Differential PCM Representation of Acoustic Instruments and Wavetable Synthesis

Steven Trautmann
Software Laboratory/DCES Technology Center, Texas Instruments
e-mail: s-trautmann1@ti.com

Abstract
Sample based synthesis (including wavetable synthesis) of acoustic instruments usually relies on standard 16-bit pulse code modulation (PCM) for storage, playback and manipulations such as pitch-shift. However, storing the samples as differential pulse code modulation (DPCM) data has certain advantages. When shifting the pitch using linear interpolation, in most cases fewer operations are required when working with DPCM as opposed to PCM. Also the range of a DPCM signal is typically less than that of the equivalent PCM signal. This can improve coded representations of the signal.

1 Introduction
Normally, sampling based music synthesis uses PCM recordings of instruments which are played back at different speeds to produce different pitches. This is equivalent to advancing in the PCM data by some number of samples other than 1 to get the next output sample. Unless this number is an integer, some form of interpolation will be required to produce values in-between PCM samples. The ideal way to do this is to recreate the unique continuous band-limited function represented by the PCM data, allowing an output sample to be taken at any location. However, even good approximations to this ideal interpolation are too computationally expensive to be done in real-time. Luckily classical linear interpolation, when used between samples, offers output values which are adequate for many real-time applications.

Figure 1 illustrates this situation. The dotted line represents the ideal interpolating curve between the two PCM samples. Note that the difference between the two samples, dpcm[j], is smaller than the samples themselves, as is often the case. The point v(j,x) represents the desired value which is between the existing pcm[j] and pcm[j+1] values. The value of x represents the distance from pcm[j] and can be between 0 and 1.

2 Computational advantages
To linearly interpolate between two PCM samples, the function for a line between the two PCM data points is created and evaluated at the appropriate location. The formula is

\[ v(j, x) = (1 - x) \cdot pcm[j] + x \cdot pcm[j + 1] \]

where j is the integer part and x is the fractional part of the desired sample location v(j,x). This can be reformulated to

\[ v(j, x) = pcm[j] + x \cdot (pcm[j + 1] - pcm[j]) \]

= sum + x \cdot (dpcm[j])

Where dpcm[j] is the jth DPCM value which is equal to (pcm[j+1] - pcm[j]) and sum refers to the sum of all dpcm[i] with i from 0 to j-1, which equals pcm[j]. Thus if the values of sum and dpcm[j] are known, the computation consists of only a multiply and an addition, whereas the direct formula requires a subtraction, two multiplications and an addition.

Although less computation is required to do the linear interpolation, a running sum must be kept, which requires regular addition operations. In fact, the number of summation operations required is simply 2^x where x is the pitch shift in octaves. Thus, if the step size is less than 1, the addition to produce the next sum will not be required every time, so between
2 and 3 math operations are required for each output on average. If the step size is between 1 and 2, then at least one addition to produce the next sum will be required every time, but not more than two, so between 3 and 4 math operations are required for each output on average. Likewise if the step size is more than 2, more than 4 math operations are required for each output. Since traditional linear interpolation on PCM values requires 4 math operations, pitch shifts by more than an octave up, which require step sizes greater than 2, will require more computation with the differential coding. However in normal cases where the pitch is shifted down, resulting in a step size less than 1, or shifted up by a small amount, resulting in a step size between 1 and 2, some savings in computation will occur as shown in Figure 2.

Unfortunately this analysis doesn’t take into account the extra operations required for looking up PCM data, differential data, table lookup, and computations to keep track of the output location. The computations to keep track of output location are identical with both systems. However, these extra instructions have the effect of watering down the potential computational savings. For instance if 5 extra instructions are required to calculate the output location, and to store the output, then assuming no transposition, rather than a ratio of 3 to 4, or 75% of the original, the computation requires 8 to 9, or 88.8% of the original, as shown in Figure 3.

It should be noted that the above analysis is based on a count of fundamental instructions in the C language. Actual speed will depend on the final target, if instructions can be performed in parallel or if special operations can be used.

