Piecewise-Linear Approximation of Additive Synthesis Envelopes: A Comparison of Various Methods

Author(s): Andrew Horner and James Beauchamp

Source: *Computer Music Journal*, Summer, 1996, Vol. 20, No. 2 (Summer, 1996), pp. 72-95

Published by: The MIT Press

Stable URL: http://www.jstor.com/stable/3681333

Andrew Horner
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
horner@cs.ust.hk

James Beauchamp
School of Music and Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61820, USA
j-beauch@uiuc.edu

Piecewise-Linear Approximation of Additive Synthesis Envelopes: A Comparison of Various Methods

Piecewise-linear approximation of additive-synthesis amplitude and frequency envelopes is one of the most common data-reduction techniques used in sound synthesis. This article examines and compares a number of methods for determining a prescribed number of breakpoint times in order to achieve the best line-segment approximations to the amplitude envelopes of a sound's harmonics. The method uses a set of breakpoint times that are common to all harmonics, and it samples amplitude values from the original envelopes at the selected times. Resynthesis uses additive synthesis or, more efficiently, linear interpolation between waveforms that represent the spectrum at each breakpoint.

Our results, based on a simple error measure, show that a genetic algorithm (GA) and a sequential enumeration (greedy) method consistently outperform other methods, such as those utilizing equally spaced breakpoint times, random breakpoints, and hill climbing. Also, the GA method with common breakpoint times does as well as or better than several extensions to this method, including quadratic approximation, use of scalars to optimize breakpoint amplitudes, and use of independent breakpoint times for each harmonic. With some small modifications, the GA method also works well for approximating frequency envelopes. While the GA method generally yields the best solution, the greedy method has the advantage of taking less time to find a good solution. This article presents results for a trumpet tone, a piano tone, and a guitar tone. Formal listening tests indicate that for the average listener to misidentify synthetic tones as original acoustic tones half of the time, twelve breakpoints are required for the trumpet tone, and nine for the piano and guitar tones.

Computer Music Journal, 20:2, pp. 72–95, Summer 1996

Background

Computer music applications commonly use *piecewise-linear* (straight-line segment) *approximations* (PLAs) to model amplitude envelopes. Line segments require a minimal amount of data to specify, and are easy to modify and generate. Using PLAs dates back to the early days of computer music, and over the years many digital synthesizers have employed them. Additive synthesis is one of the main synthesis applications that has often utilized piecewise-linear approximations. Examples of commercial digital synthesizers using additive synthesis with piecewise-linear envelopes include the Fairlight CMI (1979), the Kurzweil 150 Fourier Synthesizer (1986), and the Lyre Fourier Digital Synthesizer (1986).

The general problem is to find the best set of amplitude $\{t_{A_{k,j}}, A_{k,j}\}$ and frequency $\{t_{f_{k,j}}, f_{k,j}\}$ coordinates, where $A_{k,j}$ and $f_{k,j}$ are the amplitude and frequency of the kth harmonic at the jth time points, $t_{A_{k,j}}$ and $t_{f_{k,j}}$, respectively. (This article generally uses the term "harmonic" to refer to a partial, whether or not its frequency is an integer multiple of the fundamental frequency.) We then use these coordinates to construct piecewise-linear amplitude and

frequency envelopes $a'_k(t)$ and $f'_k(t)$ that are "good" approximations to the original time-varying amplitudes $a_k(t)$ and frequencies $f_k(t)$ in the Fourier sum:

$$s(t) = \sum_{k=1}^{N_{har}} a_k(t) \cos(2\pi \int f_k(t)dt + \theta_k), \quad (1)$$

where $s(t)$ is the original signal, k the harmonic number, q_k the kth harmonic's starting phase, and N_{har} the number of harmonics.

A closed-form solution to the line-segment approximation problem does not exist, because the problem is very nonlinear; moving just one breakpoint changes how well the approximation matches the original envelope over the length of the neighboring segments.

Several researchers have reported fitting amplitude and frequency envelopes by hand (Risset and Mathews 1969; Grey and Moorer 1977; Chamberlin 1980), and at least three have proposed automated methods for reducing the approximation error below some predetermined threshold (Beauchamp 1969; Strawn 1980; Serra, Rubine, and Dannenberg 1990).

James A. Moorer and his colleagues gave hand-fitted PLAs for amplitude and frequency envelopes to various instrument tones, including violin, clarinet, oboe, and trumpet (Moorer, Grey, and Snell 1977; Moorer, Grey, and Strawn 1977, 1978). Engineers have since used this data extensively to test various real-time digital synthesizers.

James Beauchamp's 1969 LINSEG method for automatically determining PLAs optimized each amplitude envelope separately. After smoothing each envelope to remove microvariations, the procedure used a series of least-squares-fit straight lines to approximate the data. The LINSEG method used the longest lines that kept the absolute difference error, $|a_k(t) - a'_k(t)|$, below a specified threshold. For complex envelopes, this method tended to generate a multitude of straight lines (usually many more than a comparable "hand-fit" would have produced).

However, our objective in this article is to determine the best possible approximation for a specified number of line segments. For example, if someone wanted to design an instrument patch using a keyboard synthesizer constrained to only five line segments per envelope, an automatic procedure would be needed that specified how to best utilize the line segments. John Strawn's 1980 procedure (ADJUST) attempted this, but it required an initial estimate of the solution, and the solution it ultimately found was usually only slightly improved over the initial guess. This is because ADJUST used a hill-climbing procedure to improve on an initial guess, and was thus restricted to converging to some nearby local optimum.

While most PLA attempts in the past, whether by hand or automatic method, used independent breakpoints for each envelope, we have decided to focus on the problem of picking the N best interior breakpoints that are common to all harmonics of a sound. For amplitude envelopes, we assume zero-valued boundary breakpoints at the beginnings and ends of tones, and thus do not count them among the breakpoints we must optimize.

Using common breakpoint times has several advantages. First, common breakpoints immediately save close to a factor of two in data storage over independent breakpoint times, because we only have to retain one set of breakpoint times instead of N, along with the usual N sets of breakpoint amplitudes. Second, the methods investigated in this article run considerably faster when restricted to finding common breakpoint times. Third, as we will show later, common times give equally good or better results compared to independent times when we take into account equivalent memory requirements. Finally, and perhaps most importantly, one can easily show (see the Appendix) that additive synthesis is mathematically equivalent to wavetable interpolation synthesis if the partials' time-varying frequencies $f_k(t)$ are harmonically related (i.e., $f_k(t) = kf_1(t)$) and the phases are consistent. In most situations, straight-line wavetable interpolation synthesis is much faster than additive synthesis, since only two wavetables are used at a time.

To our knowledge, the first synthesizer that synthesized sounds based on linear interpolation between wavetables was the Prism, built by Kinetic

Figure 1. Analysis/synthesis system with breakpoint data reduction.

Figure 2. Amplitude envelopes for harmonics 1–4 (left) and 5–10 (right) of the original trumpet tone played at a fundamental frequency of 350 Hz.

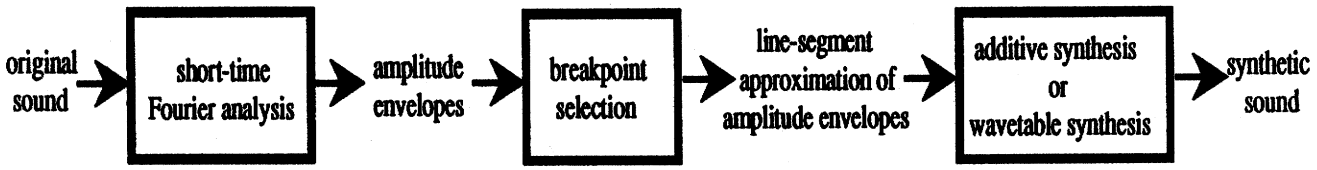


Figure 1

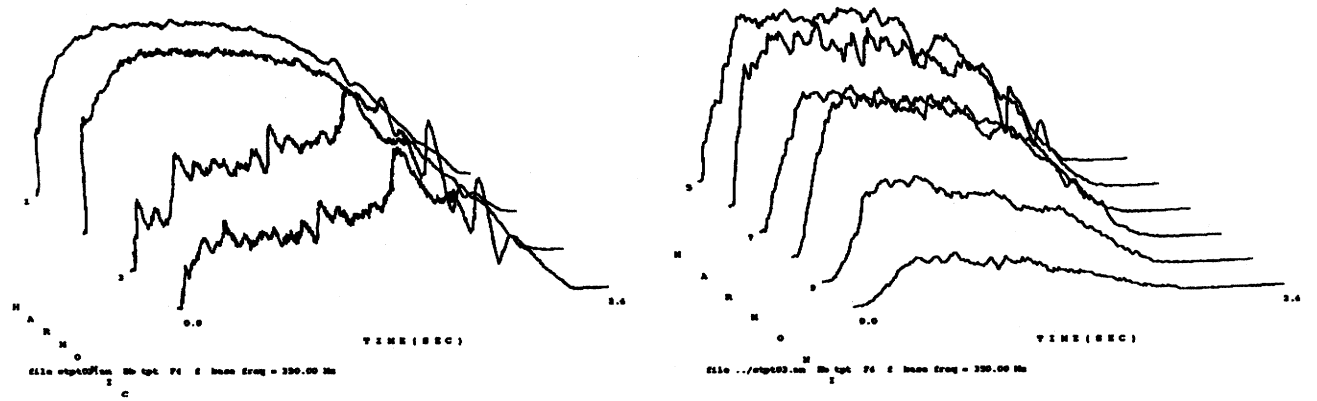


Figure 2

Systems in the early 1980s. Also, Hal Chamberlin employed this method for microcomputer real-time synthesis during the same time period (Chamberlin 1980). The method was explored in detail by Serra, Rubine, and Dannenberg (1990), who devised a technique for adding breakpoints until the maximum mean-squared error was brought below a prescribed value. Note that our goal is quite different, since we use a different error criterion and wish to find the best breakpoints that satisfy that criterion.

The main analysis/synthesis steps needed for breakpoint data reduction of harmonics' amplitude envelopes are outlined in Figure 1. First, we apply a pitch-synchronous short-time Fourier transform (STFT) to the musical signal we wish to approximate. The fast Fourier transform (FFT) uses a Hamming window with a length equal to twice the average frequency period of the signal (Beauchamp 1993), resulting in two analysis frames for each period. The result is a sampled amplitude envelope $ak(n)$ for each harmonic k and each analysis frame number $n \cong 2f_a t$, where f_a is a fixed frequency close to the fundamental frequency of the original sound. Thus, the total number of frames for each envelope is $N_{\text{frames}} = 2f_a t_1$, where t_1 is the duration of the original tone; for example, a 2-sec middle C (261.6 Hz) would require 1,046 analysis frames.

The goal is to find the best N breakpoint times to approximate the amplitude envelopes produced by the STFT analysis, such as the trumpet envelopes shown in Figure 2. In general, the methods reported in this article use the original amplitude values occurring at each of the selected breakpoint times. This guarantees that we will have exact matches at these times, as well as good matches at surrounding points. Changing the time and amplitude values may refine the solution a little, but likely will not offer significant improvement over the use of carefully optimized breakpoints.

We can reconstruct the sound by performing additive synthesis on the line-segment approximations, using linear interpolation to generate amplitude values between the breakpoints. Alternatively, we can use straight-line interpolation between the waveforms defined at each breakpoint time, assuming that the frequencies are harmonically related and

that the harmonics' phases are consistent. In the Appendix we demonstrate that additive synthesis and wavetable interpolation synthesis are mathematically equivalent under these conditions.

To evaluate how well an approximation matches the original signal, we define a relative error measure as follows:

$$\text{Relative Error} = \frac{1}{N_{\text{frames}}} \sum_{n=1}^{N_{\text{frames}}} \sqrt{\frac{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{\text{har}}} a_k(n)^2}} \quad (2)$$

where $a'k(n)$ is the piecewise-linear approximation to the kth harmonic amplitude at the nth frame, $ak(n)$ is the corresponding amplitude of the original signal, N_{har} is the number of harmonics used, and N_{frames} is the number of analysis frames spanning the particular musical sound.

This relative error measure gives an equal weight to each time frame during the tone. An absolute error measure would give excessive weight to the louder portions of the tone, which is undesirable since low-amplitude attack and decay frames are at least as important as their steady-state counterparts. Therefore, the relative error is more likely to match our perceptual sense of error.

However, even this definition does not give enough weight to the critically important attack, which may only last a short fraction of an entire tone's duration. Therefore, we have modified Equation 2 so that the attack receives a weight equal to the remainder of the tone:

$$\text{Relative Error} = \frac{w_1}{\text{PeakFrame-1}} \sum_{n=1}^{\text{PeakFrame}-1} \sqrt{\frac{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{\text{har}}} a_k(n)^2}} \quad (3)$$

$$+ \frac{w_2}{N_{\text{frames}} - \text{PeakFrame} + 1} \sum_{n=\text{PeakFrame}}^{N_{\text{frames}}} \sqrt{\frac{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{\text{har}}} a_k(n)^2}}$$

We have used $w_1 = w_2 = 0.5$, which insures that the attack receives a substantial weighting, but, of course, other weights and subdivisions are possible. Also, we use the peak RMS amplitude (within the first 100 msec) to determine when the attack ends, and set PeakFrame to the frame number where this occurs. Equation 3 may actually de-emphasize the attack in very short sounds where the attack is longer than the rest of the tone, but in our experience it gives reasonable results for most sounds. Nevertheless, we would like to emphasize that the error measure is the key to the success of any approximation algorithm; how to determine the best formulation for measuring the perceptual error between two similar time-varying spectra is still an open question.

