Approximation and Syntactic Analysis of Amplitude and Frequency Functions for Digital Sound Synthesis

1. Introduction

Of the various models proposed and used for analyzing and synthesizing musical sound, additive synthesis is one of the oldest and best understood. In recent years, time-variant Fourier methods implemented as computer programs have made it possible to analyze digitally a large variety of tones from traditional musical instruments. Such research has led to a better understanding of the physical and perceptual nature of musical sound as well as to new techniques for digital sound synthesis.

The heterodyne filter (Moorer 1973) and the phase vocoder (Pettit 1976, Moorer 1978) provide time-varying amplitude and frequency functions for each harmonic of a tone being analyzed. This quickly leads to an almost innumerable increase in the amount of data used to represent the original tone. The question of reducing the amount of data without sacrificing tone quality is one of potential interest to hardware and software designers as well as musicians and psychoacousticians.

Copyright © 1980 John Brown.


One approach to data reduction has involved the use of line-segment approximations (Pitter 1989; Beauchamp 1969; Gery 1975), in which an amplitude or frequency function is represented by a relatively small number of line segments. Depending on the degree of reduction, these approximations often sound as though they were produced by the original instrumental source and in many cases cannot be distinguished perceptually from the original tone.

There is still no definitive answer to the question of how much data can be omitted without changing the tone significantly. The ultimate goal would be to use the smallest possible amount of data to produce a tone that would be perceptually indistinguishable from the (digital) recording of the original tone (Prong and Clark 1977). Gery was unable to explore the question of the degree of acceptable data reduction because at the time only analog tape recordings could be digitized for analysis by computer at the CCRMA for Computer Research and Analysis (CCRMA). Thus the synthesized tone could be discriminated from the original tone merely by the absence of tape hiss. Using the digital recording facility (Moorer 1977) it is now possible to record traditional musical instruments at CCRMA in digital form.

An important problem in data reduction concern the selection of features to be retained in th
Fig. 1. (a) The amplitude versus time function of the 5th harmonic of a single trumpet tone, analyzed by the heterodyne blue and normalized to a maximum amplitude of 1.0. The waveform represents an oscillating function on an arbitrary linear scale. This function has been chosen as an interesting test case for investigating methods of approximating transition of the waveform from the sine wave and the presence of glitch in the attack. Presumably such glitches play an important role in the timbre of the tone from which this function is derived. (b) The frequency versus time function for the same harmonic of the same tone as in (a). Both functions contain 230 points.

simplified representation of the amplitude and frequency waveforms. In this article, features will be used in a very narrow sense to mean components of functions that are presumably important perceptually. An example of this would be the so-called glitches which typically occur in tone bursts, such as those shown in Fig. 1 (cf. Spering and Clark [1967]; Moore, Gray, and Strauss [1978]). The central hypothesis of the work discussed here might be formulated as follows: it is possible to reduce time-varying amplitude and frequency functions derived from traditional instrument tones to some maximum number of line segments such that a digitally synthesized tone using such line segments is perceptually indistinguishable (according to some suitable measure) from a digital recording of the original tone. Reducing the number of line-segments further, that is, minimizing some function, results in tones which can be distinguished from the original.

Various manual and automatic algorithms for generating line-segment approximations were used in previous research. In the first half of this paper we will review several algorithms from the literature on pattern recognition which have been developed for analyzing such diverse data as contours on maps (Wiener [1961], electrocardiograms (Farabou and Poewer [1974]), chromosomes (Fe [1974], outlines of human heads [Kelly [1972]), and growth banding curves (Knecht [1961]), but which have not yet been applied to musical problems. After reviewing the difficulties inherent in such 'low-level' techniques for the problem at hand, preliminary results from a syntactic, hierarchical scheme for analyzing amplitude and frequency functions will be presented. Since Gray concluded that it was necessary to retain time-varying information for both frequency and amplitude functions (1970), a method for analyzing both will be discussed.

The algorithms which will be outlined below draw extensively from the literature on approxima-
2. Line Segments

Formally speaking, the various approximations will be treated in first-degree splines. The following definition has been adapted from that of Cox (1971).

Let \( f(t) \) be a sampled function, where \( t = \{ t_0, t_1, \ldots, t_n, t_{n+1} \} \) is a sequence of real numbers representing time defined at \( t = q \Delta T \). It is a sequence of real numbers representing time defined at \( t = q \Delta T \). Then a first-degree spline approximation \( s(t) \) to \( f(t) \) is given by

\[
\begin{align*}
s(t) &= a(t) + b(t)(t - a(t)), \quad t \leq a(t)
\end{align*}
\]

\[
\begin{align*}
s(t) &= a(t) + b(t)(t - a(t)), \quad t > a(t)
\end{align*}
\]

where \( a(t) = t_0 \) if \( t_0 \leq t \leq t_1 \) and \( a(t) = t_{n+1} \) if \( a(t) \) in also required to be continuous across \( t_0 \leq t \leq t_{n+1} \), with continuity defined as

\[
\Delta a(t) = a(t) - a(t - \\

Any \( a(t) \) is obviously continuous across \( t \in \{ t_0, t_1, \ldots, t_{n+1} \} \).

