Journal of New Music Research
2002, Vol. 31, No. 1, pp. ••–••

0929-8215/02/3101-001$16.00

Real-Time Adaptive Tunings Using Max

William A. Sethares

Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI, USA

Abstract

Adaptive tunings adjust the pitches of notes in a musical performance based on some criterion such as fidelity to a desired set of intervals, or minimization of a measure of dissonance (or maximization of consonance) of the currently sounding notes. This paper presents a real-time implementation of an adaptive tuning algorithm that changes the pitches of notes in a musical performance so as to minimize sensory dissonance. The algorithm operates in real time, is responsive to the notes played, and can be readily tailored to the timbre (or spectrum) of the sound. One issue with adaptive retunings is that the pitches may wander; the pitch of a "C" note one time may differ from the pitch of the "same C" at another time. This is addressed in Adaptun by the use of a *context*, an (inaudible) collection of partials that are used in the calculation of dissonance within the algorithm, but that are not themselves adapted or sounded. Several sound examples are presented that highlight both strengths and weaknesses of the implementation. The program Adaptun is written in the Max language and is available for download.

1. Introduction

A musical scale typically consists of a fixed, ordered set of intervals that (along with a reference frequency such as A = 440 Hz) define the pitches of the notes used in a given piece. Numerous scales have been proposed through the years including Meantone, Pythagorean, various Just Intonations, scales by Werkmeister, Vallotti and Young, Partch, Carlos, and various subsets of 12-tone equal-temperament such as the major and minor scales [28]. The use of fixed musical scales is not confined to western music: for instance, the pelog and slendro scales of Indonesian gamelan [8], the (roughly) 7-tone equal division of traditional Thai music, and the various multi-toned scales of Indian and Arabic musics.

A recent innovation is the idea of an *adaptive tuning*, one that allows the tuning to change dynamically as the music is performed. The trick is to specify criteria by which the retuning occurs. Carlos [2] and Hall [5] have introduced quantitative measures of the ability of fixed scales to approximate a desired set of intervals. Since different pieces of music contain different intervals, and since it is mathematically impossible to devise a single fixed scale in which all intervals are perfectly tuned, Hall [5] suggests choosing tunings based on the piece of music to be performed. Polansky [15] suggests the need for a "harmonic distance function" which can be used to make automated tuning decisions, and points to Wagge's [27] "intelligent keyboard" which utilizes a logic circuit to automatically choose between alternate versions of thirds and sevenths depending on the musical context. More recently, Denckla [4] has extended this idea by using sophisticated tables of intervals that define how to adjust the pitches of the currently sounding notes given the musical key of the piece, and DeLaurentis [3] has proposed a spring-mass paradigm that models the tension between the currently sounding notes (as deviations from an underlying just intonation template) and adapts the pitches to relax the tension.

An alternative criterion based on the psychoacoustic notion of *sensory dissonance* was proposed in [19]. Helmholtz [6] attributes the perception of sensory dissonance to the beating between partials of a sound. Plomp and Levelt [14] extend this to show that regions of maximal beating correspond to the critical band. Sethares provides a simple computational model in [18] that gives a numerical measure of sensory dissonance as a function of the tuning (fundamental pitch) and timbre (spectrum) of the currently sounding notes. This model is used to define a "cost" function for a gradient-based optimization procedure in [19]. Minimizing this cost adjusts the pitches automatically so as to mimize the sensory dissonance (or equivalently, to maximize the sensory consonance). This can be viewed as a generalization of the methods of Just Intonation, but it can operate without specifically musical knowledge such as key and tonal center, and is applicable to timbres with inharmonic spectra as well as the more common harmonic timbres.

Correspondence: William A. Sethares, Department of Electrical and Computer Engineering, 1415 Engineering Drive, University of Wisconsin-Madison, Madison, WI 53706, USA. E-mail: sethares@ece.wisc.edu

This paper dicusses several simplifications to the algorithm that allow real time implementation, and describes the resulting program, Adaptun, which is written in the musical programming language Max [29]. The inputs to Adaptun are the spectra of the sounds and a MIDI data stream. At each moment the program calculates the amount of sensory dissonance caused by the currently sounding notes and then changes their pitches in the direction that minimizes the dissonance. Over time, the pitches stabilize to form intervals that are more consonant (less dissonant) than all surrounding intervals. When new notes are added or removed, the program reacts by readapting, continuously retuning towards intervals and chords that minimize the sensory dissonance. Adaptun extends and modifies the original algorithm in several ways to provide a more useful musical tool. It may be downloaded from [1].

Section 2 reviews the Plomp and Levelt model of sensory dissonance [14] as parameterized by Sethares [18]. The gradient-based algorithm for adaptive tuning is then reviewed in Section 3, and several alternative implementations are suggested. The program Adaptun is discussed at length in Section 3.2, and both its strengths and limitations are highlighted. One of the key new features is the addition of a "memory," a primitive way to implement a kind of musical "context" (in Section 4) that may make the algorithm more musically useful. Section 5 provides several examples that demonstrate the performance of the algorithm. Many of the examples are available in mp3 format on the web. Parts of this paper were presented at the 6th annual meeting of the Research Society for the Foundations of Music, in Ghent, Belgium, see [22].

2. Calculation of sensory dissonance

Measures of sensory dissonance are typically stated in terms of interactions between the sine wave partials of a sound; the dissonance between all pairs of partials are combined additively to give the sensory dissonance of the sound. The psychoacoustic work of Plomp and Levelet [14] provides a basis on which to build such a measure. (Related work can be found in Terhardt [25,26] and Parncutt [11,12]). Leman [9] has recently generalized the models to operate directly on digitized sound files (rather than on the partials of the sound), and research continues to improve the match between experimental results and models of sensory dissonance.