Existing Breakpoint Selection Methods

This section evaluates some familiar methods for determining a fixed number of shared breakpoint times for line-segment approximation of harmonics' amplitude envelopes, and discusses the methods' advantages and disadvantages. These methods range from simple intuitive procedures to more elaborate iterative techniques.

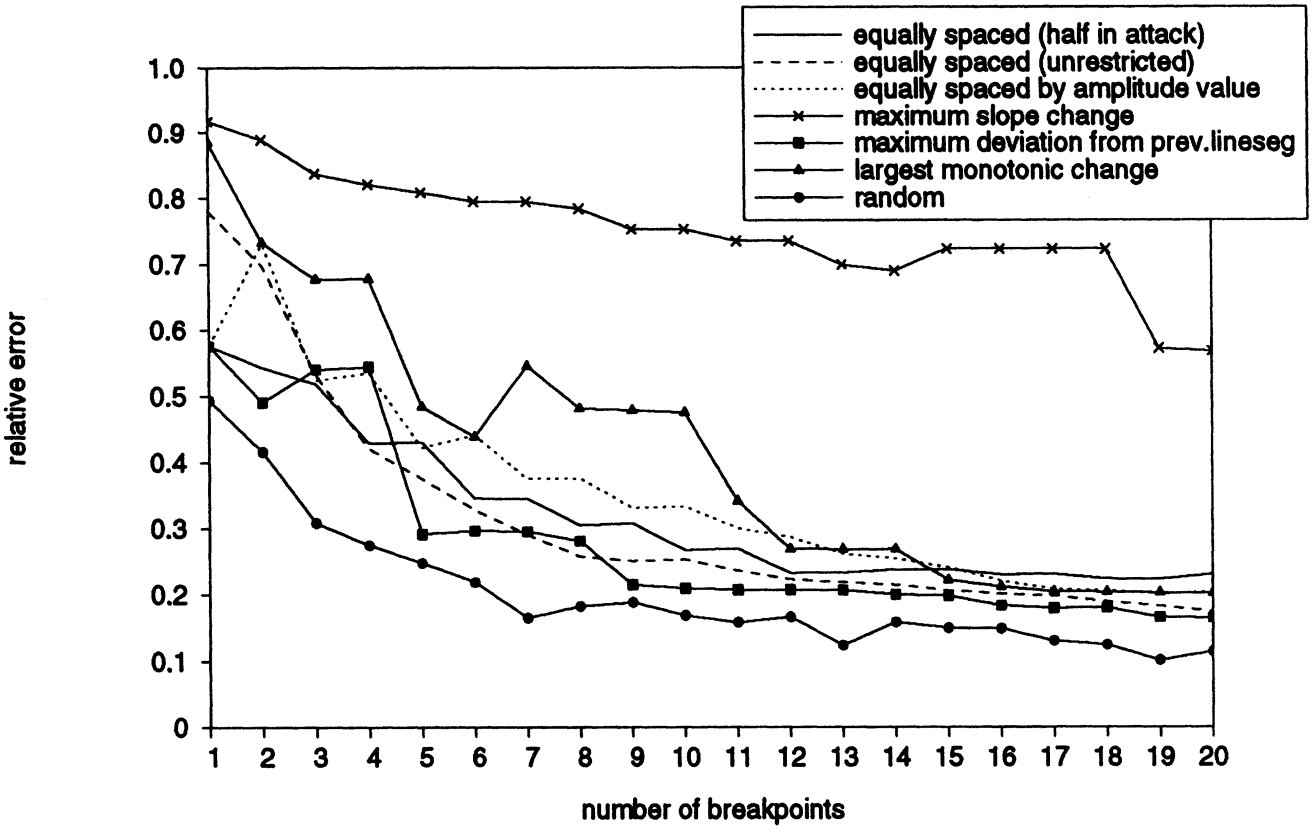
Simple Strategies

There are a number of rather simple and intuitive ways to select the breakpoints. One method is to divide the tone into equal time segments and pick the breakpoints at the segment boundaries. Alternatively, since our fitness function emphasizes the attack, we can select half of the times to be equally spaced in the attack and the other half distributed over the remainder of the tone.

As another approach, we could use amplitude value, rather than time, to pick the points to be equally spaced. For instance, with seven breakpoints and a peak RMS amplitude of 100, we could select the breakpoints to correspond to RMS amplitudes 25, 50, 75, 100, 75, 50, and 25. Other possible criteria for selecting breakpoints include: (1) picking points where maximum slope changes occur in the amplitude envelopes, (2) selecting points at positions of worst error for previous approximations, and (3) picking points that bracket the largest monotonic changes in the amplitude envelopes.

These methods and their variants are quite in-

Figure 3. Minimum relative errors found by simple and random breakpoint selection methods for different numbers of common breakpoints on a trumpet tone.



flexible. They perform well for some types of envelope shapes, such as slowly varying smooth curves, but fail for many others, especially those with rapid fluctuations. It is all too easy to pick a bad breakpoint inadvertently. We need methods that have more adaptability than these simple approaches if we want to succeed for different kinds of acoustic instrument sounds.

Still another alternative is to randomly generate several sets of breakpoint times and choose the best set based on the error function of Equation 3. While the random approach is simple and efficient, it has the disadvantage that even after a large number of trials it may not produce a very good solution. Also, since the random method generates solutions independently of one another, it never uses the good solutions to build something better; it starts from scratch each time. We need a method that explores the possibilities more systematically.

Figure 3 compares the relative errors returned by these various breakpoint selection strategies, plotted as functions of the number of breakpoints, for PLA approximation of the trumpet tone. An error of zero corresponds to a perfect match, while an error of 0.1 corresponds to a 10-percent spectral error. Using the best of 100 random trials, the random approach consistently outperformed the other methods, demonstrating their limited usefulness. A good method should certainly do better than a strictly random method.

Breakpoint Selection Based on Hill Climbing

Hill climbing (gradient search) is a procedure that takes an initial set of breakpoints and then systematically tries to move each breakpoint individually by a single small increment in order to gradually reach the summit of a "hill" (in our case, when Equation 3 is minimized). The move that gives the

Figure 4. Fitness of random breakpoint-time sets (times given in frames) before and after hill climbing.

	fitness	breakpoints
random before HC	0.381128	{244, 266, 279, 560, 760, 826, 960, 1217, 1379}
after Hill Climbing	0.367248	{240, 267, 283, 559, 760, 825, 960, 1206, 1378}
random before HC	0.225909	{10, 158, 351, 988, 1193, 1196, 1312, 1481, 1556}
after Hill Climbing	0.220895	{11, 160, 351, 986, 1195, 1199, 1321, 1482, 1557}
random before HC	0.290187	{369, 694, 746, 870, 1270, 1290, 1559, 1600, 1678}
after Hill Climbing	0.254799	{353, 695, 747, 869, 1259, 1293, 1515, 1602, 1670}
random before HC	0.386128	{146, 260, 761, 868, 991, 1079, 1185, 1262, 1299}
after Hill Climbing	0.377573	{138, 259, 760, 868, 989, 1079, 1187, 1238, 1296}
random before HC	0.563765	{29, 242, 576, 585, 799, 892, 897, 983, 1237}
after Hill Climbing	0.536761	{36, 239, 574, 580, 796, 892, 896, 982, 1240}
random before HC	0.430742	{360, 384, 531, 617, 931, 1050, 1358, 1374, 1378}
after Hill Climbing	0.424780	{353, 387, 531, 615, 935, 1047, 1359, 1373, 1378}
random before HC	0.187997	{85, 98, 220, 380, 641, 1186, 1416, 1632, 1665}
after Hill Climbing	0.173573	{81, 99, 222, 380, 642, 1187, 1414, 1608, 1653}

most improvement is kept, and the procedure is repeated until no single change yields an improvement. John Strawn's ADJUST algorithm (1980) is an example of a hill-climbing algorithm. Hill climbing is more effective as a refinement algorithm than as an optimization method for complex problems. The method will move a set of breakpoints to the nearest local optimum (hill summit), but no further. The fitness of its solution is therefore highly dependent on the fitness of the set of initial breakpoints given to the procedure.

For instance, Figure 4 shows sets of randomly generated breakpoint times (given in frames) for the trumpet tone, compared with revised sets after hill climbing. In each case only marginal improvement results; if the initial guess is poor, the result remains poor even after hill climbing. Simply selecting a few other random sets of breakpoints is likely to yield more improvement than hill climbing from a poor solution. On the other hand, hill climbing can usually improve any solution by a modest amount, so it is useful for refining solutions.

Breakpoint Selection Based on Greedy and Genetic Algorithms

This section describes two new methods for shared-breakpoint selection. (By shared breakpoints, we mean breakpoint times that are the same for all the amplitude envelopes of a sound's harmonics.) One method uses a greedy algorithm to build up a set of breakpoint times. The other method employs a genetic algorithm to search for the best set of N breakpoint times. For breakpoint amplitudes, we simply use the actual per-harmonic amplitude values that occur at the breakpoint times (instead of attempting to optimize the amplitudes, as discussed later).

Greedy Breakpoint Selection

Greedy breakpoint-time selection uses an iterative procedure to determine the best breakpoint times. First, the greedy procedure tries each of the original tone's analysis frames as a possible breakpoint, and selects the one that yields the best fit according to Equation 3 (giving two line segments per envelope). Next, keeping the first breakpoint time, the procedure tries all the other possible times, and again retains the best one. The procedure continues by following this pattern to find more times, adding one at each step.

While the first breakpoint set, consisting of a single time value, is optimal, in successive steps the procedure does not consider changing previously found breakpoint times, but rather only augments the previously found set. The greedy method therefore only returns locally optimal solutions, instead of the best ones possible. It definitely does not find optimal solutions for two or more breakpoint times. This method also has the disadvantage that it needs to compute $N - 1$ breakpoint approximations before it can solve for the Nth approximation, but since each iteration takes little time to compute, this is not a big problem.

As demonstrated below, the approximations yielded by the greedy breakpoint selection algorithm are not generally as good as those produced by the GA method for the same number of breakpoints. However, the greedy method does achieve results faster; moreover, it can achieve the same amount of error if the penalty of requiring more breakpoint times is acceptable, though the greedy algorithm takes about the same amount of time as the GA to add the extra breakpoints.

The number of calculations in the Equation 3 fitness function is proportional to the number of harmonics N_{har} and number of analysis frames N_{frames}. For N breakpoints, brute-force solution to the breakpoint problem is not practical, since it requires about $(N_{har} \times N_{frames})_N$ fitness evaluations. The greedy algorithm requires only $N \times N_{frames}$ fitness evaluations. Even though this is much better than brute force, we still must avoid excessively long run times by only considering representative analysis frames (about 100–200) in the fitness function, rather than all of them. Skipping some analysis frames reduces the turnaround time with only a modest loss of accuracy (Horner, Beauchamp, and Haken 1993a, 1993b). For a few breakpoints, with a typical instrument tone having about 30 harmonics, our greedy algorithm completes in a few seconds on a Silicon Graphics Indigo workstation (this machine benchmarks at approximately 100,000 Dhrystones/sec); for 20 or more breakpoints, the greedy algorithm requires about a minute.

Breakpoint Selection Based on a Genetic Algorithm

The PLA method proposed in this section uses a genetic algorithm (Goldberg 1989; Holland 1975; Horner, Beauchamp, and Haken 1993b) to select the breakpoint times. Genetic algorithms use "natural selection" and evolution-inspired operators like crossover and mutation to optimize combinatorial problems such as breakpoint selection. They work with a population of candidate solutions to efficiently examine the search space and explore its various local optima in parallel. This robust parallel exploration is well suited to the PLA problem, where it is better to avoid converging to local minima (as hill climbing does). We can use the GA to systematically explore the search space and find the best combination of breakpoint times. After the GA converges to a particular solution, we do an iteration of hill climbing to ensure that the answer is the best in that region of the search space. Our breakpoint optimization procedure uses a standard simple genetic algorithm with binary tournament selection, one-point crossover, and a mutation rate equal to the reciprocal of the number of bits in the encoding. Goldberg (1989) gives more details on the workings of genetic algorithms.

To use a GA, we must specify an objective fitness function and a binary encoding of the breakpoints. Equation 3 gives the error that we wish to minimize and defines the fitness function. Coding the breakpoint times is very straightforward: we simply use the frame number, expressed as a binary number, where the first frame is numbered zero.

