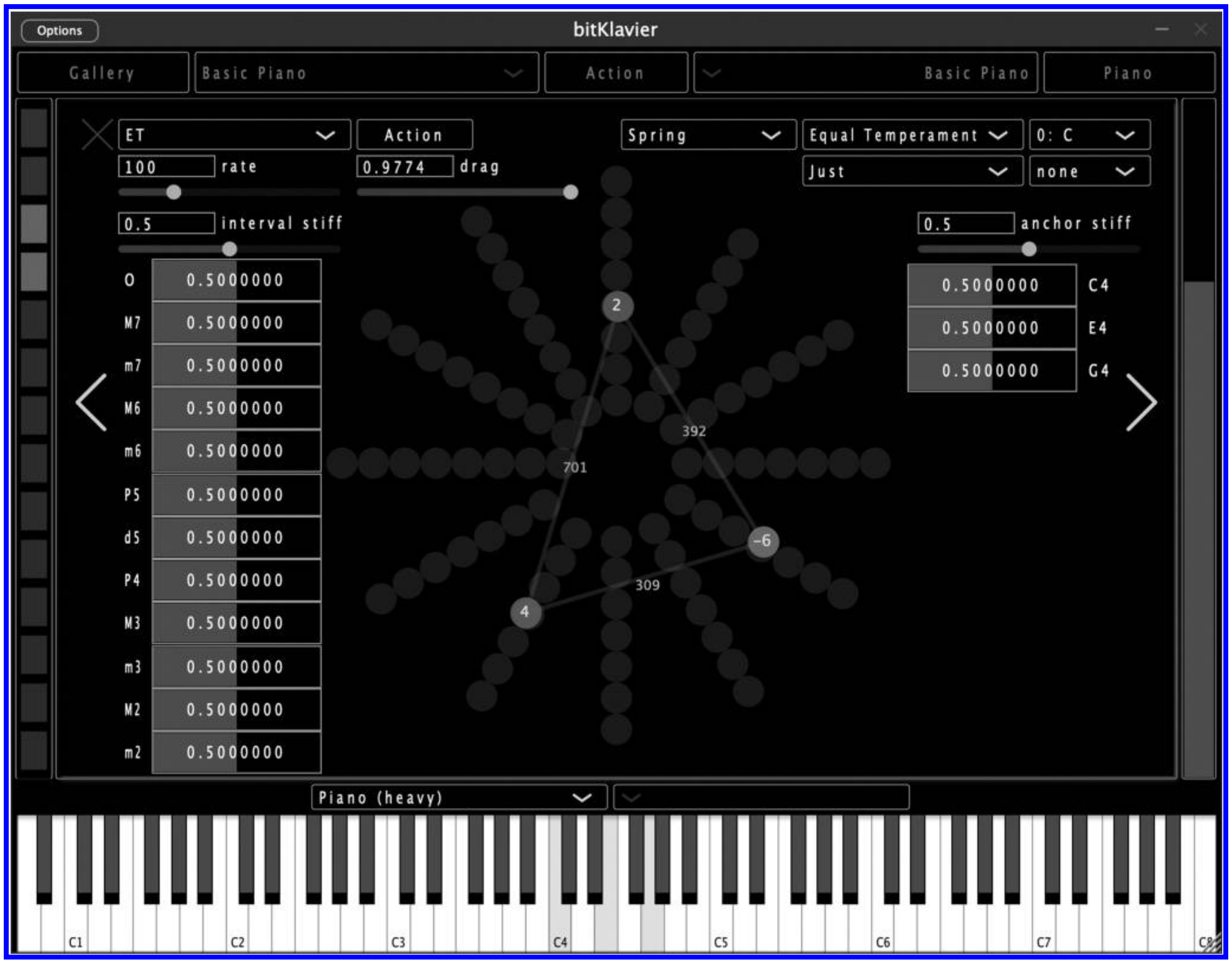
Interval	*Ratio*	*Spring Length (in cents)*
Unison	1:1	0
Minor Second	15:16	111
Major Second	8:9	204
Minor Third	5:6	316
Major Third	4:5	386
Perfect Fourth	3:4	498
Augmented Fourth/Diminished Fifth	5:7	582
Perfect Fifth	2:3	702
Minor Sixth	5:8	813
Major Sixth	3:5	884
Minor Seventh	4:7	968
Major Seventh	8:15	1088