Furthermore, one important restriction has been adopted and concerns the endpoints of the line segments used to approximate a function. Each endpoint is required to be the same as the data points in the original waveform (Fig. 1), in terms of the definition given above:

For every \( i \geq 0 \) and \( i = 0 \) such that

\[
a(t_0) = a(t) \quad \text{and} \quad a(t_0) = a(t_0)
\]

The reasons for this restriction will become clear only at the very end of the paper. In the meantime, it will merely be mentioned again when appropriate.

Since the functions being approximated were only as follows, such requirements as the existence of the 1st-order derivative were satisfied by definition (see also Prichard and Marka 1974). There is a considerable body of literature on the use of higher-order approximations, for example cubic splines. Line segments, however, have the advantage of being conceptually simple. The effect of changing the slope or intercept of a straight line is easy to conceptualize, but it is more difficult to correlate changes in higher-order polynomial coefficients with changes in the appearance of the approximation to some waveform. However, last such one-to-one correspondence is essential in the feature-oriented voice synthesis. Cubic splines are also more complicated in that a change in anything \( a(t) \) will change all of the coefficients across the entire approximation since the slopes at each \( a(t) \).
3. Error Norms

The most common measure of error is the square of the vertical distance i.e., the distance parallel to the y-axis between a function and an approximation to it. Didia and Haral (1973, p. 332) discuss a variation of this, in which the error is the perpendicular distance from a point (x) of the original function to the approximating line. Didia and Haral (1977) present an algorithm for finding a line to a set of points using this error norm. Not according to Monty (1990), experience has shown that using this error criterion does not improve the results of approximations for the class of waveforms under discussion. Since it is not a suitable, a considerable increase in computation time, this variation has not been tested in the work to be presented here.

Before various methods for approximating functions can be examined, three error norms need to be defined.

3.1 Maximum Error

The maximum error \( E_m \) is given by

\[
E_m = \max_{x \in [a, b]} |y(x) - \hat{y}(x)|.
\]

(5)

3.2 Sum-of-Squared Error

Pavlidis (1973) also uses this to find the integral square error:

\[
E_s = \sum_{x \in [a, b]} (y(x) - \hat{y}(x))^2.
\]

(6)

3.3 Mean Squared Error

The mean squared error across each \( a \) is given by

\[
E_m = \frac{1}{N} \sum_{i=0}^{N-1} (y(i) - \hat{y}(i))^2.
\]

(7)

The \( E_s \) norm is a measure of error across long line segments than the \( E_m \) and \( E_s \) norms.

4. Algorithms for Line-Segment Approximation

There are two basic approaches to solving the problem of approximating using splines. In the first, the number of segments is specified in advance and the algorithm is required to minimize some measure of error. The number of points is changed in the other method until the measure of error lies in class as possible to, but still under, some predetermined threshold.

4.1 Minimizing Error: ADJUST

One method for minimizing error with a given number of line segments is presented by Pavlidis (1973). These two typographical errors occurred in Eq. (6) of Pavlidis’s error, (which should read \( p' = \sum_{i=0}^{N-1} a_i \)) and since \( y_i \) forms an integral part of the procedures discussed in the next section, Pavlidis’s algorithms will be discussed in some detail.

The algorithm, called ADJUST, in the rest of this paper, is given in Fig. 3. For each iteration, the coordinates of successive segments (the odd-numbered segments, then the even-numbered ones) are moved by some number \( M' \), always set to 1 in the work discussed in this paper. Larger values of \( M' \) could be specified for the first few iterations if the initial approximation was thought to be significantly different from the expected final one. If the error at the approximation using the trial break points is less than the original error, then the trial
breakpoint replaces the original breakpoint.

Pavlidis presents a proof to show that the algorithm will converge in a finite number of steps, and more importantly, that no cycling is possible. It must be emphasized that this algorithm is useful for finding local minima, not for a globally optimal solution.

Pavlidis states that the maximum error norm is to be preferred for most applications, although the sum of squared error and mean squared error norms have also been used successfully in the work presented here.

A note on the implementation of ADJUST: if both endpoints of some a_i are points from the function being approximated, then a_i, for example, only needs to be calculated using points \( r_{i-1} \) through \( r_{i+1} \), because the points \( r_i \) and \( r_{i+1} \) cannot contribute to the error. This represents a slight improvement over Pavlidis's algorithm, which points out this computational saving only for the beginning endpoint.

The results of applying ADJUST to two test cases are given in Fig. 4 and Table 1. The "worst" case (in terms of computational time) is given if the breakpoints for the original approximation are all gathered at one end of the function, in which case ADJUST must spread the points across the function in the process of finding the optimal approximation.
Fig. 4. Results of applying the algorithm ADJUST (given in Fig. 3) to the approximations of two functions, further data is given in Table 1. The y,... was used in all of the cases illustrated here.