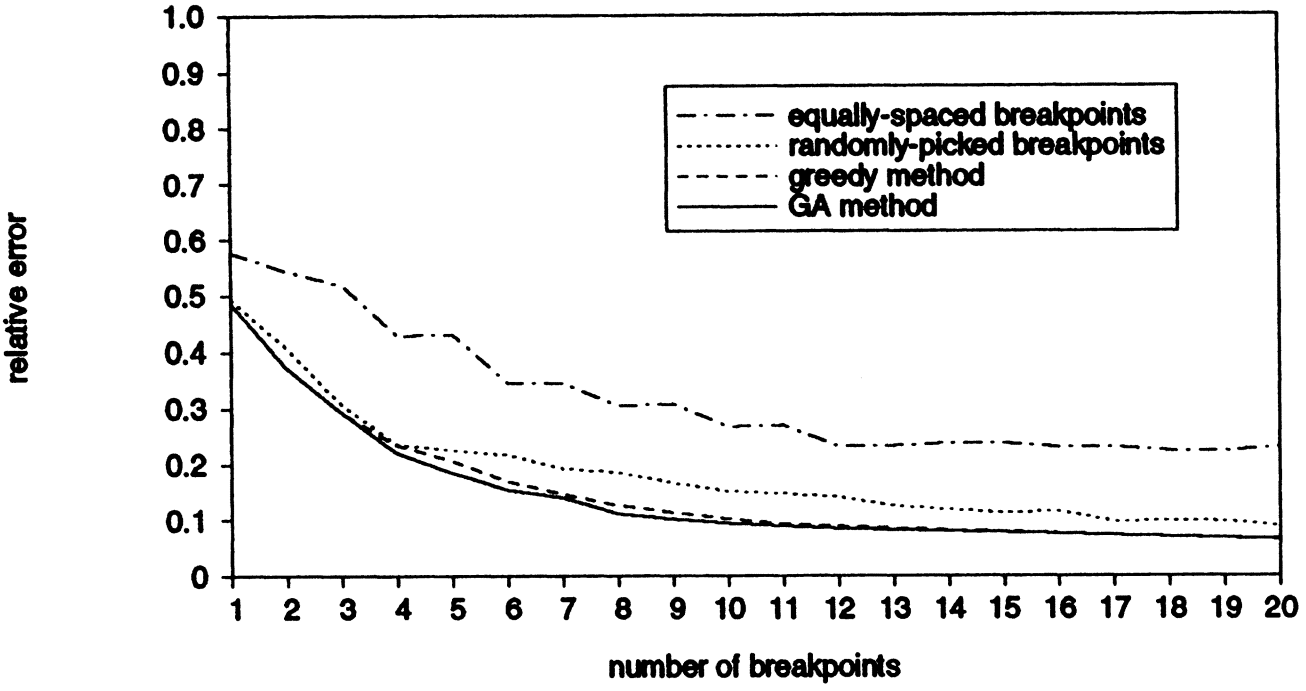
Considering its robustness in achieving near-optimal results, the genetic algorithm is an efficient iterative procedure. As with the greedy algorithm, the fitness function is used to test only a relatively small proportion of the sound's analysis frames. The speed of the GA depends on the number of breakpoints N and the number of bits required to represent the breakpoint times ($\log_2 N_{frames}$). The GA's running time also includes a relatively large constant factor, which is one of the reasons the GA runs slower than the greedy algorithm. As mentioned above, for a few breakpoints, with a typical instrument tone having about 30 harmonics, the GA completes in about a minute on a Silicon Graphics Indigo workstation.

Shared Breakpoint Selection: A Comparison of Four Methods on Three Examples

This section compares the results of applying the equally spaced, random, greedy, and GA procedures to determine common breakpoint times for three different acoustic-instrument tones. We use the modified equally spaced selection procedure that selects half the breakpoints in the attack and the rest over the remainder of the tone. For random selection, we allowed the procedure to perform about the same number of trials as the GA, and selected the best breakpoint set.

For each example, we show the original amplitude envelopes and their GA-determined line-segment approximations. We also plot the relative error of each approximation verses the number of breakpoints for the equally spaced, random, greedy, and GA methods.

Figure 5. Relative errors returned by the breakpoint selection methods for a trumpet tone. The same breakpoint times are used for each harmonic.



Trumpet

The first example is a fortissimo trumpet tone at 350 Hz (F4) with a duration of 2.4 sec. (Refer to the analysis shown in Figure 2.) Figure 5 compares the minimum relative errors found by the equally spaced, random, greedy, and GA breakpoint selection procedures, plotted as functions of the number of breakpoints used. We see that the greedy and GA error curves are almost identical and decrease monotonically as the number of breakpoints increases, while the other two methods yield inferior results and fluctuate somewhat. The equally spaced selection procedure clearly performs worse than the other methods and has a slower rate of improvement. The random method follows the GA and greedy curves up to about four breakpoints, but its error does not decrease as fast after that. It appears that the probability of randomly selecting a refined set of breakpoints diminishes beyond that point. The greedy and GA methods are consistently better than the other two methods. The point of convergence, beyond which improvements seem to diminish, occurs after about nine breakpoints with these two methods.

In Figure 6 we juxtapose the first four harmonics of the GA-optimized line-segment approximations to the trumpet example for various numbers of breakpoints. Perceptually, even the one-breakpoint GA and greedy approximations yield synthetic tones with a trumpet-like quality, though they lack the characteristic "liveliness" of the original tone. With more breakpoints, the synthetic tones become more and more life-like in character. In formal listening tests, we found that twelve or more breakpoints were required for the average listener to confuse the synthetic tones with the originals half of the time. By way of comparison, in informal listening tests the random method required 15 to 20 breakpoints to reach comparable levels of distinguishability.

Piano

The piano sound is characterized by a fast attack and a very gradual exponential decay. Figure 7 shows the time-varying spectrum of a fortissimo 261-Hz (middle C) piano tone, held for 3 sec. Most of the envelopes have smooth shapes, but the at-

Figure 6. Harmonics 1–4 of genetic algorithm (GA) piecewise-linear approximations to the trumpet.

Figure 7. Amplitude envelopes for harmonics 1–10 of the original piano tone played at 261 Hz.

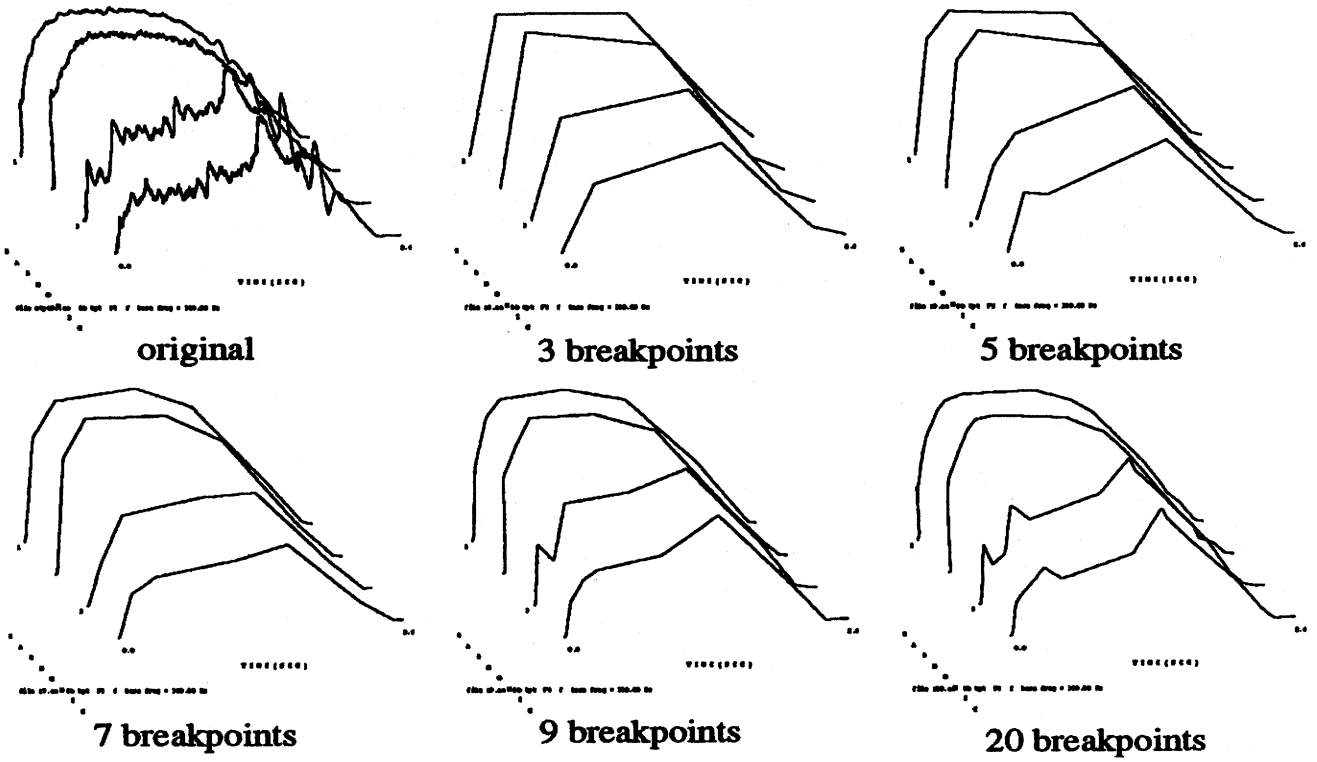


Figure 6

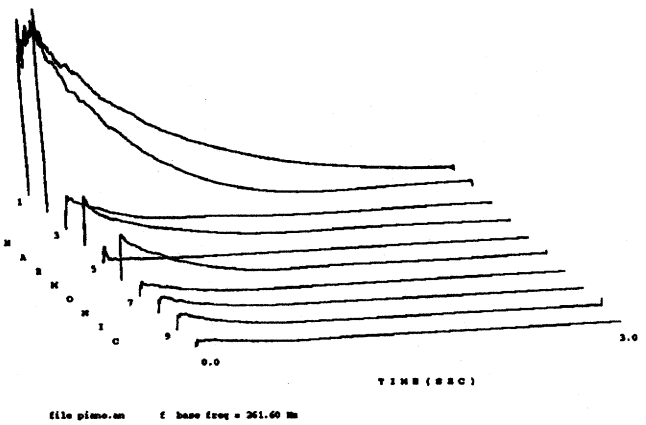


Figure 7

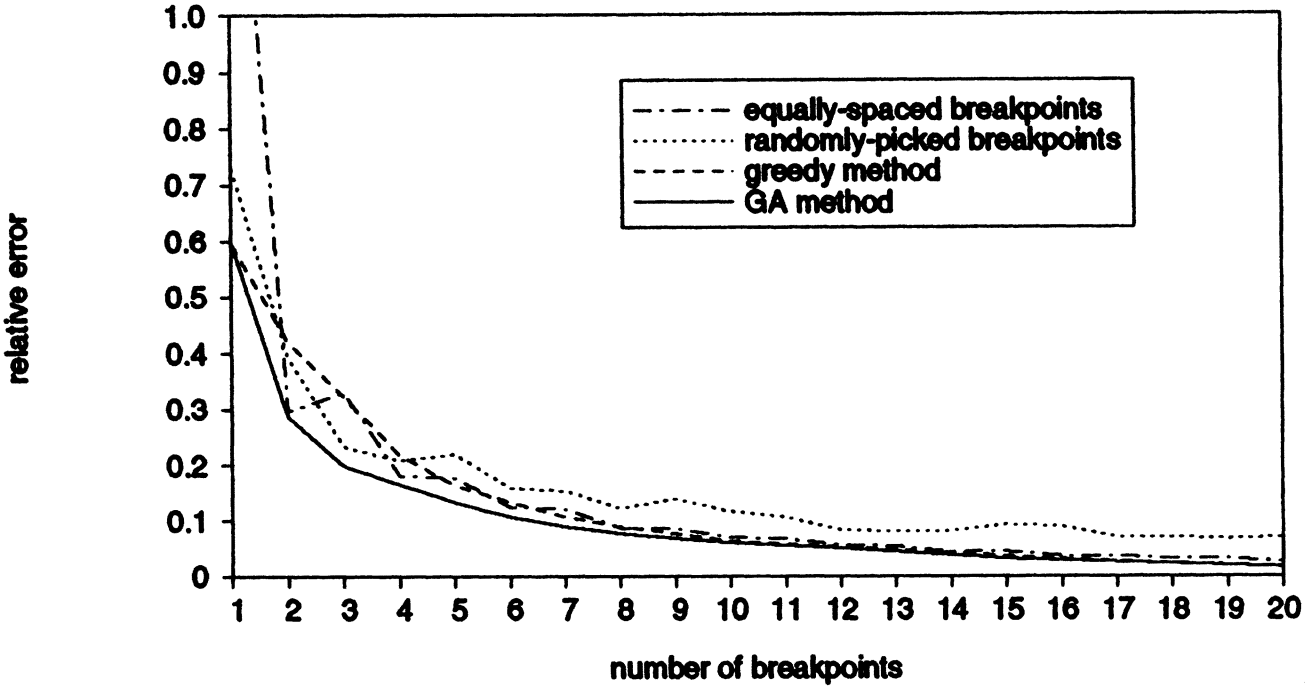
tack of the first harmonic includes some noise, and the fifth harmonic quickly disappears after the start of the tone.