the complete scale in just intonation.) The default unit for spring length is cents, and so, for instance, any two notes a major second apart will have a spring with resting length of 204 cents (and, by extension, 1404 cents for a major ninth, and so on; as of this writing, bitKlavier does not treat compound intervals differently from simple intervals).

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 Cs 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 offset in cents 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

Figure 13. *Spring-tuning interface in bitKlavier, 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

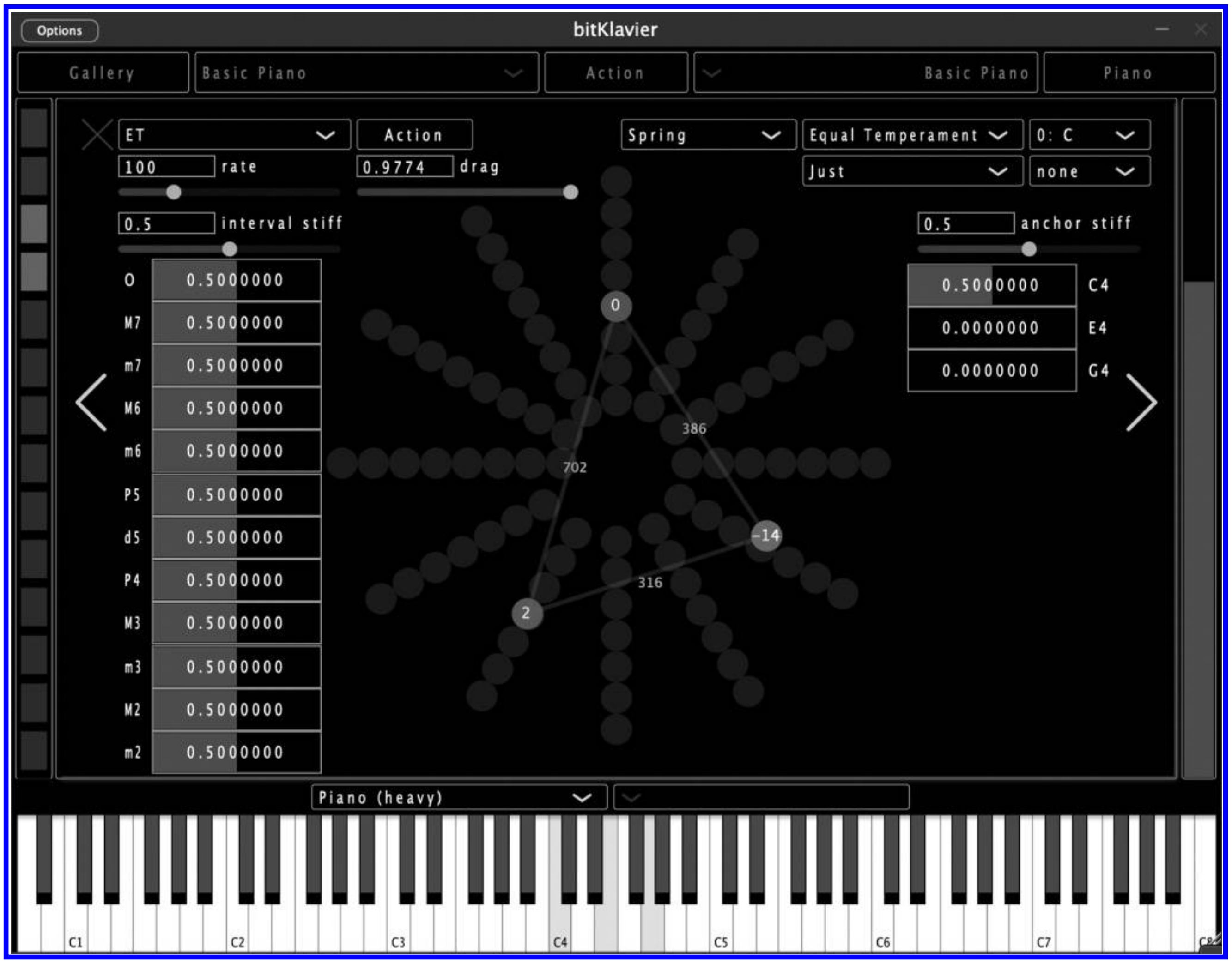
O	0.5000000
M7	0.5000000
m7	0.5000000
M6	0.5000000
m6	0.5000000
P5	0.5000000
d5	0.5000000
P4	0.5000000
M3	0.5000000
m3	0.5000000
M2	0.5000000
m2	0.5000000