(a) This test case consists of two diagonal lines, each corresponding to 34 units on the x-axis, separated by a horizontal line one unit in length. The original smooth diagonal lines have been modified by adding some random variations in the amplitude of which depends on the y-value of the original line. (b) An arbitrary initial placement of four line segments approximating our case (a) (c) Approximation (a) and case (d), with four line segments, after algorithm ADJUST has reached a solution. (d) As in (c), but with 11 line segments. Perhaps if the 3 discontinuities were distributed more evenly, the curve could be reduced further and the step on the left-hand side might be avoided. But ADJUST can write onto a locally optimal solution, which is not necessarily optimal globally. (e) One-half of a sine wave, again spread across 49 units of the x-axis and with noise added in (f) Approximation of 45. With four line segments, after ADJUST has reached a solution. (g) As in (f), with 11 line segments.
Table 1. Performance of the algorithm ADJUST for the test cases shown in Fig. 4

<table>
<thead>
<tr>
<th>Length in Fig. 4</th>
<th>Initial Error [a]</th>
<th>Final Error [a]</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>12.9</td>
<td>25.3</td>
<td>4</td>
</tr>
<tr>
<td>Height</td>
<td>43.1</td>
<td>0.055</td>
<td>11</td>
</tr>
<tr>
<td>Width</td>
<td>0.003</td>
<td>0.009</td>
<td>2</td>
</tr>
</tbody>
</table>

Note: Initial error refers to the error for the initial, arbitrary placement of the approximating line segments. Each iteration reduces the error. The final error is the error after the last iteration. In all cases, the final error was less than 0.001. The number of iterations is shown in column 4. All cases were tested and no error was found.

4.2 Initial Segmentation

These algorithms (and a variant of one of them) for approximating a function with line segments will be discussed. The first two represent solutions to the problem, discussed above, of finding an approximation such that the error does not exceed some threshold, with no a priori restrictions on the final number of segments.

One widely used method, in which each line segment provides the least mean squared error approximation to all of the function, is not considered here because of the restriction in Eq. (3). Another interesting approach (Stone 1966, Phillips 1968, Pavlidis 1971) will be mentioned here for the same reason.

4.2.1 Thresholding Sum-of-Squared Error

The sum-of-squared error norm was defined in Eq. (5). The method, as suggested by Pavlidis (1973), is quiet simple. The endpoints for the approximating segments are incremented until the sum-of-squared error exceeds some threshold. The endpoints then determined by the segments is established, and the process is repeated using the endpoint just found as the new initial point.

4.2.2 Split-and-Merge

Pavlidis and Harowitz (1974) developed a highly efficient algorithm, presented in Fig. 5. In the version to be discussed here, the error is kept each segment of the approximating function must not exceed some threshold. The initial line segments are split into smaller line segments until the threshold requirement is satisfied for the line segment of only two points. Neighboring line segments are then joined when possible by MERGE, that is, when joining them will not violate the same threshold condition. Finally, ADJUST is applied to the segmentation. The algorithm repeat until no changes are made to the breakpoints during an iteration.

In its original formulation, the split-and-merge algorithm returns to the SPLIT procedure ("want") in Fig. 5A after every iteration. But given the requirement of Eq. (3), SPLIT only needs to be invoked once (during the first pass) for the following reason. After SPLIT, the error of each segment is less than or equal to the threshold. MERGE is therefore can only result in segments with error less than or equal to the threshold. For Dy in ADJUST (cf. Fig. 3) to be accepted as a breakpoint, a and a, must be less than or equal to a and a, respectively. But when ADJUST is invoked after MERGE, a and a, must be less than or equal to the threshold. Thus MERGE and ADJUST can result in segments with an error greater than the threshold, so that there is no need to invoke SPLIT again.

4.2.3 "Case 2"

There is another version of the algorithm, called "Case 2" by Pavlidis and Harowitz (1974), in which the error across all of the segments must not exceed some threshold. For the maximum error q, both cases are identical. When using the sum-of-squared error norm, however, the sum-of-squared error across the entire function must be minimized. Modification to the split-and-merge algorithm is necessary. This variant has been tested but will not be discussed here for reasons to be summarized in Section 4.3.

4.2.4 Conclusion

There is yet another method, based on a model of computer vision (Snygg 1968, Shapiro 1973, 1979) in a study of human visual perception, Atwood and McCauley (1954)
found that the points on a curved line where its
section changes most rapidly were excised by test
subjects as endpoints for drawing an outline of the
curve. Conversely, he found that an illusory outline can
be satisfactorily reproduced by drawing straight
lines between points of high curvature, the cat-
drawing in his article has been widely replicated
[e.g., Duda and Hart 1973, p. 339].

Sees changes curvature at a point P along a digit-
tized function as the angle between RP and PQ,
where Q and P are points of the function a constant
number of samples (called M by Shira) away from
P. In this method, then, the curvature at each point
of the function to be approximated is calculated
(Fig. 6), and breakpoints are assigned as points of
high curvature. If RPQ is a straight line, \( \alpha = \beta \),
for a highly acute angle RPQ \( \alpha \) approaches \( \alpha P \).
It is important to choose an appropriate value for
M. Too small a value will result in distortions in
the approximation because features which are too
small will cause large variations in curvature and
thus be assigned breakpoints. If M is too large, sig-
significant points of high curvature will be missed (cf. Fig. 7). Shima suggests a thresholding scheme for assigning breakpoints according to curvature. This method undoubtedly works for the cases cited by Shima, in which fairly smooth lines change direction only occasionally. But as is shown in Fig. 7, it seems difficult to find a single threshold which provides a useful approximation for the kinds of functions under consideration here. This method has also been found to be sensitive to the absolute values used to represent IL, especially for small M. Another problem occurs because the curvature near a probable breakpoint often does not reach a peak, but rather stays at a plateau; this happens, for example, at t ~ 0.1 sec in Fig. 6(c). Obviously only one point needs to be assigned—but which one? I have spent considerable time and effort in an attempt to devise heuristics for selecting only appropriate peaks of curvature, with no notable success to date. Perhaps a varying curvature threshold could be applied, but this has not yet been explored.