In Figure 8, the relative error vs. number of breakpoints is compared for each method. Since the amplitude envelopes are extremely smooth, picking equally spaced breakpoint times is a good strategy in this case. With four or more breakpoints, it performs about as well as the greedy strategy. The random method gives the worst results, while the GA consistently yields the best performance over the breakpoint range. Note also that the GA's results for three to six breakpoints are better than those of the other three methods. Figure 9 shows GA approximations for different numbers of breakpoints. As more breakpoints are added, the envelope curves converge to the smooth decays of the originals. Even twelve breakpoints is not enough to model the noise in the peak of the first harmonic, but this is less important perceptually than the shape of the decay.

Guitar

Finally, we consider a forte guitar tone at 110 Hz (A2) with a duration of 7.8 sec. As shown in Figure

Figure 8. Errors returned by the breakpoint selection methods on the piano. The same breakpoints are used for each harmonic.

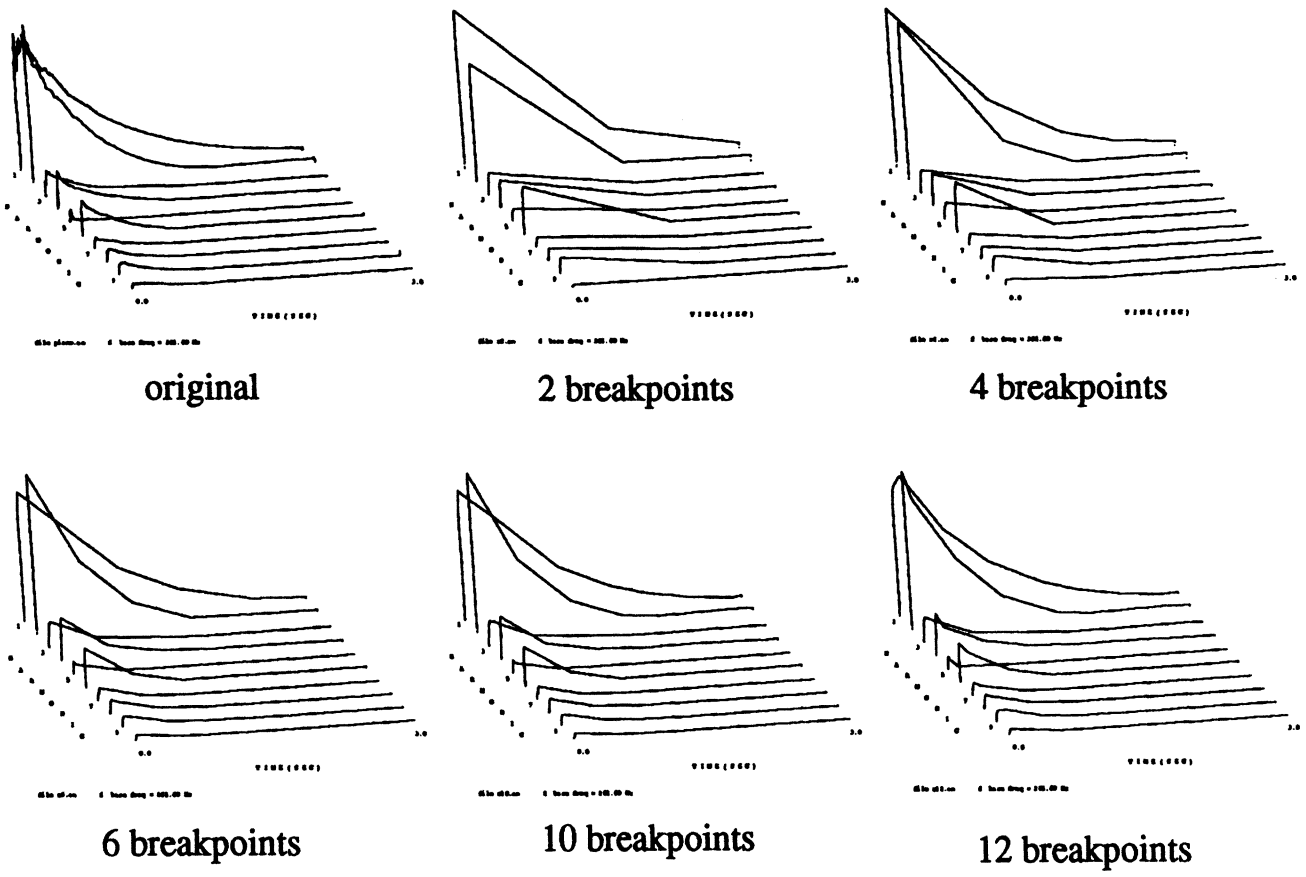


10, the amplitude envelopes of the guitar tone's harmonics all have very rapid attacks. Some of the harmonics (e.g., 1, 3, and 6) go immediately into rather long decays, although the envelope of harmonic 1 is peculiar in that a damped oscillation, at a rate of about 10 Hz, occurs immediately after the attack. Nevertheless, it is not clear that the ear actually perceives this oscillation. Other harmonics, such as the second and seventh, have "two-segment decays," where very rapid initial decay curves are followed by much slower decays. This is typical of the amplitude envelopes of string tones (Weinreich 1977). The transitions (or "knees") occur around 0.3–0.4 sec after the attack. In general, the higher harmonics tend to fade out faster than the lower ones, leaving the first, third, and sixth harmonics to dominate the sound of the final decay.

The results of the various breakpoint selection procedures are summarized in Figure 11. The equally spaced breakpoint selection procedure returned errors of more than 100 percent for seven or fewer breakpoints; the strategy simply did not select breakpoints in places that properly characterized the decay pattern. The other three methods performed about the same as one another up to three breakpoints, but the GA-selection method significantly outperformed the others in the range of four to seven breakpoints. For eight or more breakpoints, the greedy and GA methods gave about the same results.

As Figure 12 illustrates, the GA strategy indeed manages to find a reasonable solution with as few as four breakpoints. The first few breakpoints focus on the tone's decay, ignoring the attack completely. Since overestimating the amplitudes can cause relative errors greater than 100 percent (e.g., the equally spaced solutions for seven or fewer breakpoints), it makes sense to conservatively build up the decay first. The first and third harmonics of the three-breakpoint GA approximation look good and also result in a good sound, if you ignore the attack. The four-breakpoint solution sounds very much

Figure 9. Harmonics 1–10 of GA line-segment approximations to a piano tone.



like the original, while the five-breakpoint version shortens and refines the initial decay of the first and third harmonics. Indeed, this five-breakpoint version sounds quite similar to the original tone. Comparable-sounding results require ten breakpoints for the greedy method, and more than 20 breakpoints using equal spacing or random selection. Clearly, GA optimization of the breakpoint times pays off with the guitar sound.

Extensions to the Basic GA-Breakpoint Selection Method

Extensions to the basic GA breakpoint selection method include independent optimization of the breakpoint times for each harmonic's amplitude envelope, parameter optimization using quadratic (rather than line-segment) approximation, and optimization of breakpoint-amplitude scalars. In this section we present results that show how these extensions compare to the basic technique, and discuss whether they are worth the extra effort required to carry them out.

Should We Give Each Harmonic Its Own Set of Breakpoint Times?

When determining breakpoint times, we can either optimize one set common to all harmonics' amplitude envelopes, or, as many researchers have done in the past, we can find a different set for each amplitude envelope. In an earlier section of this article we examined the shared-breakpoint-time approach, which takes advantage of the fact that most acoustic instrument tones have relatively "synchro-

Figure 10. Amplitude envelopes for harmonics 1–5 (left) and 4–10 (right, different scale) of the guitar tone played at 110 Hz.

Figure 11. Errors returned by the breakpoint selection methods on the guitar. The same breakpoints are used for each harmonic.

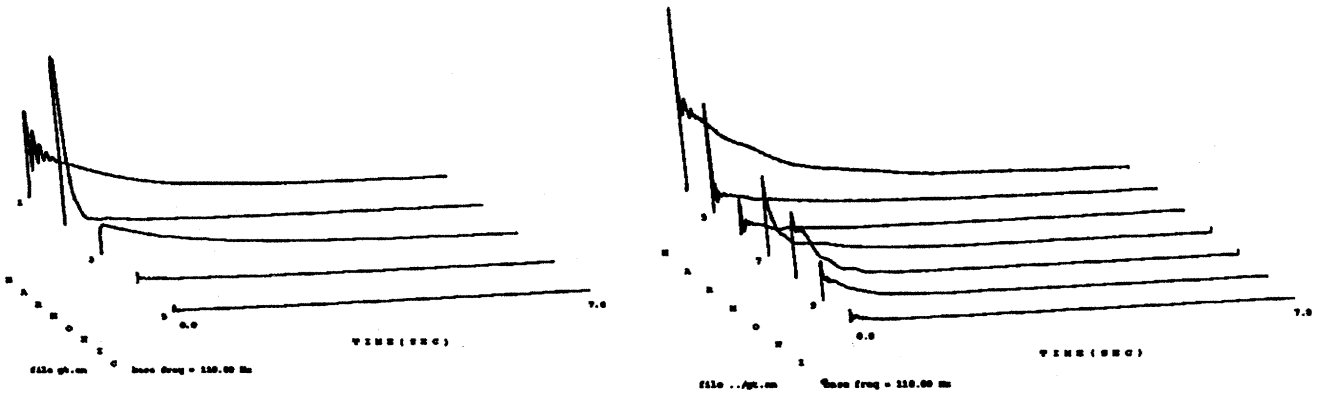


Figure 10

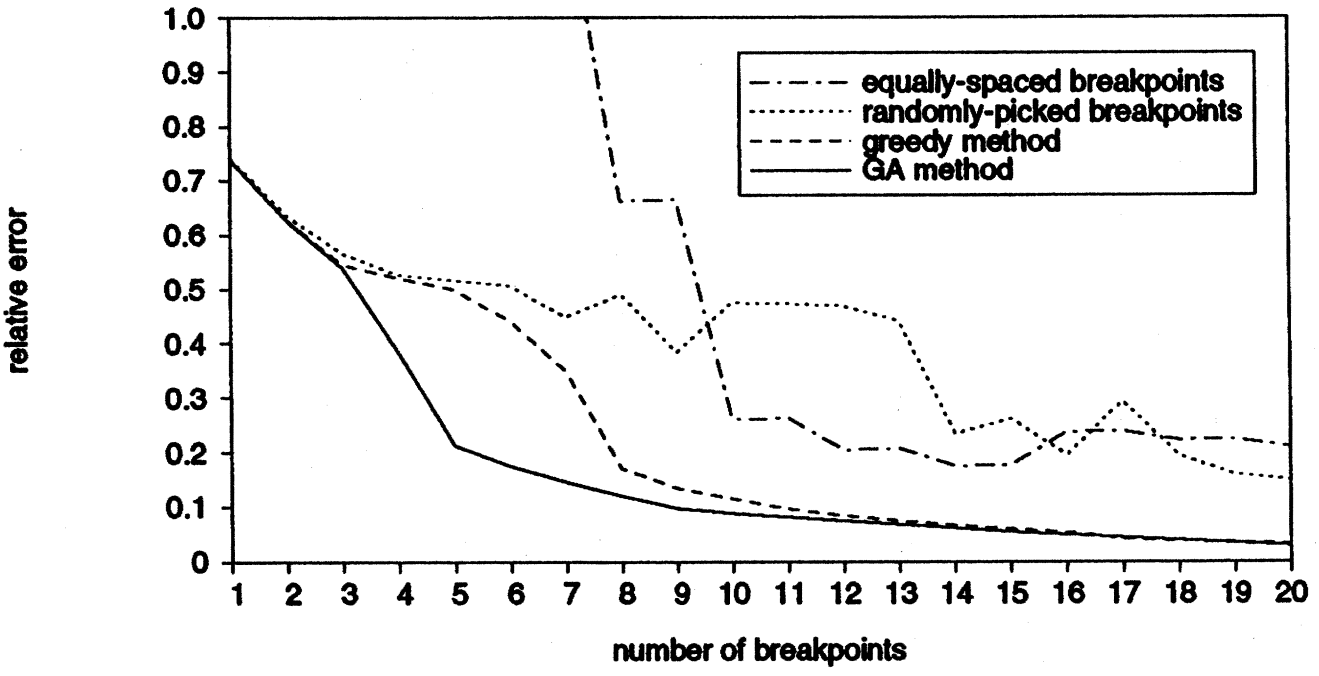
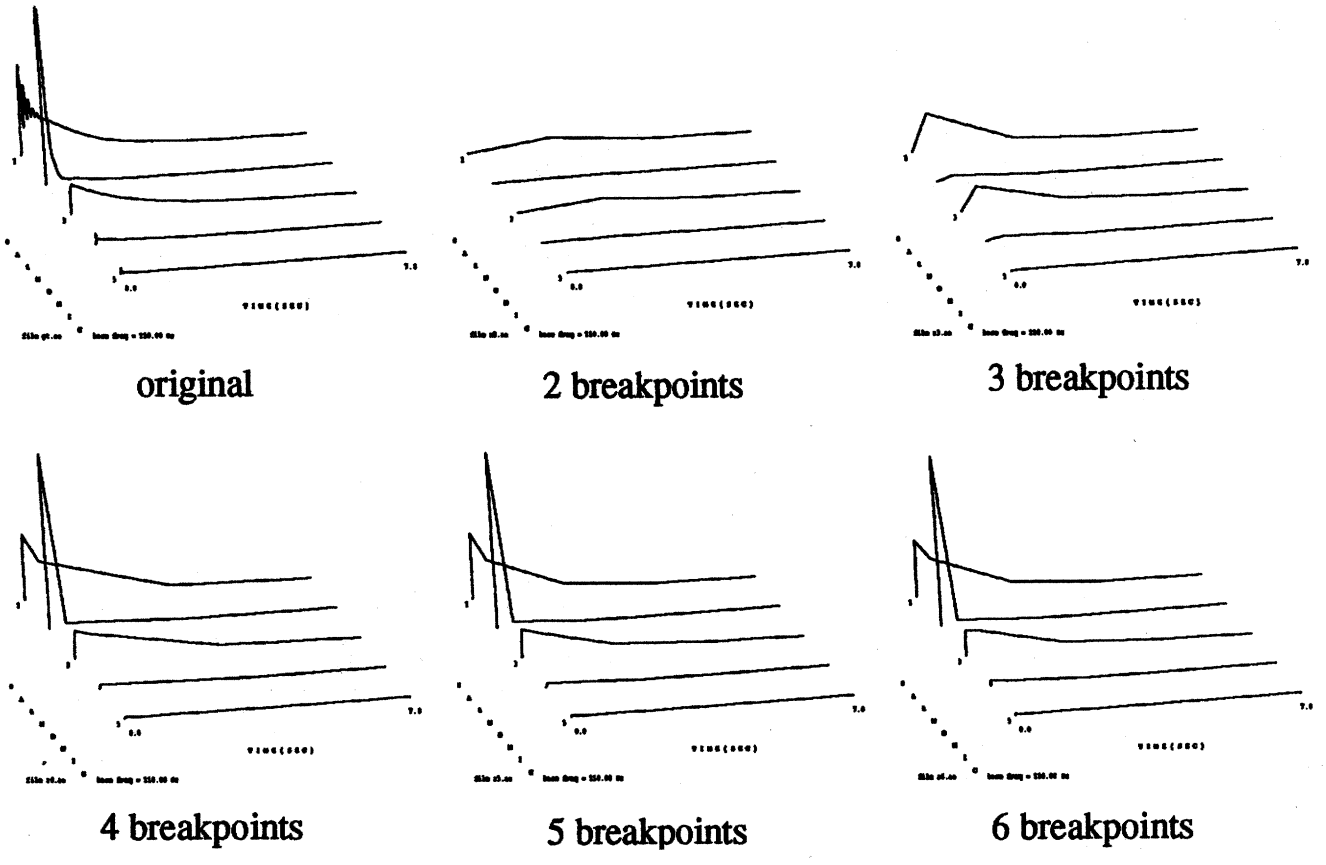