0.5000000	C4
0.5000000	E4
0.5000000	G4

these springs are at their rest lengths. Why is this so, 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 triad in true just intonation, as shown in 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 interval sliders; strong major-third sliders will bring us closer to quarter-comma meantone temperament, which has pure 4:5 major thirds, whereas strong perfect-fifth sliders will bring us closer to Pythagorean tuning, which has pure 2:3 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

Figure 14. With only one anchor spring active (note that the E4 and G4 anchor sliders near the top right are now set to 0), the major triad is now tuned as in just intonation, in contrast to the example in Figure 13.



to two—e.g. 8:9 and 9:16 for the major second and minor seventh), holding down all twelve notes, and having all the spring strengths equal (see the Appendix for a mathematical derivation). Although 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.

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.

Nonunique 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"

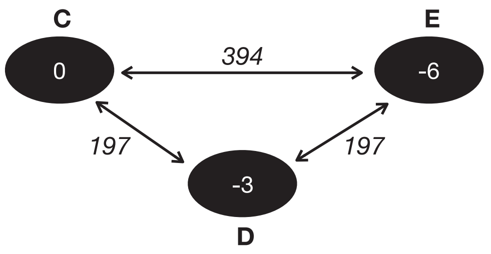


Figure 15. A major third with a whole step in between bottom and top notes. The major-second springs push the major third wide, while the major third pushes the whole-step springs narrow, leaving the springs stressed. (Audio Example 10.)

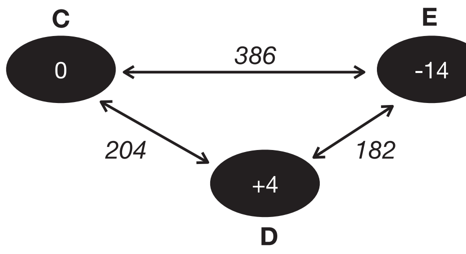


Figure 16. With C defined as the fundamental, the C–D and D–E whole steps take on different rest lengths, so this system can converge to a tuning with no stress in the springs. (Audio Example 11.)

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 major second spring will be set to 204 cents when using the just-intonation 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 major third a bit wide. In Figure 15, we see the major seconds are a bit compressed, and the major third is stretched (these steady-state values are taken directly from the bitKlavier simulation).

If we set the interval scale fundamental to C (instead of "none" in the popup menu), however, the D–E major second will use the 9:10 ratio (386 − 204 = 182 cents, from Table 1) of that just scale (which is the ratio between the major 2nd and 3rd in that scale: 8:9 × 9:10 = 4:5), so all the springs will naturally sit at their rest lengths (see 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 do not have unique interval sizes, and although this is a well-known issue (see e.g., Doty 1993), 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 spring for the major second or strengthen the spring for the major third. That would result in a more just-tuned third, 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.

The movable fundamental in bitKlavier allows us to retain some of the richness of nonunique 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, because 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 fifth, one for the familiar 2:3 ratio, and another for a "wolf" interval (27:40 between D and A in Table 1, when the fundamental is C), or the two different major seconds 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 removal 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 dynamically, on the fly. We also encounter sonorities that are systematically tuned differently depending on how they are played; consider how a dominant seventh chord would be tuned (in Last mode) when arpeggiated upwards, as opposed to downwards arpeggiation.