Plomp and Levelt asked volunteers to rate the perceived dissonance of pairs of pure sine waves, giving curves such as in Figure 1, in which the dissonance is minimum at unity, increases rapidly to its maximum somewhere near one quarter of the critical bandwidth, and then decreases steadily back towards zero. When considering sounds with spectra that are more complex, dissonance can be calculated by summing up all the dissonances of all the partials, and weighting them according to their relative magnitudes. This leads to dissonance curves such as Figure 2 which shows the

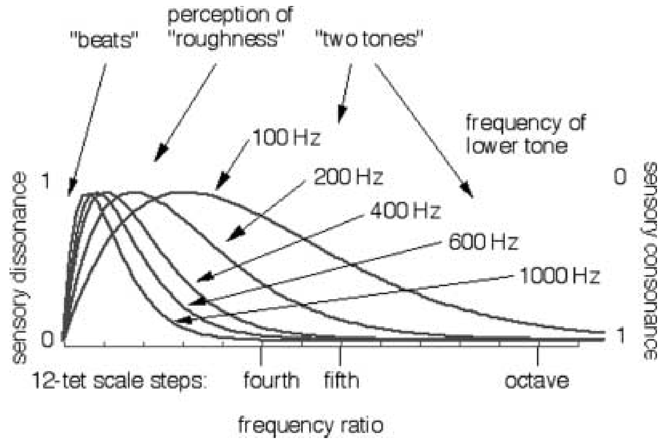

Fig. 1. Two sine waves are sounded simultaneously. Typical perceptions include pleasant beating (at small frequency ratios), roughness (at middle ratios), and separation into two tones (at first with roughness, and later without) for larger ratios. The horizontal axis represents the frequency interval between the two sine waves, and the vertical axis is a normalized measure of sensory dissonance. The different plots show how the sensory consonance and dissonance varies depending on the frequency of the lower tone.

dissonance curve for a timbre with seven harmonic partials. Note that many of the valleys in Figure 2 correspond (roughly) to intervals in the diatonic scale, suggesting a relationship between musical scales and the most consonant intervals of the dissonance curve. Observe also that dissonance curves depend on the spectrum of the sound, and consequently may be used to describe sensory consonance and dissonance not only in traditional harmonic settings, but also in nontraditional, inharmonic musical settings. These ideas are explored in depth in [21].

To be concrete, the dissonance between a sinusoid of frequency f_1 with magnitude v_1 and a sinusoid of frequency f_2 with magnitude v_2 can be parameterized as

$$d(f_1, f_2, v_1, v_2) = v_1 v_2 \left[e^{-as|f_2 - f_1|} - e^{-bs|f_2 - f_1|}\right] \tag{1}$$

where

$$s = \frac{d^*}{s_1 \min(f_1, f_2) + s_2}, \tag{2}$$

$a = 3.5$, $b = 5.75$, $d^* = .24$, $s_1 = .021$ and $s_2 = 19$ are determined by a least squares fit. The magnitude term $v_1 v_2$ ensures that softer components contribute less to the total dissonance measure than those with larger magnitudes, d^* is the interval at which maximum dissonance occurs, and the s parameters in (2) allow a single functional form to smoothly interpolate between the various curves of Figure 1 by sliding the dissonance curve along the frequency axis so that it begins at the smaller of f_1 and f_2, and by stretching (or compressing) it so that the maximum dissonance occurs at the appropriate frequency. See [18] for a derivation, justification, and discussion of this model.

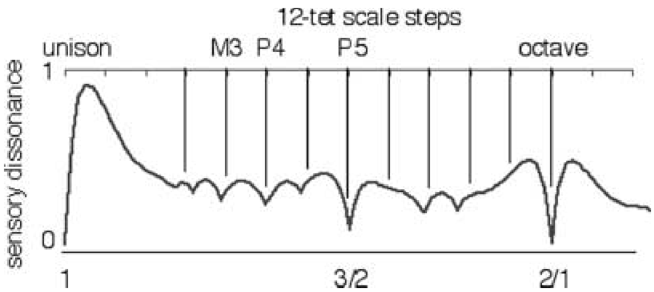

Fig. 2. Plots of the calculated dissonance of a spectrum vs. frequency interval are called dissonance curves. This figure shows the dissonance curve for a 7 partial harmonic spectrum. The minima of this curve occur at 1, 7/6, 6/5, 5/4, 4/3, 7/5, 3/2, 5/3, 7/4, and 2/1, which lie near many of the 12-tet scale steps (top axis). Dissonance values (on the vertical axis) are normalized to unity.

A line spectrum F representing a note with fundamental frequency f is a collection of n sine waves (or partials). The intervals between the partials are denoted by a_j with $a_1 = 1 < a_2 < \ldots < a_n$ so that the frequencies of the parials are $a_1 f < a_2 f < \ldots < a_n f$. The corresponding magnitudes are $v_1, v_2, \ldots, v_n$. The intrinsic dissonance of the sound F is the sum of the dissonances of all pairs of partials

$$D_F = \frac{1}{2}\sum_{k=1}^{n}\sum_{j=1}^{n} d(a_k f, a_j f, v_k, v_j). \tag{3}$$

More generally, suppose there are m different notes, each with line spectrum (timbre) F_i, fundamentals f_i, intervals between partials $a_{i1}, a_{i2}, \ldots, a_{in}$, and magnitudes $v_{i1}, v_{i2}, \ldots, v_{in}$. (If there are different numbers of partials in each timbre, set n to the maximum number and set the appropriate magnitudes v_{ij} to zero). Then the total sensory dissonance is the sum of the dissonances between all pairs of partials

$$D = \frac{1}{2}\sum_{l=1}^{m}\sum_{i=1}^{m}\sum_{k=1}^{n}\sum_{j=1}^{n} d(a_{ik} f_i, a_{lj} f_l, v_{ik}, v_{lj}). \tag{4}$$

3. Adaptive tunings

This section briefly reviews the adaptive tuning algorithm proposed in [19], and then introduces several modifications that allow real time implementation. The input is a stream of MIDI note-on events. These are retuned so as to minimize the sensory dissonance, and the output is a stream of MIDI note-on commands plus "pitch bend" commands that can be interpreted by most modern synthesizers and samplers. An implementation of the algorithm, written in the Max programming language, is currently available on our website at [1].