Figure 11

nized" amplitude envelopes across harmonics, as exemplified by the trumpet data of Figure 4. Note that even though the amplitude envelopes of the trumpet tone's third and fourth harmonics have vastly different shapes from the first and second, their slopes change at roughly the same time (i.e., all four harmonics complete their initial rise and decay together). The shared-breakpoint approach takes advantage of this natural potential data reduction.

However, in this section we will consider the problem of determining independent sets of breakpoint times that define customized sequences of line segments for fitting the shape of each enve-

Figure 12. Harmonics 1–4 of GA piecewise-linear approximations to the guitar.



lope. The advantage of this approach is that if most of the amplitude envelopes have very different shapes, each one can be optimized individually, possibly resulting in fewer line segments overall. For our three musical sound examples, however, we shall see that this method requires about twice as much data, and, it turns out, far more computation.

We can easily extend the GA method to optimize independent breakpoints. While we could also extend the simple, random, and greedy methods, we will only consider the GA method for comparison here, since we believe it gives the best performance for most situations.

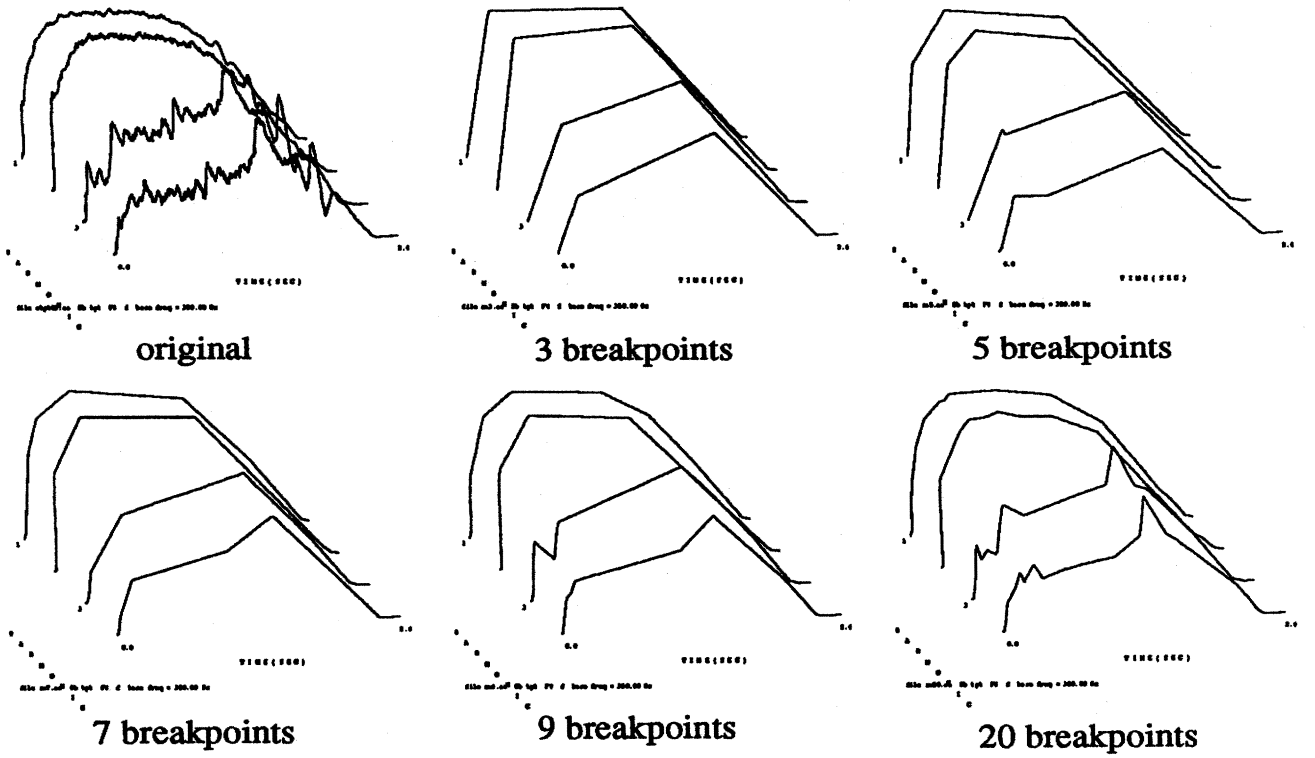
For independent breakpoint-time selection with the GA, we individually optimize each of the harmonics' amplitude envelopes to find its best set of breakpoints. We modify the fitness function of Equation 3 to:

$$\text{Relative Error}_k = \frac{w_1}{\text{PeakFrame}_k - 1} \sum_{n=1}^{\text{PeakFrame}_k - 1} \sqrt{\frac{(a_k(n) - a'_k(n))^2}{a_k(n)^2}} + \frac{w_2}{N_{\text{frames}} - \text{PeakFrame}_k + 1} \sum_{n=\text{PeakFrame}_k}^{N_{\text{frames}}} \sqrt{\frac{(a_k(n) - a'_k(n))^2}{a_k(n)^2}} \quad (4)$$

where k is the harmonic we are approximating, and w_1 and w_2 are again weights that we generally set to 0.5. Also, note that each harmonic now has its own peak-amplitude frame number, PeakFrame_k.

The GA's independent breakpoint selection for the trumpet tone is shown in Figure 13. Though the improvements in the envelopes over those in Figure 6 may appear subtle, tones synthesized with independent breakpoint times do indeed sound much better for the same number of breakpoints per envelope. For example, the resynthesis using five independent breakpoint times sounds quite like the original, and, in our judgment, results in

Figure 13. Harmonics 1–4 of original trumpet, and GA-optimized line-segment approximations with independent breakpoints.



about the same quality as the resynthesis using nine shared breakpoint times. Also, the use of independent breakpoint times results in a significantly lower objective error than does the use of their shared counterparts, as shown in Figure 14.

However, the comparison of Figure 14 is not really fair, because for the same number of breakpoints per envelope, the independent-breakpoint-time method requires about twice as much data as the shared-time approach. (The independent method requires storage of both the breakpoint times and amplitudes for each harmonic; the shared-time method only requires storage of the breakpoint amplitudes and one set of times.) Figure 15 shows the errors for both cases plotted against the total amount of data required. From this perspective, we see that the shared-breakpoint-time method does better (particularly for small amounts of data) than the independent-breakpoint-time method.

As in Figures 14 and 15, Figure 16 compares the error functions of shared and independent breakpoint times for the guitar tone. In this case, when we factor in total storage requirements, we find that the shared-breakpoint-time selection method does much better than the independent method.

The independent-breakpoint-time approach also requires much more computation than the shared-time method, since it requires a separate GA run for each harmonic. Since the guitar needs about 80 harmonics to properly characterize its attack, optimization of independent breakpoint times requires almost two orders of magnitude more computation time than does optimization of shared times. These runs can take hours on an SGI Indigo computer, one of the fastest desktop computers currently available.

Perhaps the biggest advantage of using shared breakpoint times is that they lend themselves to efficient linear wavetable-interpolation synthesis, as shown in Figure 17. With this method, we first con-

Figure 14. Errors returned by the GA method for shared and individual breakpoint-time approximations to the trumpet's amplitude envelopes.

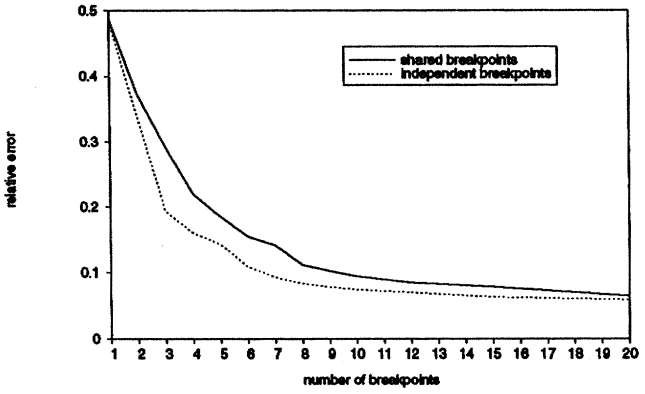
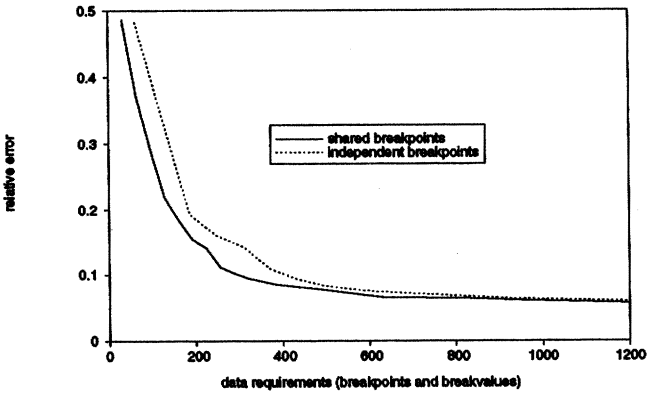


Figure 15. Relative error vs. data requirements for the shared and independent breakpoint-time methods, for the trumpet tone.



struct wavetables by Fourier synthesis using the per-harmonic amplitude data corresponding to each of the times. Synthesis simply proceeds by linear interpolation, or cross-fading, between pairs of these wavetables. As discussed in the Appendix, the method requires that all the partials have an exact harmonic relationship, and that the phases used to construct the wavetables agree. Note that an inverse fast Fourier transform (IFFT) can be used to speed the loading of the wavetables.

Wavetable interpolation (Chamberlin 1980; Serra, Rubine, and Dannenberg 1990) reduces synthesis computation by an order of magnitude over additive synthesis, so this is a major advantage. In the work of Xavier Serra, Dean Rubine, and Roger Dannenberg, breakpoints were added as needed to keep the error below some threshold. In contrast, our method optimizes a predetermined number of breakpoints in order to minimize the error. A related method, described by Horner, Beauchamp, and Haken (1993a), also optimizes a predetermined number of time points, but in that case the amplitude envelopes are not restricted to simple linear cross-fade functions, and normally all wavetables are used simultaneously.