Another approach that springs afford is the mixing and matching of spring lengths based on a fundamental with lengths 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 27:40 "wolf" fifth between D and A, when the fundamental is C. Given the central role of the fifth in the overtone series, it would be reasonable to simply ask that all perfect fifths be tuned 2:3; similarly, the complement interval, perfect fourths, should be tuned 3:4. We might also want to avoid the Pythagorean major thirds (64:81) that sometimes crop up, replacing them with 4:5 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 the spring length (F) or whether the length is set locally (L; so if all springs are set to L, then the system will behave identically as if it were in the mode "none"). Although 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, although 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 into bitKlavier, all of which can be used as presets without concern for how they work under the hood.

The modes Highest, Lowest, and Last raise the question: Can bitKlavier not 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 seventh chord) it is a reasonable one. In Automatic 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 e.g., Giordano 2010; Heller 2013). In Figure 17, we see the overtone series based on C, and then its first few intervals, in complementary octave pairings, so the perfect fourth–fifth, the major third–minor sixth, and the minor third–major sixth pairs. The algorithm for Automatic mode works down from the top of whichever chord is sounding, looking for these intervals; if it finds any of them, it returns a fundamental of C. In the event that different intervals suggest different fundamentals, the algorithm prioritizes perfect fifths and fourths over major thirds and minor sixths, and similarly major thirds and minor sixths are prioritized over minor thirds and major sixths. And by working from the top down, returning the last fundamental found, it prioritizes lower register voices. Imagine a stack of perfect fifths, the bottom fifth would determine what the fundamental is. If none of these intervals are found, then it leaves the fundamental unchanged.

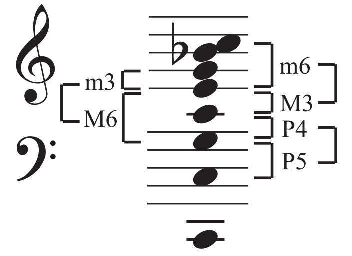
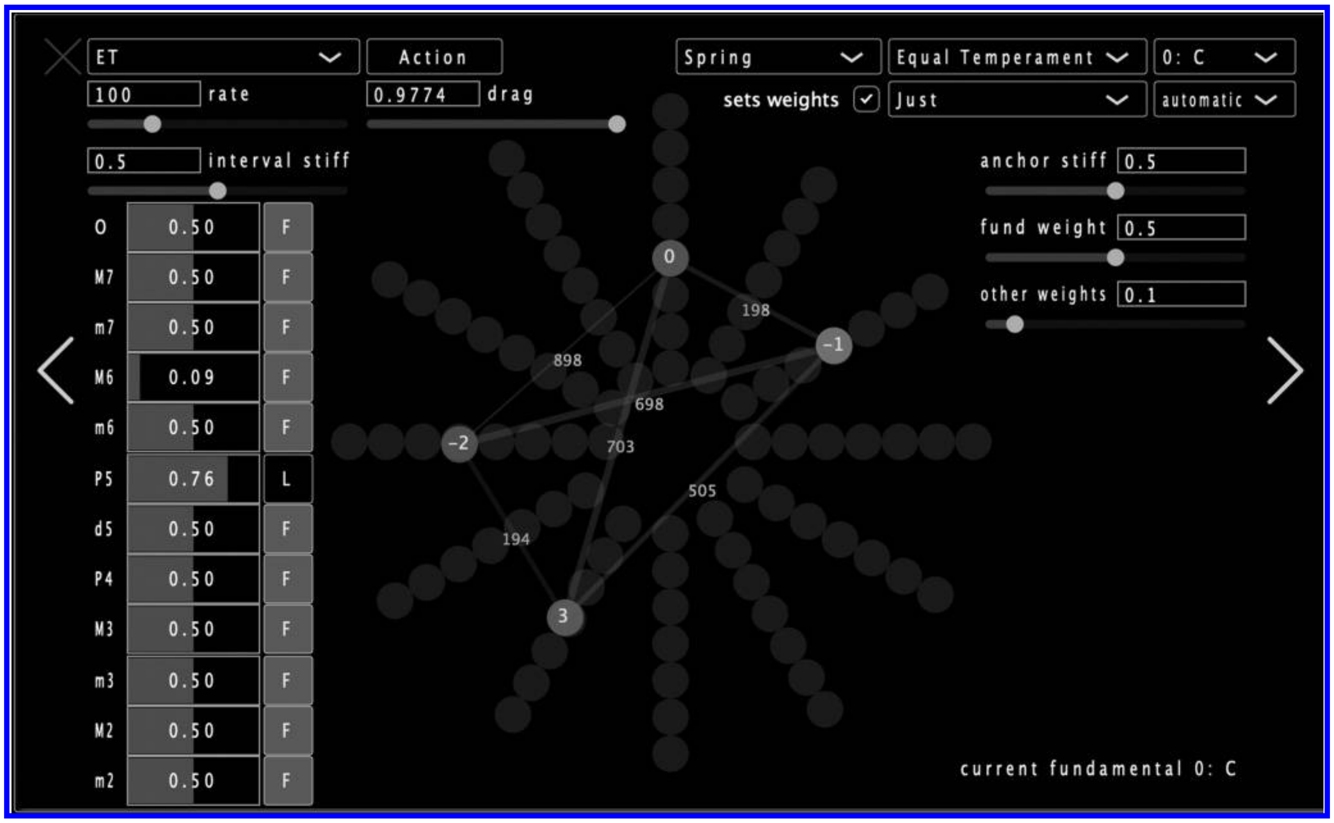