3.1. The adaptive algorithm

The sensory dissonance caused by m sounding notes, each with fundamental f_i and timbre F_i can be calculated as in (4). By making small adjustments to the tuning of the notes (to the f_i), it may be possible to decrease the dissonance. One approach is to adjust the current frequencies of the fundamentals in the (minus) direction of the gradient. The iteration is

$$\begin{Bmatrix}\text{new frequency}\\ \text{value } f_i(k+1)\end{Bmatrix} = \begin{Bmatrix}\text{old frequency}\\ \text{value } f_i(k)\end{Bmatrix} - \{\text{stepsize}\}\{\text{gradient}\} \tag{5}$$

where the *gradient* is an approximation to the partial derivative of D with respect to the i^{th} fundamental frequency, and k is the iteration counter. The behavior of this adaptive tuning algorithm is to continuously adjust the fundamentals of the notes so as to descend the m-dimensional landscape defined by D in (4).

To be explicit, the "cost" function D is defined to be the sum of the dissonances of all the intervals at a given time, and the iteration updates the f_i by moving "downhill." This is

do
  for $i = 1$ to m

$$f_i(k+1) = f_i(k) - \mu \frac{dD}{df_i(k)} \tag{6}$$

  endfor
until $|f_i(k+1) - f_i(k)| < \delta \forall i$

An explicit expression for the gradient term $\frac{dD}{df_i(k)}$ is given in [19]. Thus the frequencies of all notes are modified in proportion to the change in the cost and to the stepsize μ until convergence is reached, where convergence means that the change in all frequencies is less than some specified δ. Thus the iteration ceases when the gradient term is (approximately) zero, which occurs when the dissonance is at a (local) point of inflection. The minus sign insures that the algorithm descends to look for a local minimum of the dissonance (rather than ascending to a local maximum), that is, that the inflection point is a minimum rather than a maximum.

3.2. A real time implementation in Max

Figure 3 shows the main screen of the adaptive tuning program Adaptun. The user must first configure the program to access the MIDI hardware. This is done using the two menus labelled Set Input Port and Set Output Port, which list all valid (OMS) MIDI sources and destinations. The figure shows the input US-428 Port 1 which is my hardware, and the output is set to IAC Bus # 2, which is an OMS interapplication (virtual) port that allows MIDI data to be transferred between applications. The interapplication ports allow Adaptun to exchange data in real time with sequencers, software synthesizers, or other programs. In particular, the output of Adaptun can be recorded by setting the input of a MIDI sequencer to receive on the appropriate IAC bus.

In normal operation, the user plays a MIDI keyboard. The program rechannelizes and retunes the performance. Each

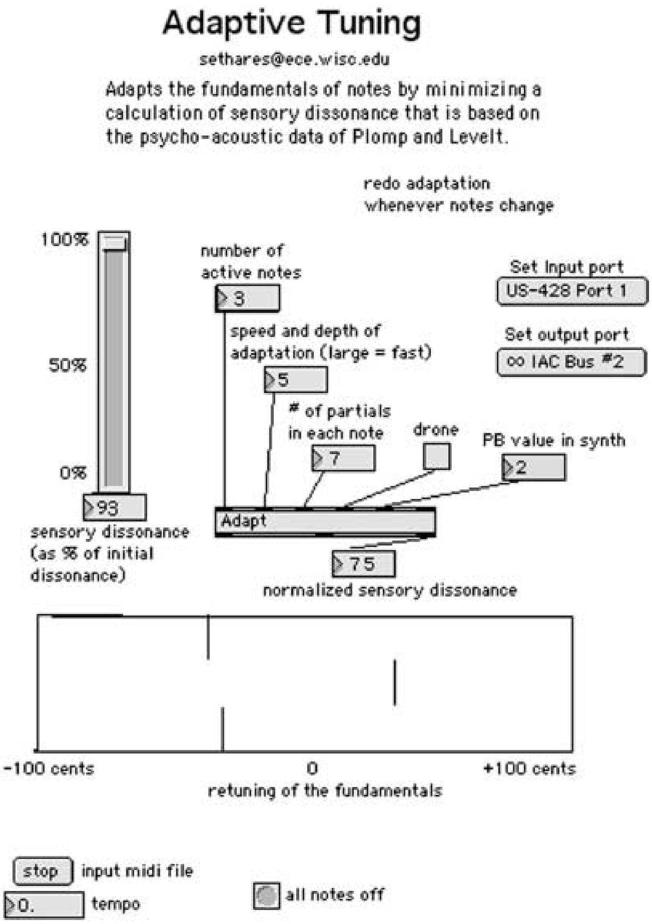

Fig. 3. Main screen of the adaptive tuning program Adaptun, implemented in the Max programming language.

currently sounding note is assigned a unique MIDI channel, and the adapted note and appropriate pitch bend commands are output on that channel. As the algorithm iterates, updated pitch bends commands continue to fine tune the pitches. The MIDI sound module must be set to receive on the appropriate MIDI channels with "pitch bend amount" set so that the extremes of ±64 correspond to the setting chosen in the box labelled PB value in synth. The finest pitch resolution possible is about 1.56 cents when this is set to 1 semitone, 3.12 cents when set to 2 semitones, etc.