Approximating Amplitude Envelopes with Quadratic Curves

So far we have only considered piecewise-linear fits. What about quadratic curves and higher-order approximations? For a fixed number of breakpoints, we can fit a curve as easily as a straight line between two points, but is the extra data worth it? We will address this issue in this section.

The GA method can be extended for fitting curves as well as straight lines, using a succession of parabolas. We have done this in the following way. First, we find an extra time value between each breakpoint-time pair, through which we fit a quadratic curve, so, in addition to the N breakpoint times, we optimize $N + 1$ additional times to define the quadratic curves. (This increases the running time of the GA by about a factor of two.) Then, for each candidate solution of times (in ascending order), we take every other one as a curve point and as a segment breakpoint, respectively (see Figure 18). The breakpoint times correspond to the ends of the parabolas, where the greater breakpoint time of each parabola corresponds to the lesser breakpoint time of the next parabola. Finally, for each time triplet, consisting of Brkpt_i, Curvept_{i+1}, Brkpt_{i+1} (note that we start with $\text{Brkpt}_0 = 0$, which is predetermined), we take values from the original data and exactly fit a quadratic curve through the points. This avoids any discontinuities at the breakpoint junctures. As an example, Figure 19 shows a piecewise-quadratic fit of the first harmonic of the trumpet.

A GA-optimized piecewise-quadratic fit to the trumpet tone's time-varying spectrum is shown in Figure 20. The shapes look aesthetically more pleasing (e.g., note the seven-breakpoint example) than

Figure 16. Relative error vs. number of breakpoints (left) and data requirements (right) for shared and independent breakpoint times, for the guitar tone.

Figure 17. Cross-fade amplitude envelopes for wavetable resynthesis (using shared breakpoints).

Figure 18. Sorted breakpoint and curvepoint times coded for the GA.

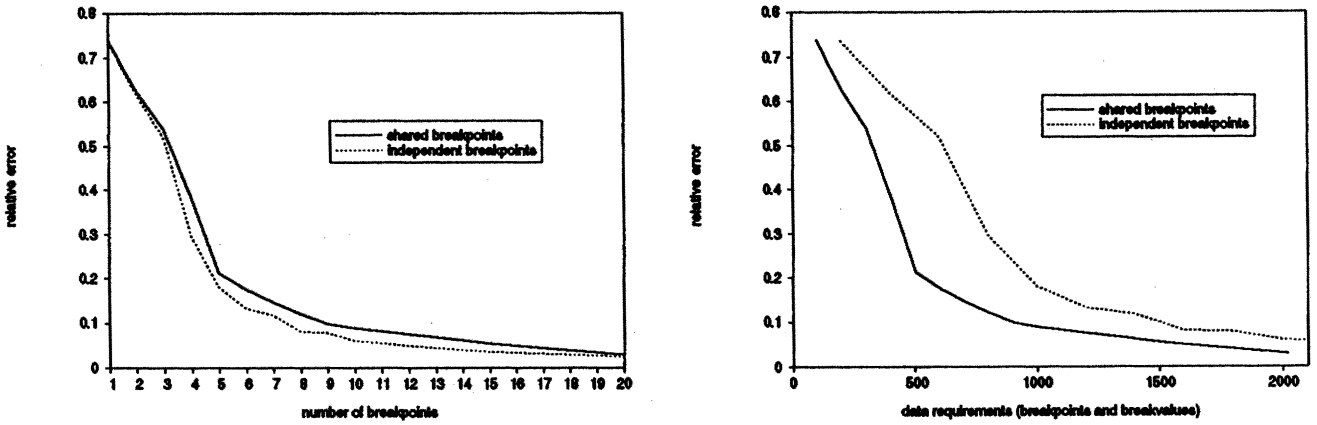


Figure 16

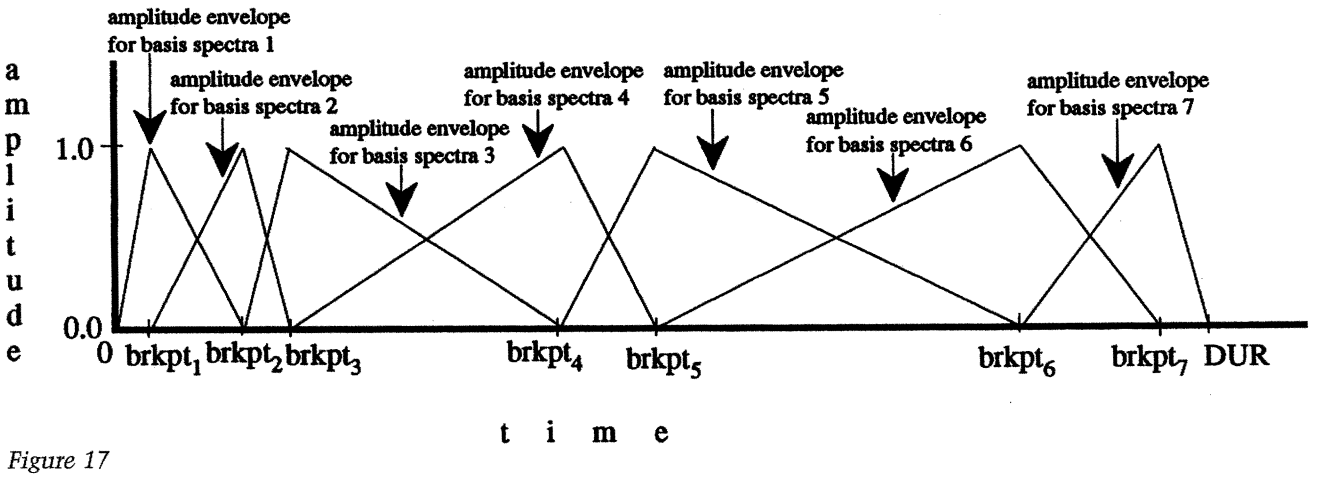


Figure 17

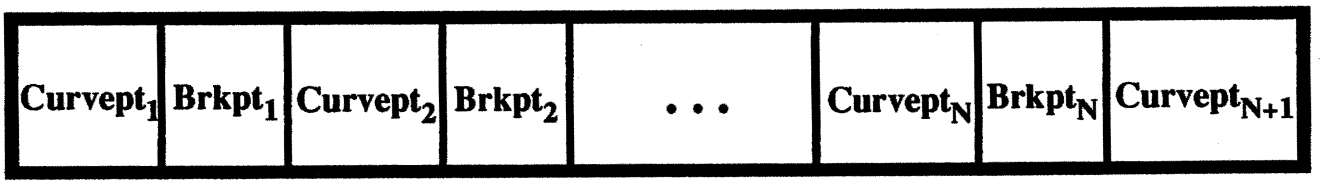


Figure 18

their respective piecewise-linear approximations. The rounded outline of the first and third harmonics, in particular, matches the original quite well. Also we see that the 9- and 20-breakpoint approximations capture most of the important details of the third and fourth harmonics quite well.

Nevertheless, when compared in terms of data storage, the line-segment and quadratic approximations result in almost identical synthesis quality. Figure 21 shows the relative errors for the two approximation methods plotted against the number of breakpoints and the total data requirements. The

Figure 19. Piecewise-quadratic approximation of the amplitude envelope of the trumpet's first harmonic.

Figure 20. Harmonics 1–4 of original trumpet and GA-optimized piecewise-quadratic approximations (with shared breakpoint times).

Figure 21. Relative error vs. number of breakpoints (left) and data requirements (right) for line-segment and quadratic approximations of the trumpet.

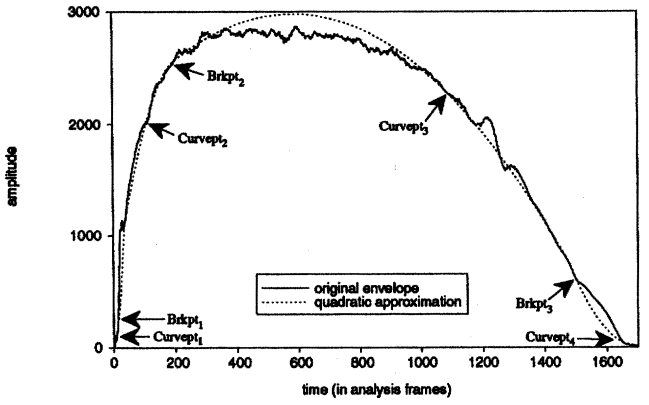


Figure 19

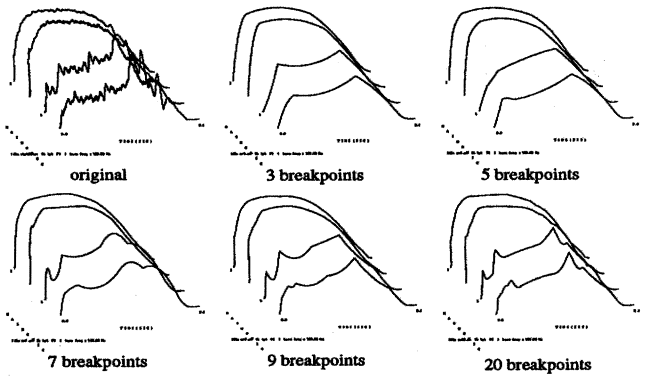


Figure 20

quadratic approach shows some significant improvements over the linear method for the number of breakpoints, but not for data requirements. This makes sense, because both methods find breakpoint times where the original values are exactly matched, the only difference being that the PLA method linearly interpolates between adjacent breakpoints, whereas the quadratic-approximation method fits parabolas through sequences of three points. Moreover, parabolas are no more common than straight lines in these types of data. Therefore, we would expect them to use similar amounts of data for fits of similar quality.

Informal listening tests on synthetic tones confirm the results shown in Figure 21. Tones synthesized using the quadratic approximations sound about the same as those resulting from PLAs when the same amount of data is used. Therefore, on this basis, we see no improvement using quadratic rather than line-segment approximations.

We can construct cross-fade envelopes for implementing a quadratic approximation analogous to the wavetable-synthesis cross-fade envelopes shown in Figure 17. However, quadratic cross-fade envelopes must be curves rather than simple pairs of linear ramps. Figure 22 shows the cross-fade envelopes needed to implement the quadratic approximation shown in Figure 19. We can efficiently implement quadratic and higher-order cross-fade envelopes using difference equations (Maher 1991). Third-order

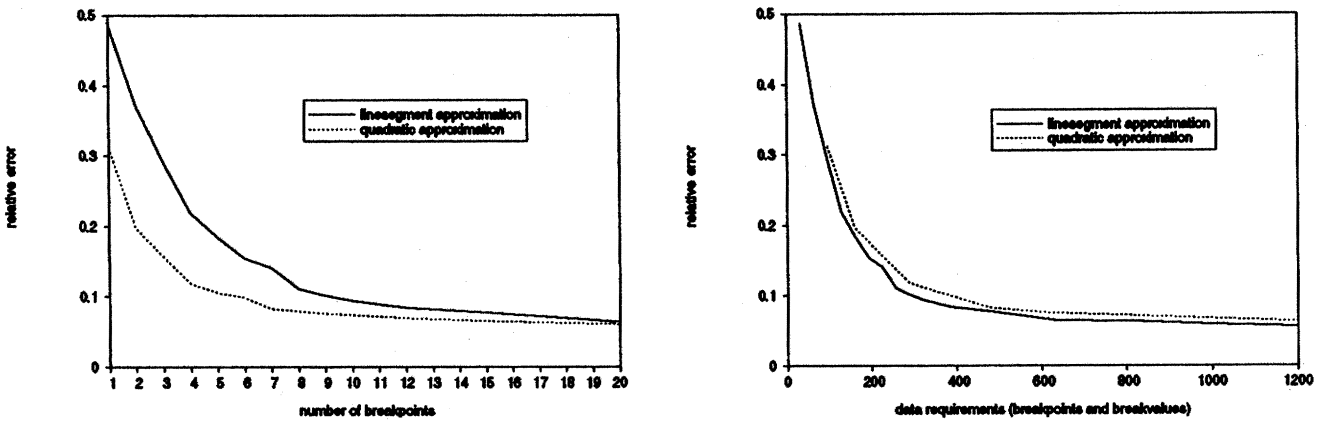
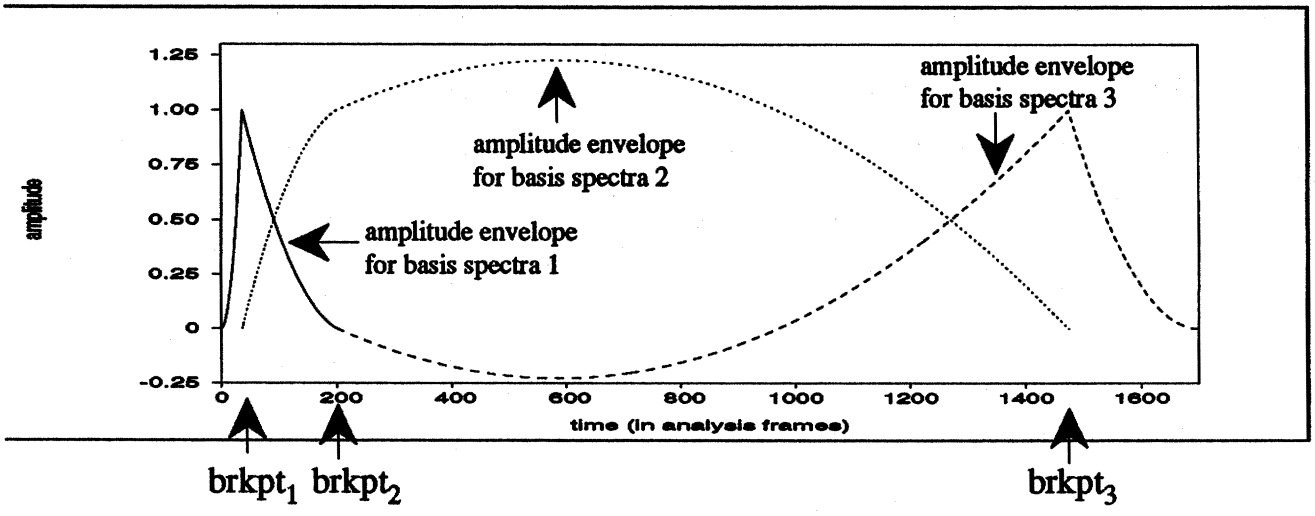


Figure 21

Figure 22. Cross-fade amplitude envelopes for wavetable resynthesis of a three-breakpoint quadratic approximation for the trumpet tone (shared breakpoints).



spline envelopes can be useful in situations where first-derivative discontinuities resulting from PLA synthesis are not acceptable, such as when a high-pass filter is used as the final stage of the synthesis.