Figure 17. Intervals in the overtone series used to automatically determine a reasonable fundamental for the currently active notes. The algorithm works from the top down, favoring fifths and fourths over thirds and sixths when the intervals would point to different fundamentals.

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, and yet others that work with Sethares's (2005) "spectral dissonance" measures, all of which we are examining. 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 seventh chord, and so on. It is a starting point.

One final component of 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 such as Last and Automatic, enabling the anchor springs to track the changing fundamentals. In Figure 18, we see this and both the automatic


ET | Action | Spring | Equal Temperament | 0: C

100 rate

0.9774 drag

sets weights ✓ | Just | automatic

0.5 interval stiff

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

anchor stiff 0.5

fund weight 0.5

other weights 0.1

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 perfect fifth set to Local mode and weighted more heavily than the major sixth. 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. (Audio Example 12.)

fundamental finding and the setting for Local or Fundamental interval springs, on a stack of perfect fifths, C–G–D–A.

Here, beause 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 perfect fifth set to Local mode (indicated by the letter "L"), the D–A perfect fifth spring is set to the typical 702 cents, rather than the 680 cents that the Fundamental mode ("F") would define. This creates a tension between the C–A major sixth spring and the D–A perfect fifth spring, which is weighted to favor the perfect fifth, so the fifths in this sonority will be close to 2:3 ratios. Remember that this is a dynamic system, though, so the major sixth won't always be compromised in this way. For instance, if we simply release the D, the C–A major sixth will shrink down closer to its normal 5:3 884 cents length.

Comparisons with Other Systems

Stange et al.'s adaptive system addresses interval sizes that are not unique by referencing a small table of common variations on particular intervals; when confronted with, say, a major second, 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. Because 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-and-match spring system. It remains unclear, however, how the Stange et al. system determines where to locate nonunique intervals in a particular sonority; for instance, in the C–D–E example of Figures 15 and 16, it's not clear how their system determines the whole step with which it uses the alternative ratio, since either option will result in identical stretching. One advantage of the moving fundamental approach here is that it orders the nonunique intervals according to the structural logic of the tuning systems themselves, which reflect particular priorities.