There are several displays that demonstrate the activity of the program. First, the message box directly under the block labelled Adapt shows the normalized sensory dissonance of the currently sounding notes. The bar graph on the left displays the sensory dissonance as a percentage of the original sensory dissonance of the current notes. A large value means that the pitches did not change much, while a small value means that the pitches were moved far enough to cause a significant decrease in sensory dissonance. The large display in the center shows how many notes are currently adapting (how many pieces the line is broken into) and whether these notes have adapted up in pitch (the segment moves to the right) or down in pitch (the segment moves to the left). The screen snapshot in Figure 3 shows the adaptation of three notes; two have moved down and one up. There is a wrap-around in effect on this display; when a note is retuned more than a semitone, it returns to its nominal position. The number of actively adapting tones is also displayed numerically in the topmost message box.

The user has several options which can be changed by clicking on message boxes.[1] One is labelled speed and depth of adaptation in Figure 3. This represents the stepsize parameter μ from (5) and (6). When small, the adaptation proceeds slowly and smoothly over the dissonance surface. Larger values allow more rapid adaptation, but are less smooth. In extreme cases, the algorithm may jump over the nearest local minimum and descend into a minimum far from the initial values of the intervals. The relationship between the speed of adaptation and "real-time" is complex, and depends on the speed of the processor and the number of other tasks occurring simultaneously. The message box labelled # of partials in each note specifies the maximum number of partials that are used. (The actual values for the partials are discussed in detail in Section 3.3.)

There are two useful tools at the bottom of the main screen. The menu labelled input MIDI file lets the user replace (or augment) the keyboard input with data from a standard MIDI file. The menu has options to stop, start, and read. First, a file is read. When started, adaptation occurs just as if the input were arriving from the keyboard. The message box immediately below the menu specifies the tempo at which the sequence will be played. This is especially useful for older (slower) machines. A SMF can be played (and adapted) at a slow tempo and then replayed at normal speed, increasing the apparent speed of the adaptation. Finally, the all notes off button sends "note-off" messages on all channels, in the unlikely event that a note gets stuck.

3.3. The simplified algorithm

In order to operate in real time (actual performance depends on processor speed), several simplifications are made. These involve the specification of the spectra of the input sounds, using only a special case of the dissonance calculation, and a simplification of the adaptive update.

The dissonance measure in (4) is dependent on the spectra of the currently sounding notes, and so the algorithm (6) must have access to these spectra. While it should eventually be possible to measure the spectra from an audio source in real time, the current MIDI implementation assumes that the spectra are known *a priori*. The spectra are defined in a table, one for each MIDI channel, and are assumed fixed through-

[1] When a Max message box is selected, its value can be changed by dragging the cursor or by typing in a new value. Changes are output at the bottom of the box and incorporated into subsequent processing.

out the piece (or until the table is changed). They are stored in the collection[2] file timbre.col. The default spectra are harmonic with a number of partials set by the user in the message box on the main screen, though this can easily be changed by editing timbre.col. The format of the data reflects the format used throughout Adaptun; all pitches are defined by an integer

$$100 * (\text{MIDI Note Number}) + (\text{Number of Cents}). \quad (7)$$

For instance, a note with fundamental 15 cents above middle *C* would be represented as 6015 = 100 ∗ 60 + 15 since 60 is the MIDI note number for middle *C*. Similarly, all intervals are represented internally in cents: an octave is thus 1200 and a just major third is 386.

Second, the calculation of the dissonance is simplified from (4) by using a single "look-up" table to implement (1). A nominal value of 500 Hz is used for all calculations between all partials, rather than directly evaluating the exponentials. In most cases this will have little effect, though it does mean that the magnitude of the dissonances will be underestimated in the low registers and overestimated in the treble. More importantly, the amplitude parameters v_1 and v_2 are set to unity. Combined with the assumption of fixed spectra, this can be interpreted as implying that the algorithm operates on a highly idealized, averaged version of the spectrum of the sound.

The numerical complexity of the iteration (6) is dominated by the calculation of the gradient term, due to its complexity (which grows worse in high dimensions when there are many notes sounding simultaneously). One simplification uses an approximation to bypass the explicit calculation of the gradient. Adaptun adopts a variation of the Simultaneous Perturbation Stochastic Approximation (SPSA) method of [24], which is itself a variant of the classic Kiefer-Wolfowitz algorithm [7]. To be concrete, the function

$$g(f_i(k)) = \frac{D(f_i(k) + c\Delta(k)) - D(f_i(k) - c\Delta(k))}{2c\Delta(k)}$$

where $\Delta(k)$ is a randomly chosen Bernoulli ±1 random vector, can be viewed as an approximation to the gradient $\frac{dD}{df_i(k)}$ which grows closer in the limit as c approaches zero. The algorithm for adaptive tuning is then

$$f_i(k+1) = f_i(k) - \mu g(f_i(k)). \quad (8)$$

In the standard SPSA, convergence to the optimal value can be guaranteed if both the stepsize μ and the pertubation size c converge to zero at appropriate rates, and if the cost function D is sufficiently smooth [23]. In the case of adaptive tunings, it is important that the stepsize and perturbation size *not* vanish, since this would imply that the algorithm becomes insensitive to new notes as they occur.