3 Reduction of range and compression

It’s also true that for most normal PCM recordings, the average magnitude of the differences between adjacent samples is much smaller than the average magnitude of the samples themselves, even though the range of potential differences is twice as large. For example, the values of 16-bit PCM data range from -32768 to 32767, so the maximum difference is 32767-(-32768) or 65535, and the minimum difference is –65535 doubling the range. However, most normal audio signals do not oscillate wildly between extreme PCM values. Instead there is a high correlation between one PCM value and the one after it. Figure 4 shows 7 samples taken from a single organ instrument.

Figure 4. 16-bit PCM pipe organ samples

Figure 5 shows on the same scale, the differences between the adjacent values in Fig. 2.

The original values can be reconstructed from the differences by adding the next difference to the current value to produce the next value, starting with
zero. Thus it is clear that the range has been reduced significantly with no loss of information. Note that most of the extreme values in figure 3 are actually artifacts occurring where one sample ends and the next sample begins and would not be used in reconstruction.

By storing only the difference between these two values, the range typically is reduced significantly. This means fewer potential values will be required to store the data with no loss, possibly allowing fewer bits to be used. More importantly it also means less error for any given number of bits less than 16 used to represent these differences, since the total range those bits need to represent will be smaller than the original PCM range. This can be further improved by using a lookup table to determine the value the bits represent rather than using a linear scale.

4 8-bit coding experiment

For convenience, 8-bits codes were used to represent the difference values. This allows 2:1 compression versus 16-bit PCM data, and allows exactly two symbols per 16-bit word, making handling the bit-stream (relatively) easier on 16-bit DSPs. Also, the 256-entry lookup table is not too large. The values in the lookup table were chosen based on analysis of some of the DPCM sample data to be encoded. To create this 256-entry lookup table, various methods were tried. A histogram was made and vector quantized, iterative error reduction was applied and even a genetic algorithm was tried.

4.1 Using the code table for encoding and decoding

Encoding of data is done simply by taking the difference between a current value (started at 0) and the next PCM value. The closest table entry to that actual difference is found, and that table entry is added to the current value, while the 8-bit table index for that entry is appended to the encoded data.

Decoding is done by the sound synthesis engine itself. Each 8-bit table index is used to find the corresponding difference, which is added to a running sum to produce the next sum. Each sum approximates the value of the original PCM data at that point. Linear interpolation between sums can be easily done by taking the next difference, and scaling it by the fractional part which describes the location between two sums, or data points. This is just added to the most recent sum to get the output value.

4.2 Looping

Special attention needs to be paid to the looping section of the data. Many instrument samples include a looping section where some part of the data is continually repeated in order to prolong the sound indefinitely until the note is turned off. With exact differential coding this is not a problem since the difference between the loop end and loop beginning is also exact so the same value will be reached each time. In essence the sum of all the differences in the loop including the difference between the end and beginning will be 0, similar to a closed curve. However, with inexact differential coding, this is not always the case. If the sum of the coded difference values in the loop do not sum to 0 due to error, the loop will drift by whatever the loop’s ‘error sum’ is each time. Thus if the sum of the differences in the loop is 1, the final PCM values outputted will increase by 1 each time through the loop, eventually saturating the output. To avoid this problem the loop’s initial PCM value (or the correct sum when the loop begins) is stored. Then when the loop end is reached, the running sum is reset to the loop’s initial value. This way the same loop is produced every time. Another way to avoid this problem is to somehow insure that the differences in the loop sum to 0, but simply storing the initial looping value seems easier and less prone to error.

4.3 Discussion

Since it is impossible to represent all possible differences in a 256-entry table, this compression method is lossy, and some error will result. Although a detailed mathematical analysis has not yet been performed, intuitively the noise should be fairly Gaussian, with some remaining correlation to the original signal. Experiments where the encoded/decoded signal is subtracted from the original signal seem to support this hypothesis. This noise degrades quality only slightly however, as it is generally very low compared to the signal (fairly high Signal to Noise ratio), so although the difference is audible, the music produced is not objectionable for many applications simple applications (such as toys, greeting cards etc), especially when the benefits are considered. However for some instruments the table doesn’t work as well as for others. Although the results of this experiment were unsatisfactory for some instruments regardless of the table used, other instruments responded very well to this treatment, producing very little extra noise. It will be further investigated if using different tables for different samples produces better results.

Reference