Note that the quadratic cross-fade curves often include values greater than 1.0 and less than 0. This is because the quadratic approximation allows the curve to move away from the next breakpoint before converging with it, which occurs between the second and third breakpoints in Figure 19. When quadratic envelopes cross-fade, complex cancellations can occur; hence they are in general more difficult to interpret than linear cross-fade envelopes. This is another advantage of using the simple PLA approach. When quadratic envelopes have values greater than 1.0, they emphasize differences between the breakpoint spectra. Unless everything is perfectly adjusted, these differences could generate spectra quite different from any of the original spectra.

Scalar Optimization of Breakpoint Amplitudes

So far we have only optimized the breakpoint times and have simply and somewhat arbitrarily used the original amplitude values that occur at these times for each breakpoint. However, can we do better if we optimize the breakpoint amplitude values as well—but is the extra computation worth it?

There are two main approaches to optimizing breakpoint-amplitude values. Either we can optimize all the harmonics' amplitudes individually at each breakpoint time, or we can try to find a breakpoint-amplitude scalar by which to uniformly multiply all of the originally chosen amplitudes for each harmonic at each breakpoint time. The first approach is too ambitious. It requires optimizing from 30 to 100 parameters for each breakpoint. The search space is simply too large to hope for much success. However, the second approach, which we describe below, adds only one additional parameter for each breakpoint, and is therefore much more practical.

We can modify the genetic algorithm to find both breakpoint times and breakpoint-amplitude scalars. Figure 23 depicts the GA encoding of the parameters. We allow the breakpoint-amplitude scalars to range from 0.0 to 2.55 in increments of 0.01. We can thus encode each of the breakpoint-amplitude scalars using 8 bits, a little less than is required for the breakpoint times.

As Figure 24 illustrates, scalar optimization of breakpoint-amplitude values typically results in only minor improvements compared to simply retaining original values at the optimized breakpoint

Figure 23. Breakpoint times (after sorting) and breakpoint-amplitude scalars encoded for the GA.

Figure 24. Relative error for shared breakpoint-time optimization and for optimization of shared breakpoint times and amplitude scalar, compared for the trumpet tone.

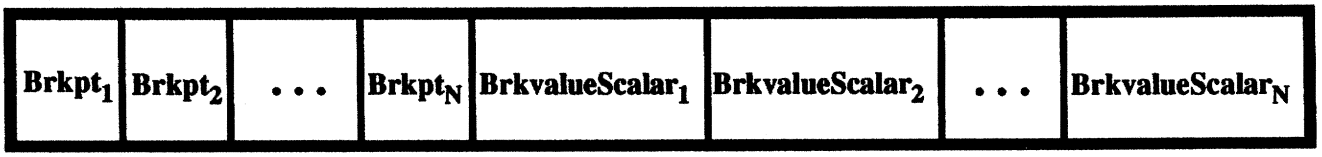


Figure 23

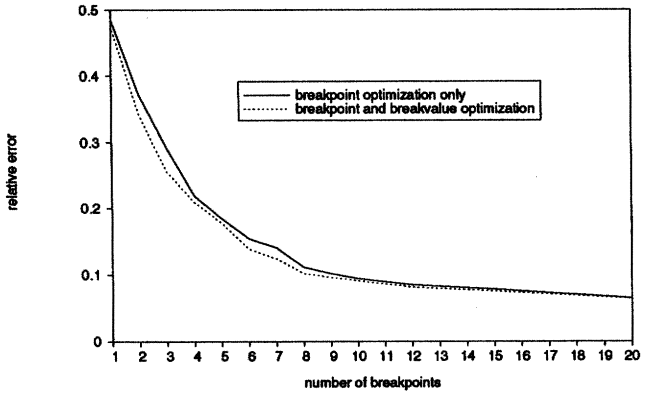


Figure 24

times. While the resulting breakpoint values do not cost us much in terms of data storage, we judge that the extra computation during optimization (about twice as much computation is required) is not justified by the subtle improvements that result. We conclude that breakpoint-amplitude optimization is simply not worth the extra effort.

Frequency Envelope Data Reduction

For tones with a "constant pitch," frequency envelopes usually make much more subtle contributions to overall timbral effects than do amplitude envelopes. Moreover, many sounds have very little inharmonicity. Therefore, representation of the individual harmonics' frequency envelopes by a single average frequency envelope, suitably scaled, is a useful method for additive-synthesis data reduction. For most harmonic musical tones, this data reduction results in a barely detectable audible difference from tones having individual envelopes (Charbonneau 1981). This is fortunate, because implementation of wavetable synthesis with shared breakpoints requires reduction of the frequency data to a single envelope. So, for this case, we define a weighted average (WA) frequency derived from the original partial amplitudes and frequencies using the formula

$$f(t) = \frac{\sum_{k=1}^{5} a_k(t)(f_k(t)/k)}{\sum_{k=1}^{5} a_k(t)} \qquad (5a)$$

This is under the assumption that the instantaneous frequencies of the harmonics can be approximated using

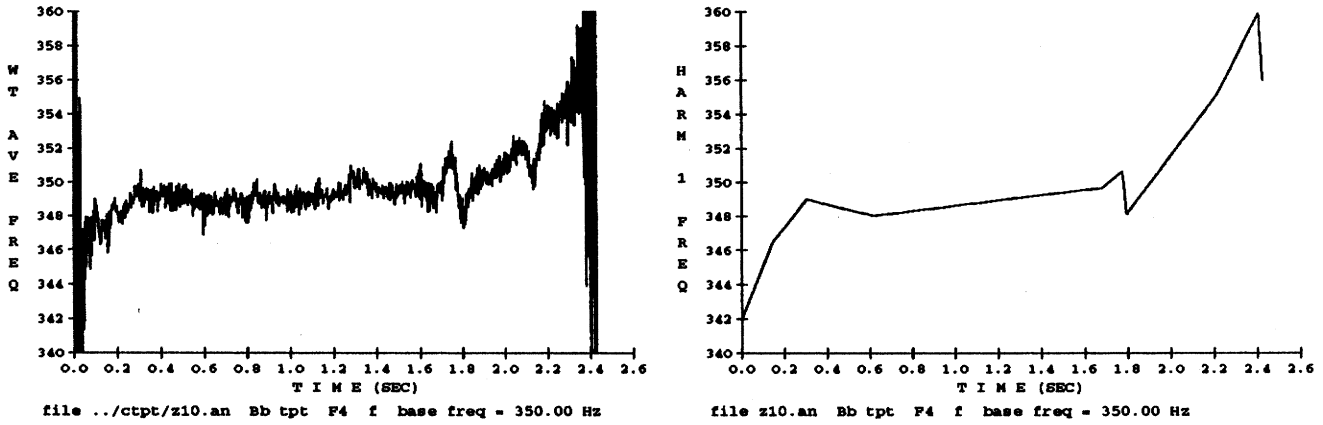
$$f_k(t) \cong kf(t). \qquad (5b)$$

Note that the formula for the WA frequency uses the harmonics' amplitudes as weights. Therefore, harmonics in the range of 1–5 with the strongest amplitudes have the largest "votes" in determining the WA frequency. This formulation avoids the problem that arises when a low-amplitude fundamental results in a poor (i.e., noisy) frequency measurement.

We can reduce the data further by using a PLA method. In fact, we have found that the same GA PLA method that works well for amplitude-envelope approximation, with slight modifications, works equally well for approximating WA-frequency envelopes. One difference is that, unlike amplitude optimization, we need to find non-zero frequency values for the beginnings and endings of tones, and we let the GA optimize these parameters as well as the breakpoint times.

The primary difference from the amplitude-envelope PLA method is that frequency-envelope optimization requires a different error function. Instead of the relative error measure of Equation 3, we use an absolute error measure:

Figure 25. Original and a ten-breakpoint piecewise-linear approximation of the trumpet's weighted-average frequency envelope function.



$$\text{Weighted Absolute Error} = \frac{1}{N_{\text{frames}}} \sum_{n=1}^{N_{\text{frames}}} \sqrt{\sum_{k=1}^{N_{\text{har}}} a_k(n)^2}\, |f(n) - f'(n)| \quad (6)$$

where $f(n)$ is the original WA frequency at the nth time frame and $f'(n)$ is the PLA version of the WA frequency. We wanted to de-emphasize the attack and decay in this error function, since frequency functions are quite noisy (due to the lack of definition and poor signal-to-noise ratios) in these regions. This de-emphasis is accomplished by using the RMS amplitude as a weight at each frame in the error calculation.

Because of this different error function and the necessity for non-zero end values, the frequency breakpoint times must be picked independently of those chosen for the amplitude envelopes. However, the computational load is considerably less, since the optimization search space is confined to a single set of frequency values rather than to one for each harmonic.

Figure 25 shows the trumpet tone's original WA frequency envelope and a ten-breakpoint match to it. The wide deviations in the original at the beginning and end of the tone are not as important as they might appear, since they occur during low-amplitude portions of the tone, and are probably artifacts of the time-varying FFT. The approximation captures the main nuances of the original function quite well, including the overall rise in pitch in the attack and release.

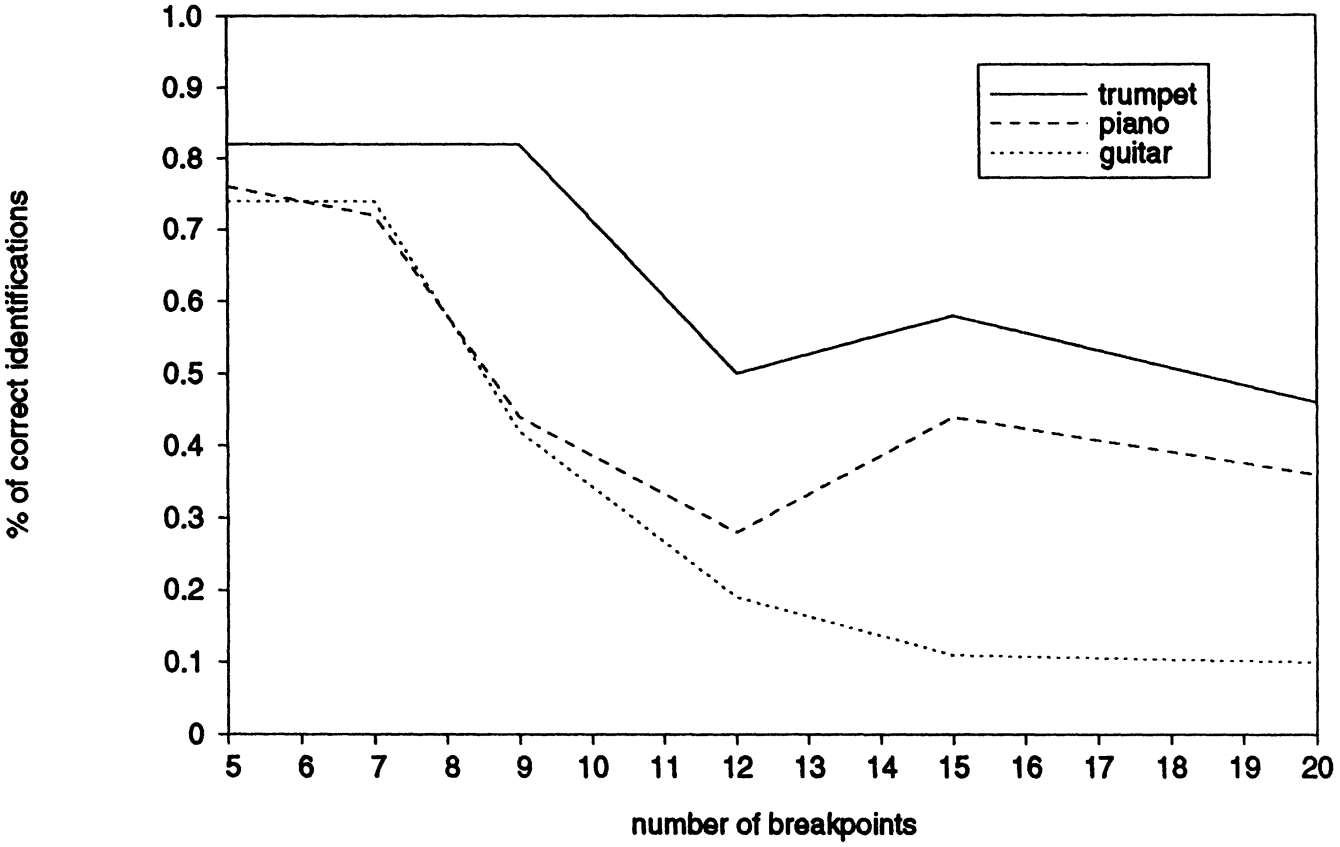
Results of Formal Listening Tests

So far we have evaluated the piecewise-linear approximations described above using admittedly artificial error functions, based on educated guesses about the perceptually important factors. Indeed, formal listening tests provide a more realistic way to evaluate the quality of the synthetic sounds. We have mentioned several informal listening tests made in the course of this investigation. For an overall evaluation of the success of the project, and to answer the basic question, How many breakpoints does one need to satisfactorily synthesize a given tone? we have carried out formal psychoacoustic listening tests.