[2] In Max, a "collection" is a text file that stores numbers, symbols and lists.

In the adaptive tuning application, there is a granularity to pitch space induced by the MIDI pitch bend resolution of about 1.56 cents. This is near to the resolving power of the ear (on the order of 1 cent), and so it is reasonable to choose μ and c so that the updates to the f_i are (on average) roughly this size. This is the strategy followed by Adaptun, though the user chooseable parameter labeled speed and depth of adaptation gives some control over the size of the adaptive steps. Convergence to a fixed value is unlikely when the stepsizes do not decay to zero. Rather, some kind of convergence in distribution should be expected, although a thorough analysis of the theoretical implications of the fixed-stepsize version of SPSA remain unexplored. Nonetheless, the audible results of the algorithm are vividly portrayed in Section 5.

4. Context, persistence, and memory

Introspection suggests that people readily develop a notion of "context" when listening to music and that it is easy to tell when the context is violated, for instance, when a piece changes key or an out-of-tune note is performed. While the exact nature of this context is a matter of speculation, it is clearly related to the memory of recent sounds. It is not unreasonable to suppose that the human auditory system might retain a memory of recent sound events, and that these memories might contribute to and color present perceptions. There are examples throughout the psychological literature of experiments in which subjects' perceptions are modified by their expectations, and we hypothesize that an analogous mechanism may be partly responsible for the context sensitivity of musical dissonance.

Three different ways of incorporating the idea of a musical context into the sensory dissonance calculation were suggested in [22], in the hopes of being able to model some of the more obvious effects.

1. The *exponential window* uses a one-sided window to emphasize recent partials and to gradually attenuate the influence of older sounds.
2. The *persistence model* directly preserves the most prominent recent partials and discounts their contribution to dissonance in proportion to the elapsed time.
3. The *context model* supposes that there is a set of priviledged partials that persist over time to enter the dissonance calculations.

All three models augment the sensory dissonance calculation to include partials not currently sounding; these extra partials arise from the windowing, the persistence, or the context. A series of detailed examples in [22] showed how each of the models explained some aspects, but failed to explain others. The context model was the most successful, though the problem of how the auditory system might create the context in the first place was left unexplored.

To see how this might work, consider a simple context that consists of a set of partials at 220, 330, 440, and 660 Hz.

When a harmonic note A or E is played at a fundamental of 220 or 330 Hz, many of their partials coincide with those of the context, and the dissonance calculation (which now includes the partials in the context as well as those in the currently sounding notes) is barely larger than the intrinsic dissonance of the A or E themselves. When, however, a G♮ note is sounded (with fundamental at about 233 Hz), the partials of the note will interact with the partials of the context to produce a significant dissonance.

The context idea is implemented in Adaptun using a static "drone." The check box labelled drone enables a fixed context that is defined in the collection file drone.col. The format of the data is the same as in (7) above. For example, the drone file for the four partial context of the previous paragraph is:

1, 4500;
2, 5202;
3, 5700;
4, 6402;

(The "02" occurs because the perfect fifth between 330 Hz and 220 Hz correponds to 702 cents, not 700 cents as in the tempered scale.) When the drone switch is enabled, notes that are played on the keyboard (or notes that are played from the input MIDI file menu) are adapted with a cost function that includes both the currently sounding notes and the partials specified in the drone file. The drone itself is inaudible, but it provides a framework around which the adaptation occurs. Examples are provided in Section 5.

5. Examples

This section provides several sound examples that demonstrate the adaptive tuning algorithm and the kinds of effects possible with the various options in Adaptun. A composition demonstrates the artistic potential.

5.1. Example 1: Listening to adaptation

The first sound example is presented in listenadapt.mp3 [30]. The adaptation is slowed so that it is possible to hear the controlled descent of the dissonance curve. Three notes are initialized at the ratios 1, 1.335, and 1.587, which are the 12-tone equal tempered (12-tet) intervals of a fourth and a minor sixth (for instance, C, F, and A♭). Each note has a spectrum containing four nonharmonic partials at f, $1.414f$, $1.7f$, $2f$. Because of the dense clustering of the partials and the particular intervals chosen, the primary perception of this tonal cluster is its roughness and beating. As the adaptation proceeds, the roughness decreases steadily until all of the most prominent beats are removed. The final adapted ratios are 1, 1.414, and 1.703.

This is illustrated in Figure 4, where the vertical grid on the left shows the familiar locations of the 12-tet scale steps. The three notes are represented by the three vertical lines, and the positions of the partials are marked by the small circles. During the adaptation, the lowest note descends while the higher two ascend, eventually settling on a "chord" defined by the intervals g, $1.41g$, and $1.7g$. The arrows pointing left show the locations of four pair of partials that are (nearly) coinciding.

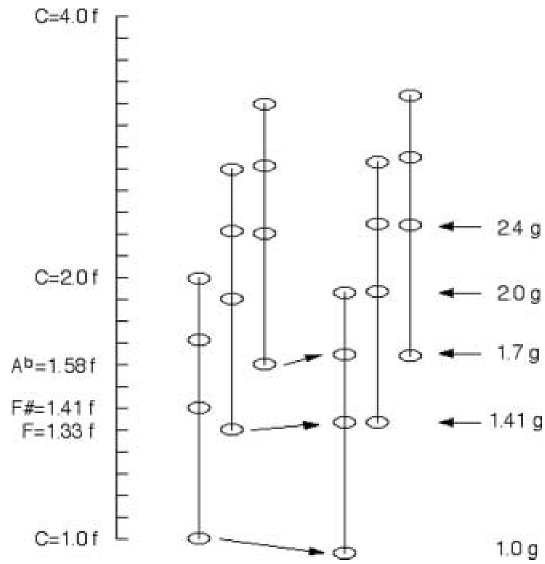

Fig. 4. Three notes have fundamentals at C, F, and A♭, and partials at $1.0f$, $1.41f$, $1.7f$, and $2.0f$. After adaptation, the C at frequency f slides down to frequency g, while the other two notes slide up to $1.41g$ and $1.70g$. The arrows on the right emphasize the resulting four pairs of (almost) coinciding partials.