4.3 Summary

Three different analyses using the split-and-merge algorithm are presented in Fig. 8. As one would expect, the approximation using a very small threshold captures many details, as the threshold is increased, some details are lost and the overall shape of the waveform is emphasized.

Similar analyses have been conducted using the "Close 2" form of the split-and-merge algorithm as well as the thresholding algorithm of Section 4.2.1. The mean squared error and maximum squared error norms have likewise been explored. In general, using any combination of these algorithms and error criteria it is possible to produce results similar to those shown in Fig. 8. Each combination admittedly reacts to changes in the threshold in a unique way.
Fig. 4. Results of the split-and-average algorithm in the form given in Fig. 3, using the sum of squared error norm given by Eq. (5). Recall that the maximum amplitude has been normalized to 1.0. (a) Threshold 0.001 yields 53 segments, some of which contain only two points of the original data. (b) Threshold 0.02, resulting in 22 segments, a few of which are still only two points long. (c) Twelve segments are found when the threshold is 0.10, the obvious segment now covers five points of the original data. When the threshold is raised to 0.1 (not shown), the five blips in the attack is missed completely. See Section 4.2 for further discussion.

way, but it seems impossible to reach any general conclusion about the superiority of any algorithm as error norm for the purposes of the work outlined in the introduction to this article. Rather, it has proven impossible to find a para-
digm for controlling the threshold for the one waveform shown here so that the projected perceptual salient features (e.g., blips in the attack) can be retained or removed without concomitant, sig-
nificant changes in the rest of the waveform. If the threshold is large, then the analysis delivers the overall waveform, with a small threshold, sup-
"
5. Syntactic Analysis

Hierarchical syntactic analysis would seem ideal for mediating between global and local considerations. In the rest of this paper, then, such an approach will be presented.

5.1 Introduction

The methods to be discussed draw extensively from the literature on pattern recognition through syntactic analysis, which is based on the similarities between the structures of patterns and formal languages. There are, of course, many other methods of pattern recognition, such as template matching, but they will not be discussed here.

In syntactic pattern recognition, a relatively small vocabulary of terminals, called primitives, is "parsed" according to the rules of a grammar to form higher-level nonterminals known as "subpatterns" and "structures." Characteristic features are further grouped into patterns (or objects). A successful parse is equated with "recognition" of the pattern. Fortunately, many of the problems involved in pattern recognition by computer can be ignored here. A considerable body of literature is devoted, for example, to the question of finding lines in digitized pictures.

5.2 Primitives

One major advantage of syntactic analysis lies in the fact that a relatively small vocabulary of primitives can be used for constructing a wide variety of patterns. The choice of primitives is thus an important issue. It would seem reasonable, for example, to require that the same primitives be used in the analysis of both amplitude and frequency functions even if it turned out that analysis of the two at higher levels followed different rules.

A generalized approach to primitive selection has not yet been found. For, however, gives the following two requirements to serve as guidelines:

(i) The primitives should serve as basic pattern elements to provide a compact but adequate description of the data in terms of the specified structural relations (i.e., the concatenation relation).

(ii) The primitives should be easily extracted or recognized by existing nonparametric methods, since they are considered to be simple and compact patterns and their structural information not important (Fu, 1979, p. 47).

Various kinds of primitives have been developed for different pattern-recognition applications. Fu gives many examples, such as Freeman's chain code and half-planes in the pattern space for representing arbitrary polygons. However, in light of the restrictions (i) and (ii), it seems reasonable to use very small line segments connecting points of the original data as primitives for syntactic analysis.

For reasons discussed by Moore (1976), the implementation of the phase vocoder used at CCRMA performs an averaging of the amplitude and frequency functions. At a sampling rate of, for example, 26.6 kHz, 12 samples of the original output (corresponding to 0.25 msec) might be averaged to form one output point. These averaged output points are perfectly suited to serve as breakpoints for line-segment primitives in syntactic analysis, and will be used as such in this paper.

5.3 Grammar

The choice of primitives having been made, the next step is to decide on a set of subpatterns and features, and to specify a grammar for parsing the primitives accordingly. Davis and Rosenfeld (1979) provide a grammar based on hierarchical relaxation
methods. In fact, a line-segment primitive is classified as having length $x$ or $2x$, and as being horizontal, sloped upward, or sloped downward. The relaxation algorithm parses the primitives into longest line segments at a first hierarchical level, at the second level, peaks and valleys are formed from the line segments of the first level, and the final level expresses the whole function as concatenation of valleys and peaks. These are productions on the grammar for containing not only two adjacent primitives, but also two primitives separated by another.