Both the de Laubenfels and Stange et al. 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 and coworkers build a model of "intonational memory," a damped decaying function with an approximately 3-sec decay factor (chosen on the basis of psychoacoustic studies), and then uses this for "horizontal" adaptive tuning, corresponding roughly to the horizontal de Laubenfels springs, although 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 already constrains the system sufficiently, 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 et al. system in intent. Stange and colleagues' 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 et al. 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* of the total pitch space" (original italics). 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, an approach in line with our own thinking here. For a fascinating exploration of this in Tenney's work, see Michael Winter's (2008) detailed article about Tenney's last composition, *Arbor Vitae*.

Discussion

> Every tuning system makes an ideology audible, however (un)justly it might strike our ears (Moseley 2016, p. 108).

The longstanding 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 (1:2) and the perfect fifth (2:3)—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. Equal temperament 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). This compromise enabled the expansive modulating tradition of 19th-century tonal music, as well as the extended triadic harmonies of jazz, among much else. But, as has been well established, this was not the tuning of J. S. Bach and his well-tempered clavier (cf. e.g., Swich 2011), and is most certainly not the tuning of Indian classical music, Indonesian gamelan music, and countless other musics from around the world (cf. the many tunings explicated in e.g., Polansky 1987; Polansky et al. 2009; Forster 2010).

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 and the pentatonic scale of the black keys emerges; add another two, 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 gain broad acceptance, perhaps because their complexity seems daunting, both for our hands and for our 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. Although 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. Ross Duffin (2008), invoking William James, implores us to embrace the "booming, buzzing confusion" of this in-between music. 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 2016, 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 Etude No. 4, "Marbles," from *Nostalgic Synchronic*). 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.

Acknowledgments

We are grateful to Jeff Snyder for his feedback on the manuscript 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: 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.

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(1–2):458–503.

Giordano, N. 2010. *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: Random House.

Jakobsen, T. 2003. "Advanced Character Physics." Report. Copenhagen: IO Interactive. Reprint available 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 California, 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.

Nicholson, T., and M. Sabat. 2018. "Fundamental Principles of Just Intonation and Microtonal Composition." Report. Berlin: Universität der Künste. Available online at 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.

Polansky, L. 2018. "A Few Words About Tuning." *Sound American* 20. Available online at archive.soundamerican.org/sa_archive/sa20/sa20larrypolanskyafewwords.html. Accessed January 2020.

Polansky, L., et al. 2009. "A Mathematical Model for Optimal Tuning Systems." *Perspectives of New Music* 4(1):69–110.

Reinhard, J. 2009. *Bach and Tuning*. Bern: Peter Lang.

Repetto, D. 2011. "Doing It Wrong." *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2010 Symposium*. Washington, D.C.: National Academies Press.

Sabat, M., and W. Schweinitz. 2004. "The Extended Helmholtz-Ellis JI Pitch Notation." Technical Report. Available online at www.marcsabat.com/pdfs/notation.pdf. Accessed 3 April 2019.

Sethares, W. 2005. *Tuning, Timbre, Spectrum, Scale.* Berlin: Springer.

Stange, K., C. Wick, and H. Hinrichsen. 2018. "Playing Music in Just Intonation: A Dynamically Adaptive Tuning Scheme." *Computer Music Journal* 42(3):47–62.

Swich, L. 2011. "Further Thoughts on Bach's 1722 Temperament." *Early Music* 39(3):401–408.

Trueman, D., and M. Mulshine. 2020. "Preparing the Digital Piano: Introducing bitKlavier." *Computer Music Journal* 43(2–3):48–66.

Waage, H. 1985. "The Intelligent Keyboard." *1/1: 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* 27(1):131–150.

Wooley, N. 2018. "The Just Intonation Issue." *Sound American* 20. Available at archive.soundamerican.org/sa_archive/sa20/sa20fromtheeditor.html. Accessed January 2020.

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 (the first of which allows for two variants). We have compared these derivations with the actual behavior of the spring-tuning system in bitKlavier and found them to be sufficiently close approximations. 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 the system. Also, here we assume equal spring strengths, to simplify the derivations. The actual system in bitKlavier does not have this simplification.