In the sound example, the adaptation is performed three times, at three different speeds. The gradual removal of beats is clearly audible in the slowest. When faster, the adaptation takes on the character of a sliding "portamento." There is still some roughness remaining in the sound even when the adaptation is complete, which is due to the inherent sensory dissonance of the sound. The remaining slow beats (about one per second) are due to the resolution of the audio equipment. Thus there are two "time scales" involved in the adaptation of a musical passage. First is the rate at which time evolves in the music, the speed at which notes occur. Second is the time during which the adaptation occurs, which is determined by the stepsize parameter. The two times are essentially independent.

One aspect that may not be apparent is that the final value of g differs from run to run. This is because the iteration is not completely deterministic; the probe directions $\Delta(k)$ in (8) are random, and the algorithm will follow (slightly) different trajectories each time. The bottom of the dissonance landscape is always defined by the ratio of the fundamentals of the notes (in this case g, $1.41g$, and $1.7g$) but the exact value of g may vary.

5.2. Example 2: Adaptive study 1

The second example, presented in adaptstudy1.mp3 [31], is orchestrated for four synthesized "wind" voices. When several notes are sounded simultaneously, their pitches are often changed significantly by the adaptation. This is emphasized by the motif which begins with a lone voice. When the second voice enters, both adapt, giving rise to pitch glides and sweeps. Since the timbres have a harmonic structure, most of the resulting intervals are actually justly intoned because the notes adapt to align a partial of the lower note with some partial of the upper. By focusing attention on the pitch glides (which begin at 12-tet scale steps), this demonstrates clearly how distant many of the common 12-tet intervals are from just.

Perhaps the most disconcerting aspect of the study is the way the pitches wander. As long as the adaptation is applied only to currently sounding notes, successive notes may differ: the *C* note in one chord may be retuned from the *C* note in the next. This can produce an unpleasant "wavy" or "slimy" sound. This effect is easy to hear in the long notes which are held while several others enter and leave. For instance, between 0:36 and 0:44 seconds (and again at 1:31 to 1:39), there is a three note chord played. The three notes adapt to the most consonant nearby location. Then the top two notes change while the bottom is held; again all three adapt to their most consonant intervals. This happens repeatedly. Each time the top two notes change, the held note readapts, and its pitch slowly and noticeably wanders. Though the vertical sonority is maintained, the horizontal retunings are distracting.

The most straightforward way to forbid this kind of behavior is to leave currently sounding notes fixed as newly entering notes adapt their pitches. This can be implemented by calculating the dissonance cost function as in either (8) or (4), but setting the stepsize μ to zero for those fundamentals that are no longer new. The problem with this approach is that it does not address the fundamental problem, it only addresses the symptom that can be heard clearly in this sound example. A better way is by the introduction of the inaudible "drone," or context.

5.3. Example 3: A melody in context

Adaptun implements a primitive notion of memory or context in its drone function. A collection of fixed frequencies are prespecified in the file drone.col, and these frequencies enter into the dissonance calculation, though they are not sounded.

The simplest case is when the spectrum of the adapting sound consists of a single sine wave partial as in parts (a) and (b) of Figure 5. The unheard context is represented by the dashed horizontal lines. Initially, the frequency of the note is different from any of the frequencies of the context. If the initial note is close to one of the frequencies of the context, then dissonance is decreased by moving them closer together. The note converges to the nearest frequency of the context, as shown by the arrow. In (b), the initial note is distant from any of the frequencies of the context. When both distances are larger than the point of maximum dissonance (the peaks of the curves in Fig. 1), then dissonance is decreased by moving further away. Thus the pitch is pushed away from both nearby frequencies of the context, and converges to some intermediate position.

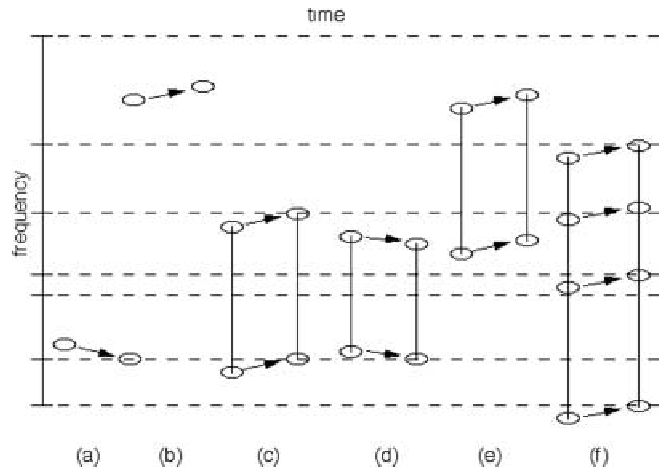

Fig. 5. The dashed horizontal grid defines a fixed "context" against which the notes adapt. When the note has a spectrum consisting of a single sine wave partial as in (a) and (b), then the note will typically adjust its pitch until it coincides with the nearest partial of the context as in (a), or else be repelled from the nearby partials of the context as in (b). When the spectrum has two partials, then the adaptation may align both partials as in (c), one as in (d), or none as in (e). In (f), the partials fight to align themselves with the context, eventually converging to minimize the beating.