Pavlidis (1971) provides examples of a grammar which can be used to synthesize line segments of very short duration and substitute in their place a single straight line. Continuums for finding more complex features such as the so-called "hams" (up-horizontal-down) or "cups" (down-horizontal-up) are given as well.

Having examined a large number of amplitude and frequency functions, it seems reasonable to the author to speculate on higher hierarchical levels for synthetic analysis. The first level attempts to remove very small features which one would attribute to noise in the waveform, artifacts of the analysis procedure, and so on. The second reduces the waveform to its overall shape in terms of fairly long line segments, and the third classifies those linear segments into parts of a note. The analysis system, which incorporates elements of the two methods just reviewed, will be presented in detail.

The primitives have already been specified as being very short line segments. Associated with each primitive is its duration and its slope, which are used to classify the line as up, down, or horizontal. The only relational operator between primitives is, coincidentally, the concatenation mentioned in the quote from Po. Pattern recognition can be used to separate other operations such as above and below or to the right, which complete the analysis significantly but these will not be discussed here.

5.3.1 Level 1: lineSeg

The grammar for the first two hierarchical levels is presented in Fig. 9. There are two subdivisions of

1. In the first, successive macroLineSeg primitives will be called are combined into lineSeg as long as (1) the next macroLineSeg is within the thresholds 
$\text{ crunchy }$ and $\text{ crunchy }$, (defined in Fig. 9), and (2) the direction of the first macroLineSeg is lineSeg being formed is the same as the direction of the next macroLineSeg, and (3) including the next macroLineSeg will not cause the duration of the lineSeg being formed to exceed some threshold $\text{ crunchy }$.

After a new lineSeg has been found, its duration is calculated as the sum of the durations of the constituent macroLineSegs. The slope for the new lineSeg is likewise calculated using the values of $\text{ crunchy }$ and $\text{ crunchy }$ at its beginning and endpoints. (This is known as a synthesized attribute of $\text{ crunchy }$.)

A similar process takes place at the second hierarchical levels.

5.3.2 Level 2: macroLineSeg

It proved advisable to insert a second subdivision into this hierarchical level in order to avoid occasional irregularities at Level 3. This is accomplished by the introduction of macroLineSeg, which are formed of one or more lineSeg, all with the same direction (up, down, or horizontal).

5.3.3 Level 3: featureLineSeg

The second hierarchical level uses a syntax with productions and conditions corresponding exactly to those of Level 1. At this second level, macroLineSeg are combined into featureLineSeg, with thresholds chosen so that only the most striking elements of the function remain.

5.3.4 Level 4: Note

At the third level, featureLineSeg are assigned to the various parts of a note: attack, steady-state, and decay, as well as to any silence which might occur before the attack. Software has been written to search out the longest horizontal segment for which the amplitude (or an amplitude function) exceeds some silence threshold and the duration exceeds some maximum time threshold. Anything
Fig. 9. Gaussians for the first two levels of an archetypical generative model. The conditional form is based on a model used by Mcardle (1971). The complete hierarchy is shown in Fig. 11. At Level 3 the
primtives (microLexicons) are passed into lexicon and then the items are com-
bined when possible into macroLexicon. The same
symmetry, but with different
thresholds and transposing changes in moments-
ness, is used in Level 2 for
combined macroLexicon into archetypical lexicon. A less
typical version of the gen-
erative text is given in section 5.3
of the text.

**Vocabulary:**
\( V_r = \text{lexicon}, \text{macroLexicon}, \text{featureLexicons} \)
\( V_r = \text{microLexicon} \)

**Abbreviations:**
\( \text{bus}, \text{fr} \) beginning point of lexicon (featureLexicon in Level 2)
\( \text{bus}, \text{fr} \) beginning point of microLexicon
\( \text{bus}, \text{fr} \) beginning point of macroLexicon (in Level 2)
\( D_r \) direction of first microLexicon in lexicon
\( D_r \) direction of microLexicon
\( D_r \) direction of first microLexicon in macroLexicon
\( D_r \) direction of lexicon
\( D_r \) direction of microLexicon in macroLexicon

**Thresholds:**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma_{-1}, \gamma &gt; \text{YThreshold} )</td>
<td>( \text{YThresh} := \text{macroLexicon} )</td>
</tr>
<tr>
<td>( \gamma_{-1}, \gamma &gt; \text{YThreshold} )</td>
<td>( \text{YThresh} := \text{featureLexicon}, \text{macroLexicon} )</td>
</tr>
<tr>
<td>( \gamma_{-1}, \gamma &gt; \text{YThreshold} )</td>
<td>( \text{YThresh} := \text{featureLexicon}, \text{macroLexicon} )</td>
</tr>
</tbody>
</table>

**Thresholds:**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma_{-1}, \gamma &gt; \text{YThreshold} )</td>
<td>( \text{YThresh} := \text{macroLexicon}, \text{featureLexicon} )</td>
</tr>
<tr>
<td>( \gamma_{-1}, \gamma &gt; \text{YThreshold} )</td>
<td>( \text{YThresh} := \text{featureLexicon}, \text{macroLexicon} )</td>
</tr>
</tbody>
</table>

129
between the initial silence and the steady-state is classified as attack, everything after the steady-state is decay. If no steady-state is found, then all the foregoing is interpreted as possible grouped together following the initial silence to form the attack, and the remaining features (length) are assigned to the decay. It should be emphasized that the attack—steady-state—decay terminology is used strictly for convenience in labeling part of the computer analysis, and is not to be taken literally.