Spring System with One Note Fixed

If just one note is fixed in a spring system, we have two approaches. First, we can aim to minimize the energy in the spring system. Alternately, we can add the additional constraint of assigning a reference pitch to one note.

Minimizing the Energy

We want to minimize the energy in the spring system, given by

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

where 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 strength or spring constant.

There are N notes, so the i and j indices range from 0 to $N-1$. To avoid double counting, we are only considering terms in the sum where j exceeds i. For simplicity, we are also assuming here that all the springs are of equal strength, i.e., $\omega_{ij} = \omega$.

To minimize the energy, we take the derivative of Equation 1 with respect to each coordinate and set it equal to zero, giving a set of N equations. We can interpret these equations physically by noting that $-\partial E/\partial x_k$ is the force exerted by the spring system on each note, and so we are solving for the equilibrium condition where the force on each note is zero.

Taking the derivative,

$$\frac{\partial E}{\partial x_k} = \sum_{\substack{i,j \\ j>i}} \omega(x_j - x_i - I_{ij})(\delta_{jk} - \delta_{ik}) = 0,$$

where we have used the fact that $\partial x_i/\partial x_k = \partial_{ik}$, whereby δ_{ik} is the Kronecker Delta function, which equals 1 when $i = k$ and 0 otherwise.

Simplifying further,

$$\begin{aligned} 0 &= \sum_{i<k}(x_k - x_i - I_{ik}) - \sum_{j>k}(x_j - x_k - I_{kj}) \\ &= \sum_{i<k}(x_k - x_i - I_{ik}) - \sum_{i>k}(x_i - x_k - I_{ki}) \\ &= (N-1)x_k - \sum_{i\neq k} x_i - \sum_{i<k} I_{ik} + \sum_{i>k} I_{ki} \\ &= Nx_k - \sum_i x_i + \phi_k \end{aligned}$$

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

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

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

Summing Equation 2 over the k index, we see that

$$\begin{aligned} 0 &= \sum_k \left(Nx_k - \sum_i x_i + \phi_k \right) \\ &= N\sum_i x_i - N\sum_i x_i + \sum_k \phi_k \end{aligned}$$

$$\implies \sum_k \phi_k = 0. \tag{4}$$

Notice that because we have made no assumptions about the intervals, Equation 4 is always true, irrespective of the specific choice of intervals. This identity will be useful later.

Adding a Constraint

The set of equations defined in Equation 1 are not uniquely solvable unless we add an additional constraint. This is because we have not yet set a reference pitch. We can do this by simply fixing the value of a single note (i.e., setting its pitch to a constant). Physically, this corresponds to a spring system where one mass is locked in place.

Say we fix x_0. Rearranging Equation 2 for $k = 0$

$$\sum_i x_i = N x_0 + \phi_0,$$

And substituting this value of $\sum_i x_i$ in Equation 2 gives

$$x_k = x_0 + \frac{\phi_0 - \phi_k}{N}, \tag{5}$$

which is the solution that minimizes the energy. Here x_k is the pitch of each note, in cents, x_0 is the fixed referenced note, and ϕ_k are constants defined in Equation 3 (or, later, in Equation 11) in terms of the provided intervals.

Some algebraic manipulation (detailed later at Equation 12) shows that

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

which allows us to re-express the solution as

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

If the intervals are chosen to be symmetric, then complementary intervals add to an octave or 1,200 cents, i.e., $I_i + I_{N-i} = 1200$. In this case, the solution simplifies to $x_k = x_0 + 1200k/N$, i.e., we recover equal temperament.

Spring System with Multiple Tethers

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 (e.g., the equal temperament pitch of each note). We index these reference pitches by x_i^{ET}.

So we have a new term in our energy equation for these tether springs.

$$E = \sum_{\substack{i,j \\ j>i}} \frac{1}{2} \omega (x_j - x_i - I_{ij})^2 + \sum_i \frac{1}{2} \epsilon (x_i - x_i^{ET})^2. \tag{6}$$

Once again, we assume that the interval springs all have the same strength ω, and the tether springs have the same strength ϵ. Typically we are working in the regime where $\omega \ll \epsilon$.