Our listening tests used the three original acoustic tones (the same trumpet, piano, and guitar tones discussed previously, with 31, 42, and 100 harmonics respectively) and three corresponding synthesized versions. The synthetic tones used GA-determined breakpoints for the amplitude envelopes, but the original tones' frequency envelopes. Each of these six tones was presented five times during the test, for a total of 30 trials.

We used five subjects (including the two authors), all of whom had strong backgrounds in both music and synthesis. We presented the tones in a random order, mixing the synthetic tones with the originals. The tests were automatically scored by the same program that played the sounds. Most of the

Figure 26. The fraction of trials in which the average subject was able to correctly identify synthetic tones from a test set of both real and synthetic tones, plotted against the number of breakpoints.



tests were taken in a sound attenuation chamber with high-quality Beyer headphones (model DT48).

The results of our formal listening tests for the trumpet, piano, and guitar tones are shown in Figure 26. The fraction of trials in which the average subject was able to correctly identify synthetic tones is plotted against the number of breakpoints used for synthesis. In order to penalize listeners for guessing, we subtracted the fraction of "false alarms" (when the listener incorrectly identified an original tone as synthetic) from the raw score for correct identification of the synthetic tones.

Figure 26 reveals that the trumpet tone required twelve breakpoints to fool the average subject about 50 percent of the time (a score of 50 percent corresponds to random guessing). The piano and guitar both required only about nine breakpoints for a similar-quality match. However, in general, the results are quite dependent on the listener and listening conditions. With excellent listening conditions, our best listeners required 25 breakpoints for the trumpet, 12 for the piano, and 20 for the guitar.

Conclusion

We have found that a genetic-algorithm-based procedure determines the best shared breakpoint times for piecewise-linear approximation of harmonics' amplitude envelopes. The method minimizes the relative error between a set of original amplitude curves and their line-segment approximations by systematically exploring a wide variety of local optima within the space of possible breakpoint times. The GA breakpoint selection method is also easily extended for determining PLA frequency envelopes. The GA calculations only take a minute or two on an SGI Indigo computer, depending on the number

of breakpoints involved. Of all the methods we have investigated, the GA has consistently given superior results, so we consider this method to be the best when looking for optimum solutions for specific numbers of breakpoints. Indeed, for hardware synthesis, where the number of breakpoints is often quite limited, the GA approach is clearly the best.

However, when ten or more breakpoints are available, the greedy method of breakpoint selection also performs very well, generally converging to a solution similar to that produced by the GA. Since the greedy approximation method usually computes faster than the GA (only taking a few seconds for each iteration), it is perhaps the best method for software synthesis, where fast analysis/synthesis could be important and the possible need for a few extra breakpoints (to reduce the error to a level as low as the GA would give) would not be a great concern.

While an objective fitness function is necessary to compute good PLA solutions, these results still do not tell us how many segments are needed for a "good synthesis." However, formal listening tests can yield realistic information on the quality of the synthetic tones. Our listening tests showed that twelve GA-optimized breakpoints were needed for a trumpet tone, and nine for piano and guitar tones, for the average subject to confuse synthetic tones with original ones at least 50 percent of the time.

We investigated various extensions of the basic GA technique, but found they were not worth their additional costs. For the same amount of parametric data, neither the independent PLA breakpoint-time selection method nor the piecewise-quadratic approximation method yielded better matches than those obtained with the basic linear shared-breakpoint method. Also, breakpoint-amplitude scalar optimization, while simple enough to carry out, did not improve the basic results enough to warrant its use.

Moreover, an important argument for using shared breakpoint times with linear connecting segments is that it allows for a mathematically equivalent, efficient alternative to additive synthesis: wavetable interpolation synthesis.

In summary, genetic algorithms provide a powerful general tool for data reduction of spectral data, as we have seen for the piecewise-linear approximation problem. We expect to find further spectral-modeling applications for the GA method in future work.

Acknowledgments

This work was supported in part by the Hong Kong Research Grant Council's Project HKUST597/94E. We would also like to thank Lydia Ayers for her comments on the drafts of this article.

References

Beauchamp, J. 1969. "A Computer System for Time-Variant Harmonic Analysis and Synthesis of Musical Tones." In H. von Foerster and J. Beauchamp, eds. *Music by Computers*. New York: John Wiley and Sons.

Beauchamp, J. 1993. "UNIX Workstation Software for Analysis, Graphics, Modification, and Synthesis of Musical Sounds." Preprint 3479, *Audio Engineering Society*.

Chamberlin, H. 1980. "Advanced Real-Timbre Music Synthesis Techniques." *Byte* (April): 70–94, 180–196.

Charbonneau, G. 1981. "Timbre and the Perceptual Effects of Three Types of Data Reduction." *Computer Music Journal* 5(2):10–19.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, Massachusetts: Addison-Wesley.

Grey, J., and J. Moorer. 1977. "Perceptual Evaluation of Synthesized Musical Instrument Tones." *Journal of the Acoustical Society of America* 62:454–462.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press.

Horner, A., J. Beauchamp, and L. Haken. 1993a. "Methods for Multiple Wavetable Synthesis." *Journal of the Audio Engineering Society* 41(5):336–356.

Horner, A., J. Beauchamp, and L. Haken. 1993b. "FM Matching Synthesis with Genetic Algorithms." *Computer Music Journal* 17(4):17–29.

Maher, R. C. 1991. "A Method for Envelope Warping in Digital Audio Synthesis." *Journal of the Audio Engineering Society* 39(12):934–944.

Moorer, J. A., J. Grey, and J. Snell. 1977. "Lexicon of Analyzed Tones—Part I: A Violin Tone." *Computer Music Journal* 1(2):39–45.
Moorer, J. A., J. Grey, and J. Strawn. 1977. "Lexicon of Analyzed Tones—Part II: Clarinet and Oboe Tones." *Computer Music Journal* 1(3):12–29.
Moorer, J. A., J. Grey, and J. Strawn. 1978. "Lexicon of Analyzed Tones—Part III: The Trumpet." *Computer Music Journal* 2(2):23–31.
Risset, J., and M. Mathews. 1969. "Analysis of Musical Instrument Tones." *Physics Today* 22(2):23–30.
Serra, M.-H., D. Rubine, and R. Dannenberg. 1990. "Analysis and Synthesis of Tones by Spectral Interpolation." *Journal of the Audio Engineering Society* 38(3):111–128.
Strawn, J. 1980. "Approximation and Syntactic Analysis of Amplitude and Frequency Functions for Digital Sound Synthesis." *Computer Music Journal* 4(3):3–24.
Weinreich, G. 1977. "Coupled Piano Strings." *Journal of the Acoustical Society of America* 62:1474–1484.

Appendix: Sufficient Conditions Under Which Wavetable Interpolation Synthesis Is Mathematically Equivalent to Additive Synthesis

A brief proof of these conditions has been given (Serra, Rubine, and Dannenberg 1990). Here we present a more-detailed proof, which includes the possibility for time-varying frequencies.

In Equation 1, above, a formulation for an additive sinusoidal representation of the original signal was given, where $a_k(t)$ and $f_k(t)$ are the harmonics' amplitudes and frequencies, respectively. In our piecewise-linear approximation (PLA), we replace $a_k(t)$ by $a'_k(t)$, and the $f_k(t)$ are assumed to be related to some time-varying fundamental frequency so that $f_k(t) = kf(t)$. Let the breakpoint coordinates for each $a'_k(t)$ be given by $\{t_{A_{k,j}}, A_{k,j}\}$. Then, in general, for any time t within the time interval $(t_{A_{k,j}}, t_{A_{k,j+1}})$, $a'_k(t)$ can be written

$$a'_k(t) = (1 - \alpha(t))\, A_{k,j} + \alpha(t)\, A_{k,j+1} \qquad \text{(A.1a)}$$

where

$$\alpha(t) = (t - t_{Ak,j}) \,/\, (t_{Ak,j+1} - t_{Ak,j}) \qquad \text{(A.1b)}$$

is the linear interpolation or cross-fade function of time.

Let $\phi_k(t)$ be the total instantaneous phase of the kth harmonic. The approximated signal is then

$$s'(t) = \sum_{k=1}^{N_{\text{har}}} a'_k(t) \cos(\phi_k(t)) \qquad \text{(A.2a)}$$

$$= \sum_{k=1}^{N_{\text{har}}} (1 - \alpha(t))\, A_{k,j} \cos(\phi_k(t)) \qquad \text{(A.2b)}$$

$$+ \sum_{k=1}^{N_{\text{har}}} \alpha(t)\, A_{k,j+1} \cos(\phi_k(t))$$

$$= (1 - \alpha(t)) \sum_{k=1}^{N_{\text{har}}} A_{k,j} \cos(\phi_k(t)) \qquad \text{(A.2c)}$$

$$+ \alpha(t) \sum_{k=1}^{N_{\text{har}}} A_{k,j+1} \cos(\phi_k(t))$$

for the same time interval. Now, if the frequency is steady, i.e., $\phi_k(t) = 2\pi kft + \theta_k$ (no time dependence), we can interpret the two summations as ordinary Fourier series that add to the fixed waveforms $W_j(2\pi ft)$ and $W_{j+1}(2\pi ft)$. Note that the phases θ_k should be the same for both waveforms. However, what happens when the frequency varies with time? In this case, we can write $\phi_k(t)$ as:

$$\phi_k(t) = k2\pi \int f(t)dt + \theta_k = k\Theta(t) + \theta_k. \qquad \text{(A.3)}$$

We now see that the Fourier series for W_j can be generalized to:

$$W_j(\Theta(t)) = \sum_{k=1}^{N_{\text{har}}} A_{k,j} \cos(k\Theta(t) + \theta_k), \qquad \text{(A.4)}$$

where $\Theta(t)$ is the waveform phase that takes on values between 0 and 2π. It is important to realize that the instantaneous phase of the wavetable is locked to the instantaneous phases of the individual partials, provided that $(\phi_k(t) - \theta_k)/k$ is the same for all partials—i.e., they have harmonically related frequencies. Thus, the necessity for harmonicity under this development.

Finally, combining Equations A.2c, A.3, and A.4, we arrive at:

$$s'(t) = (1 - \alpha(t))W_j(\Theta(t)) + \alpha(t)W_{j+1}(\Theta(t)) \qquad \text{(A.5)}$$

for each time interval $(t_{A_{k,j}}, t_{A_{k,j+1}})$, which is the desired result. W_j and W_{j+1} correspond to waveforms based on the harmonics' amplitudes that occur at the end points of the time interval and the same

initial phases θ_k. Thus, Equation A.5 should be interpreted as a linear interpolation between these two waveforms over the time interval while the phases of the wavetables are varying synchronously but independently of the interpolation functions.

For this to work, we emphasize that it is sufficient (and probably necessary) for each of the wavetables to be formed using the same set of per-harmonic starting (zero-table) phases θ_k, i.e., the phases must be consistent. However, the actual per-harmonic phases that occur at the end points of each interpolation time interval also depend on the function $\Theta(t)$. Note that when this function is zero or some multiple of 2p, the actual per-harmonic phases will be the same as the starting phases. Note also that it will generally be impossible to exactly match the phases of the original signal at the breakpoint times (other than at $t = 0$) using the wavetable interpolation method, since the instantaneous harmonic approximation ($f_k(t) \cong kf(t)$) is only an approximation to real behavior, which is almost always slightly inharmonic. However, for most quasi-harmonic sounds, this has little audible consequence.