5.4 Analysis of Amplitude Functions

Figure 10 shows an analysis of the waveform of Fig. 1 at every stage of the hierarchical analysis. None of these plots represents the ultimate form of the output from this method of analysis, further processing is envisioned, as will be discussed. However, by adjusting the thresholds properly, data reduction at, say, the lineart level (Fig. 10a) can be achieved which will probably be useful in a wide variety of cases.

The model for the analysis of an amplitude waveform assumes that the note will consist of the parts discussed above. Thus (attack) steady-state—decay model has been widely used in commercial analog synthesizers. The fact that it appears here is not coincidental. The class of waveforms for which this software has been optimized is restricted to waveforms derived from a limited set of root tones lasting typically not-quarter of a second or longer.

Ultimately these programs will be used to analyze two or three such notes played in succession, either separated by silence or occurring in some way legato, portato, etc. In fact, the software has already been implemented to handle such groups of notes. Briefly, the suspect 0 is given channel of the phone vocoder, the file compared with an amplitude threshold. Notes are initially defined within the entire function and being separated by periods of silence. The hierarchical analysis is then performed on a note-by-note basis. Each period of silence is assigned to the note immediately following. The entire function is thus represented as a linked list of notes followed by an optional silence. Each note, in turn, includes an optional initial silence, attack, decay, and optional steady-state. Within each note, the line segments at a given hierarchical level are represented as a linked list of records. A line segment at one hierarchical level also has pointers to one or more line segments at the next lower hierarchical level which the line segment at the higher level encompasses.

5.5 Analysis of Frequency Functions

This preliminary division of a function into notes incorporates the notion of gliding originally formulated by Mason [1963] but used here as developed by Kelly [1971]. Planning is further applied to the analysis of the frequency functions produced by the phone vocoder, which is especially problematical at low amplitudes where the frequency trace varies widely [Fig. 13]. There is also the problem of pitch containment, which occasionally produces characteristic spikes in the frequency trace. Before the frequency traces are submitted to a system analysis, these spikes are removed from the regions of the frequency trace bounded by the attack—steady-state—decay parts of the notes in the amplitude function. These same portions of the frequency traces are then analyzed syntactically using the grammar already given in Fig. 9. Details of the assignment of featureLinear in the notes of a note vary slightly but the frequency functions but the basic approach is the same. Fig. 11 shows the results of a syntactic, hierarchical analysis of the frequency trace belonging to the same harmonicist as the note shown in Fig. 10. Obviously the thresholds are different for amplitude and frequency functions. The notes have no obvious characteristics that are disconcerting. The note is not recognizable as containing a frequency function. It would not be surprising if the higher-level frequency functions could eventually be simplified or modified somewhat.

An entire musical phrase is thus represented as a linked list of phone vocoder channel outputs. Each channel consists of a line amplitude and frequency function. Each function in turn points to a linked
Fig. 13. Synthetic hierarchical analysis of the amplitude waveform shown in Fig. 1 (a) time-
scale, resulting from analysis of a line in the form of a line segment with DurThresh = 0.05 sec, YThresh = 0.05, and
SYThresh = 0.1. The line segment is combined into a line segment by the new LineSeg with DurThresh = 0.01 sec, YThresh = 0.02, and SYThresh = 0.4.
(a) Analysis of a line segment line segment of a new line segment. The "direction" of the angle line from t = 0.1 sec to t = 0.75 sec is noted as being di-
agonal (pointing downward) and therefore is not classified as belonging to a
steady-state. There are 65 line segments in the analyzed segment of (a), 19 line segments in (b), and so
on.131
Fig 11: Spectro-hist
artificial analysis of the
frequency function of Fig
10b. The vertical line ex-
tending to the x-axis in
the right-hand side is an
artefact of the display sou-
source used for plotting. com-
pared to that of Fig. 10b at
lineage, resulting in an
analysis with Dct/Thresh
= 0.25 sec.; YThresh = 2.
and XThresh = 5
(YThresh and XThresh in
Hertz); the function of the
values have been
transformed using
with Dct/Thresh = 0.1 sec.;
YThresh = 10; and XThresh
= 30. These analysis of
function using
parts of a wave. The "de-
lay" slope slightly down-
ward because it is con-
structed to end at a point
which does at the same
time as the final point of
the note found in the am-
plitude function. As stated
in the text, however, use of
the terms delay, nurse,
vacation, and decay is merely
for the sake of com-
venience. The top-down
point allows such pre-
liminary analysis to be
revised accordingly below
arrived at a final set of
headphones for such a
function. There are 90 line
segments in the approx-
imation of (b). 40 line
segments in (b), and 14 in
tc.
5.4 Refinement: A Top-Down Pass

This "bottom-up" game is only the beginning of a projected system, shown in Fig. 13. Cite the attack, steady-state, and decay portions of the amplitude function have been found, they can be utilized in terms of planning as a guide for a "top-down" directed search for features to be included in the final set of breakpoints. The output would thus no longer be grouped according to attack, steady-state, or decay. Rather, it would consist of the breakpoints found and confirmed by the top-down analysis.
Fig. 13. Overview of the proposed analysis system in its final form modeled after a system used in psycho-acoustic research (Hartley, 1974, p. 5; Riemann, 1961, p. 165).