Minimizing the energy (or setting forces to zero) as before,

$$\frac{\partial E}{\partial x_k} = \sum_{\substack{i,j \\ j>i}} \omega (x_j - x_i - I_{ij})(\delta_{jk} - \delta_{ik}) + \sum_i \epsilon (x_i - x_i^{ET}) \delta_{ik} = 0.$$

Simplifying,

$$0 = \omega \left(\sum_{i<k} (x_k - x_i - I_{ik}) - \sum_{j>k} (x_j - x_k - I_{kj}) \right) + \epsilon (x_k - x_k^{ET}).$$

Expanding out the sums and simplifying as in Equation 2, we get

$$0 = \left(N x_k - \sum_i x_i + \phi_k \right) + \frac{\epsilon}{\omega} (x_k - x_k^{ET}), \tag{7}$$

where ϕ_k is a known function of the intervals, defined earlier in Equation 3. To solve, we sum over k.

$$0 = N\sum_i x_i - N\sum_i x_i + \sum_k \phi_k + \frac{\epsilon}{\omega}\left(\sum_k x_k - \sum_k x_k^{ET}\right)$$
$$= \sum_k \phi_k + \frac{\epsilon}{\omega}\left(\sum_k x_k - \sum_k x_k^{ET}\right)$$

Recall that $\sum_k \phi_k = 0$ (Equation 4). So, this reduces to

$$\sum_i x_i = \sum_i x_i^{ET},$$

that is, the constraint ensures that, once the energy is minimized, the average value of the notes equals the average value of the reference notes. Substituting this into Equation 7, we arrive at the solution

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

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 if the tether springs are not very weak, the resultant intervals will shrink due to the ϵ/ω term in the denominator.

In the limit of very weak tethers ($\epsilon \ll \omega$ or $lim_{\frac{\epsilon}{\omega}\to 0}$), this 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, $\phi_k = \phi_0 - 1200k$ (see the Section on "Interval Algebra"), so we recover equal temperament.

Spring System with Tethers to a Single Reference Note

Here, instead of tethering each note to a separate reference pitch, we tether them all to the same reference pitch x^*, following the approach of Stange, Wick, and Hinrichsen (2018). This is a special case of the previous configuration, Equation 6, with $x_i^{ET} = x^*$. Substituting this into Equation 9 gives

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

yielding the solution

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

Once again we have an interval stretching if the tethers are not very weak. 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. If the intervals are symmetric, $\phi_k = \phi_0 - 1200k$ and we recover equal temperament.

Interval Algebra

All of our solutions are expressed in terms of ϕ_k, a set of constants defined in terms of the provided intervals:

$$\phi_k = -\sum_{i<k} I_{ik} + \sum_{i>k} I_{ki}.$$

We can simplify the above expression by noting that our intervals have the form $I_{ij} = I_{j-i}$, i.e., the intervals depend only on the difference of indices. Making this change,

$$\phi_k = -\sum_{i=0}^{k-1} I_{k-i} + \sum_{i=k+1}^{N-1} I_{i-k}$$
$$= -\sum_{j=1}^{k} I_j + \sum_{j=1}^{N-k-1} I_j, \tag{11}$$

where, for clarity, we have switched to indices $j = k - i$ for the first sum and $j = i - k$ for the second sum.

We can see from the above equation that for $k = 0$,

$$\phi_0 = \sum_{j=1}^{N-1} I_j.$$

So, subtracting ϕ_0 from ϕ_k,

$$\phi_k - \phi_0 = -\sum_{j=1}^{k} I_j + \sum_{j=1}^{N-k-1} I_j - \sum_{j=1}^{N-1} I_j$$

$$= -\left(\sum_{j=1}^{k} I_j + \sum_{j=N-k}^{N-1} I_j\right)$$

$$= -\sum_{j=1}^{k} (I_j + I_{N-j}). \qquad (12)$$

If the intervals are symmetric, $I_j + I_{N-j} = 1200$, so $\phi_k = \phi_0 - 1200k$.