Generally the timbre will be more complex than a single sine wave. Figure 5 shows several other cases. In parts (c), (d), and (e), the timbre consists of two sine wave partials. Depending on the initial pitch (and the details of the context) this may converge so that both partials overlap the context as in (c), so that one partial merges with a frequency of the context and the other does not as in (d), or to some intermediate position where neither partial coincides with the context (as in (e)). Part (f) gives the flavor of the general case when the timbre in complex with many sine wave partials and the context is dense. Typically, some partials converge to nearby frequencies in the context and some will not.

To see how this might function in a (more) realistic setting, suppose that the current context consists of the note *C* and its first 16 harmonics. When a new harmonic note occurs, it is adapted not only in relationship to other currently sounding notes, but also with respect to the partials of the *C*. Because partials of the adapting notes often converge to coincide with partials in the context (as in Fig. 5 part (f)), there is a good chance that a partial of the note will align with a partial of the context. When this occurs, the adaped interval will be just, formed from the small integer ratio formed by the harmonic of the note with the harmonic of the context.

Thus the context provides a structure that influences the adaptation of all the sounding notes, like an unheard drone. In this way, it can give a horizontal consistency to the adaptation that is lacking when no memory is allowed.

5.4. Example 4: Adaptive study 2

The fourth example, presented in adaptstudy2.mp3 [32], is orchestrated for four synthesized "violin" voices. Like the first study, the adaptive process is clearly audible in the sweeping and gliding of the pitches. For this performance, however, a context consisting of all octaves of *C* plus all octaves of *G* was used.[3]

The context encourages consistency in the pitches, maintaining (an unheard) template to which the currently sounding notes adapt. Though the study still contains significant pitch adaptation, the final resting places are constrained so that the adjusted pitches are related to the unheard *C* or *G*. Typically, some harmonic of each adapted note aligns with one of the octaves of the *C* or *G* template.

In several places throughout the piece adjacent notes (of the 12-tet scale) are played simultaneously. For the specified timbres, this is near the peak of the dissonance curve. Depending on exactly which notes are played, the order in which they are played, and the vagaries of the random test directions (the $\Delta(k)$ in (8)), sometimes the two pitches adapt to an interval at about 316 cents (a just minor third) by moving apart in pitch, and sometimes they merge into a unison at some intermediate pitch. In either case, the primary sensation is of the motion.

5.5. Example 5: *Local Anomaly*

The piece *Local Anomaly* [20] was created from a standard MIDI drum track using Adaptun. In a SMF drum track, each type of drum (snare, tom, high hat, etc) is assigned to a different note. The first step was to randomize the notes (within reasonable limits). These were then played using various percussive stringed instrument sounds (guitars, basses, pianos, etc.). This extremely dissonant but highly rhythmic soundscape was input into Adaptun, and the notes adapted towards consonance. A simple context consisting of all octaves of the note *C* was used. The output was recorded, and the resulting MIDI file was then more carefully orchestrated.

As expected from the previous examples, one of the most prominent features of the piece is the pitch glides. These give an "elasticity" to the tuning, analogous to a guitar bending strings into (or out of) tune. All pitch glides in *Local Anomaly* are created by the adaptive process. The piece has no clear notion of musical "key", yet does maintain consonance by converging pitches to intervals defined primarily by small integer ratios. Thus adaptation provides a kind of "intelligent" portamento that begins wherever commanded by the performer (or MIDI file), and slides smoothly to a nearby "most consonant" set of intervals. The speed of the slide is directly controllable and may be (virtually) instantaneous or as slow as desired.

[3] The drone file contained all the *C*'s 2400, 3600, 4800, 6000, . . . plus all the *G*'s 3100, 4300, 5500, 6700, ...

6. Discussion

The adaptive tuning strategy can be viewed as a generalization of Just Intonation in two respects. First, it is independent of the key of the music being played, that is, it automatically adjusts the intonation as the notes of the piece move through various keys. This is done without any specifically "musical" knowledge such as the local "key" of the music, though such knowledge can be incorporated in a simple way via the "context," the unheard drone. Second, though this application has not been stressed here, the adaptive tuning strategy is applicable to inharmonic as well as harmonic sounds. This broadens the notion of "Just Intonation" to include a larger palette of sounds.

By functioning at the level of successions of partials (and not at the level of notes) the sensory dissonance model does not deal directly with pitch, and hence does not address melody, or melodic consonance. Rasch [17] describes an experiment in which "Short musical fragments consisting of a melody part and a synchronous bass part were mistuned in various ways and in various degrees. Mistuning was applied to the harmonic intervals between simultaneous tones in melody and bass . . . The fragments were presented to musically trained subjects for judgments of the perceived quality of intonation. Results showed that the melodic mistunings of the melody parts had the largest disturbing effects on the perceived quality of intonation . . ." Interpreting "quality of intonation" as roughly equivalent to melodic dissonance, this suggests that the misalignment of the tones with the internal template was more important than the misalignment due to the dissonance between simultaneous tones.

Such observations suggest why attempts to retune pieces of the common practice period into Just Intonation, adaptive tunings, or other theoretically ideal tunings may fail;[4] squeezing harmonies into Just Intonation requires that melodies be warped out of tune. If the melodic dissonance described by Rasch dominates the harmonic dissonance, then the process of changing tunings may introduce more dissonance, albeit of a different kind. This does not imply that it is impossible (or difficult or undesirable) to compose in these alternative tunings, nor does it suggest that they are somehow inferior; rather, it suggests that pieces should be performed in the tunings in which they were conceived.