1. Select
2. Input
3. number of channels
4. Input
5. number of channels
6. Select
7. number of meters
8. Select
9. number of meters
10. Select
11. number of meters
12. Select
13. number of meters
14. Select
15. number of meters
16. Select
17. number of meters
18. Select
19. number of meters
20. Select
21. number of meters
22. Select
23. number of meters
24. Select
25. number of meters
26. Select
27. number of meters
28. Select
29. number of meters
30. Select
31. number of meters
32. Select
33. number of meters
34. Select
35. number of meters
36. Select
37. number of meters
38. Select
39. number of meters
40. Select
41. number of meters
42. Select
43. number of meters
44. Select
45. number of meters
46. Select
47. number of meters
48. Select
49. number of meters
50. Select
51. number of meters
52. Select
53. number of meters
54. Select
55. number of meters
56. Select
57. number of meters
58. Select
59. number of meters
60. Select
61. number of meters
62. Select
63. number of meters
64. Select
65. number of meters
66. Select
67. number of meters
68. Select
69. number of meters
70. Select
71. number of meters
72. Select
73. number of meters
74. Select
75. number of meters
76. Select
77. number of meters
78. Select
79. number of meters
80. Select
81. number of meters
82. Select
83. number of meters
84. Select
85. number of meters
86. Select
87. number of meters
88. Select
89. number of meters
90. Select
91. number of meters
92. Select
93. number of meters
94. Select
95. number of meters
96. Select
97. number of meters
98. Select
99. number of meters
100. Select
101. number of meters
102. Select
103. number of meters
104. Select
105. number of meters
106. Select
107. number of meters
108. Select
109. number of meters
110. Select
111. number of meters
112. Select
113. number of meters
114. Select
115. number of meters
116. Select
117. number of meters
118. Select
119. number of meters
120. Select
121. number of meters
122. Select
123. number of meters
124. Select
125. number of meters
126. Select
127. number of meters
128. Select
129. number of meters
130. Select
131. number of meters
132. Select
133. number of meters
134. Select
135. number of meters
136. Select
137. number of meters
138. Select
139. number of meters
140. Select
141. number of meters
142. Select
143. number of meters
144. Select
145. number of meters
146. Select
147. number of meters
148. Select
149. number of meters
150. Select
151. number of meters
152. Select
153. number of meters
154. Select

Frequency bands: (2) higher bands: large values; (3) higher bands: large values; (4) higher bands: large values; (5) higher bands: large values; (6) higher bands: large values; (7) higher bands: large values; (8) higher bands: large values; (9) higher bands: large values; (10) higher bands: large values; (11) higher bands: large values; (12) higher bands: large values; (13) higher bands: large values; (14) higher bands: large values; (15) higher bands: large values; (16) higher bands: large values; (17) higher bands: large values; (18) higher bands: large values; (19) higher bands: large values; (20) higher bands: large values; (21) higher bands: large values; (22) higher bands: large values; (23) higher bands: large values; (24) higher bands: large values; (25) higher bands: large values; (26) higher bands: large values; (27) higher bands: large values; (28) higher bands: large values; (29) higher bands: large values; (30) higher bands: large values; (31) higher bands: large values; (32) higher bands: large values; (33) higher bands: large values; (34) higher bands: large values; (35) higher bands: large values; (36) higher bands: large values; (37) higher bands: large values; (38) higher bands: large values; (39) higher bands: large values; (40) higher bands: large values; (41) higher bands: large values; (42) higher bands: large values; (43) higher bands: large values; (44) higher bands: large values; (45) higher bands: large values; (46) higher bands: large values; (47) higher bands: large values; (48) higher bands: large values; (49) higher bands: large values; (50) higher bands: large values; (51) higher bands: large values; (52) higher bands: large values; (53) higher bands: large values; (54) higher bands: large values; (55) higher bands: large values; (56) higher bands: large values; (57) higher bands: large values; (58) higher bands: large values; (59) higher bands: large values; (60) higher bands: large values; (61) higher bands: large values; (62) higher bands: large values; (63) higher bands: large values; (64) higher bands: large values; (65) higher bands: large values; (66) higher bands: large values; (67) higher bands: large values; (68) higher bands: large values; (69) higher bands: large values; (70) higher bands: large values; (71) higher bands: large values; (72) higher bands: large values; (73) higher bands: large values; (74) higher bands: large values; (75) higher bands: large values; (76) higher bands: large values; (77) higher bands: large values; (78) higher bands: large values; (79) higher bands: large values; (80) higher bands: large values; (81) higher bands: large values; (82) higher bands: large values; (83) higher bands: large values; (84) higher bands: large values; (85) higher bands: large values; (86) higher bands: large values; (87) higher bands: large values; (88) higher bands: large values; (89) higher bands: large values; (90) higher bands: large values; (91) higher bands: large values; (92) higher bands: large values; (93) higher bands: large values; (94) higher bands: large values; (95) higher bands: large values; (96) higher bands: large values; (97) higher bands: large values; (98) higher bands: large values; (99) higher bands: large values; (100) higher bands: large values; (101) higher bands: large values; (102) higher bands: large values; (103) higher bands: large values; (104) higher bands: large values; (105) higher bands: large values; (106) higher bands: large values; (107) higher bands: large values; (108) higher bands: large values; (109) higher bands: large values; (110) higher bands: large values; (111) higher bands: large values; (112) higher bands: large values; (113) higher bands: large values; (114) higher bands: large values; (115) higher bands: large values; (116) higher bands: large values; (117) higher bands: large values; (118) higher bands: large values; (119) higher bands: large values; (120) higher bands: large values; (121) higher bands: large values; (122) higher bands: large values; (123) higher bands: large values; (124) higher bands: large values; (125) higher bands: large values; (126) higher bands: large values; (127) higher bands: large values; (128) higher bands: large values; (129) higher bands: large values; (130) higher bands: large values; (131) higher bands: large values; (132) higher bands: large values; (133) higher bands: large values; (134) higher bands: large values; (135) higher bands: large values; (136) higher bands: large values; (137) higher bands: large values; (138) higher bands: large values; (139) higher bands: large values; (140) higher bands: large values; (141) higher bands: large values; (142) higher bands: large values; (143) higher bands: large values; (144) higher bands: large values; (145) higher bands: large values; (146) higher bands: large values; (147) higher bands: large values; (148) higher bands: large values; (149) higher bands: large values; (150) higher bands: large values; (151) higher bands: large values; (152) higher bands: large values; (153) higher bands: large values; (154) higher bands: large values;
Consider the dips in the attack portion of the amplitude waveform analyzed in Fig. 10. A dip can be represented as a characteristic succession of line segments, e.g., up-down-trip or up/kinematic-down-up) at the fast time level. Such a pattern can easily be found using spectral analysis. The notion of "attack" and "steady-state" can be refined similarly, for example, by "tripping" to extend each part as far as possible (Shir et al., 1973) or to include an initial "overshoot" at the beginning of the steady-state. After each function is refined in this manner, the entire family of functions can be analyzed to confirm the acceptability of various features for each sound (cf. Fig. 13). For example, a flag can be set to retain a dip in an amplitude function only if the dip occurs at the same place as, say, more than two harmonics. The initial segregation of the phase-locked-output output into notes can likewise be confirmed by comparing all of the functions. It might happen that a spurious "silencer" detected in some amplitude functions would result in an erroneous initial division of one note into two. Such an error can be detected at this stage. This also provides, at least partly, the justification for the sharpening process described in Section 4.
for requiring that all breakpoints in the approximation be identical in points of the original waveform (Fig. 10). If the approximation at the various harmonical levels would include other points, then a large amount of remultiplication would be necessary every time a breakpoint were modified (Fig. 10). In other words, it seems impossible at this stage to require that the analysis mean some of the low-level approximations, in the form of the original data points, at higher levels of the path. This requirement may have to be modified later, however, if it turns out that a significant reduction in the final number of segments can be achieved by doing so.

Other methods from pattern recognition will probably prove useful in this work. Perhaps syntactic analysis (Ra 1974) in the bottom-up process would reduce the amount of trial and error to be done later. The Hough transform (Veda and Har 1975, Tomlinson and Shapiro 1978) might also prove useful for detecting, for example, the four points defining a square (segment s, and m1, l > 1), which actually colinear and that the entire function between them could be represented as a single line.

6. Summary and Conclusion

Various algorithms from the literature on approximation theory and pattern recognition have been shown to be useful for approximating waveforms in digital signal synthesis. A syntactic analysis scheme has also been presented which will ultimately be extended to a generalized parser for amplitude and frequency functions.

This work represents one aspect of the growing application of artificial intelligence (AI) techniques to music synthesis problems. The system and data structure here could easily be extended to a system for instrument identification similar to the speech-identification system Hayashi, Itoh, and LeRoux 1975. Egan 1976. A scheme for automatic-transcription of music could also be devised which in turn could be used to drive a music-analysis system such as that developed by Tinney (1977). The syntactic analysis presented here will also prove useful for approximating other time-varying waveforms in a wide variety of musical applications.

7. Acknowledgments

Dexter Morrill (Collage University) provided the support and encouragement necessary for this project. In addition to assisting at every stage of this work, James A. Mount (CCSM) contributed considerably to the design and implementation of the syntactic analyzer. I would also like to express my appreciation to my teachers and colleagues for their time, patience, and many helpful suggestions: John Cheeseman, John Grey, and Julius Smith (CCSM), Barry Sorkin (Stanford AI Project), and Curtis Roads.

References


136
Proceedings of The 1980 International Computer Music Conference

Peristaltic Transmission, November 28, 1979, San Francisco, California.

1. Introduction

2. Theoretical Background

3. Experimental Methodology

4. Results

5. Discussion

6. Conclusion

Acknowledgments

References

Appendices

Biographical Sketches

Index