7. Conclusions

Just as the theory of four taste bud receptors cannot explain the typical diet of an era or the intricacies of French cuisine, so the theories of sensory dissonance cannot explain the

[4] The effort to improve Beethoven or Bach by retuning pieces to Just Intonation produced a sense that the music was "unpleasantly slimy" (to quote George Bernard Shaw when listening to Bach on Bosanquet's 53-tone per octave organ [10]) or badly out of tune due to the melodic distortions.

history of musical style or the intricacies of a masterpiece. Even restricting attention to the realm of sensory dissonance, the average amount of dissonance considered appropriate for a piece of music varies widely with style, historical era, instrumentation, and experience of the listener.

The intent of Adaptun is to give the adventurous composer a new option in terms of musical scale: one that is not constrained *a priori* to a small set of pitches, yet that retains some control over consonance and dissonance. The incorporation of the "context" feature helps to maintain a sense of melodic consistency, while still allowing the pitches to adapt to (nearly) optimal intervals.

References

[1] Adaptun is currently available for download at our website http://eceserv0.ece.wisc.edu/~sethares/adaptun/adaptun.zip

[2] Carlos, W. (1987). "Tuning: At the Crossroads," Computer Music Journal, Vol. 11, No. 1, 29–43, Spring.

[3] DeLaurentis, J. "Adaptive Tuning Web Site", http://www.adaptune.com/

[4] Denckla, B. (1997). *Dynamic Intonation for Synthesizer Performance*, Masters Thesis, MIT, Sept.

[5] Hall, D.E. (1974). "Quantitative Evaluation of Musical Scale Tunings," *Am. J. Physics*, 543–552, July.

[6] Helmholtz, H. (1954). *On the Sensations of Tones*, (1877). Trans. A.J. Ellis, Dover, New York.

[7] Kiefer, J. & Wolfowitz, J. (1952). "Stochastic Estimation of a Regression Function," *Ann. Math. Stat. 23*, 462–466.

[8] Kunst, J. (1949). *Music in Java*, Martinus Nijhoff, The Hague, Netherlands.

[9] Leman, M. (2000). "Visualization and calculation of the roughness of acoustical musical signals using the synchronization index model," Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00), Verona, Italy, December 7–9.

[10] McClure (1948). Studies in Keyboard Temperaments, GSJ, i, pp. 2840.

[11] Parncutt, R. (1989). *Harmony: A Psychoacoustic Approach*, Springer-Verlag.

[12] Parncutt, R. (1994). "Applying psychoacoustics in composition: 'harmonic' progressions of 'nonharmonic' sonorities," Perspectives of New Music.

[13] Partch, H. (1974). *Genesis of a Music*, Da Capo Press, NY.

[14] Plomp, R. & Levelt, W.J.M. (1965). "Tonal consonance and critical bandwidth," *J. Acoust. Soc. Am. 38*, 548–560.

[15] Polansky, L. (1987). "Paratactical Tuning: An Agenda for the Use of Computers in Experimental Intonation," Computer Music Journal, Vol. 11, No. 1, 61–68, Spring.

[16] Rameau, J.P. (1722). *Treatise on Harmony*, Dover Pub., NY (1971), original edition.

[17] Rasch, R.A. (1985). "Perception of melodic and harmonic intonation of two-part musical fragments," Music Perception, Vol. 2, No. 4, Summer.

[18] Sethares, W.A. (1993). "Local consonance and the relationship between timbre and scale," *J. Acoust. Soc. Am. 94*(3), pg. 1218–1228, Sept.

[19] Sethares, W.A. (1994). "Adaptive tunings for musical scales," *J. Acoust. Soc. Am.* 96, no. 1, pg. 10–19, July.

[20] Sethares, W.A. (2002). Local Anomaly, Track 3 on *Exomusicology*, Odyssey Records, EXO-2002, Nashville, TN. (available in mp3 format at http://eceserv0.ece.wisc.edu/~sethares/adaptun/localanomaly.mp3)

[21] Sethares, W.A. (1997). *Tuning, Timbre, Spectrum, Scale*, Springer Verlag.

[22] Sethares, W.A. & McLaren, B. (1998). "Memory and context in the calculation of sensory dissonance," Proc. of the Research Society for the Foundations of Music, Ghent, Belgium, Oct.

[23] Spall, J.C. (1992). "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," IEEE Trans. *Autom. Control 37*, 332–341.

[24] Spall, J.C. (1997). "A One-Measurement Form of Simultaneous Perturbation Stochastic Approximation," *Automatica 33*, 109–112.

[25] Terhardt, E. (1974). "Pitch, consonance, and harmony," *J. Acoust. Soc. Am.*, Vol. 55, No. 5, 1061–1069 May.

[26] Terhardt, E. (1984). "The concept of musical consonance: a link between music and psychoacoustics," *Music Perception*, Vol. 1, No.2 276–295, Spring.

[27] Wagge, H.M. (1985). "The Intelligent Keyboard," 1/1, 1(4): *1*, 12–13.

[28] Wilkinson, S.R. (1988). *Tuning In*, Hal Leonard Books, Milwaukee, WI.

[29] Zicarelli, D., Taylor, G. et al. (2001). *Max 4.0 Reference Manual*, Cycling '74. http://www.cycling74.com/products/dldoc.html

[30] Available in mp3 format at http://eceserv0.ece.wisc.edu/~sethares/adaptun/listenadapt.mp3

[31] Available in mp3 format at http://eceserv0.ece.wisc.edu/~sethares/adaptun/adaptstudy1.mp3

[32] Available in mp3 format at http://eceserv0.ece.wisc.edu/~sethares/adaptun/adaptstudy2.mp3