REAL-TIME IMPLEMENTATION TECHNIQUES FOR A CONTINUOUSLY VARIABLE DIGITAL DELAY IN MODELING MUSICAL INSTRUMENTS

Timo I. Laakso1, Vesa Valimäki2, Matti Karjalainen1, and Uuno K. Laine3

1Helsinki University of Technology 2Helsinki University of Technology
Laboratory of Signal Processing and Acoustics Laboratory
Computer Technology, SF-02150 Espoo, Finland
SF-02150 Espoo, Finland
3CARTES (Computer Arts Centre at Espoo), SF-02100 Espoo, Finland

ABSTRACT
Real-time implementation of variable delay using FIR and allpass filters is considered. Simple closed-form design formulas are given and implementation costs with signal processors are discussed.

I. INTRODUCTION
One of the fundamental advantages of digital signal processing techniques over traditional "analogue" methods is the easy implementation of a constant delay: one simply shifts a delay line of a desired length and lets the signal samples stay in a buffer memory for the given time. However, when a fractional or a continuously variable delay is required, more sophisticated methods must be used. This problem occurs, e.g., in the modeling of musical instruments where the length of a resonator tube or a string has to be tuned virtually continuously to generate a musical tone with a sufficiently accurate fundamental frequency [Kar63], [Las88], [Kar97].

There are two basic strategies to approximate a digital fractional delay (variable or fixed): either to use finite-impulse-response (FIR) filters of allpass-type recursive (IR) filters. In this contribution, we compare the use of FIR and allpass filters for the real-time implementation of a continuously variable delay element. Closed-form formulas for simple designs are given and implementation costs vs. performance are discussed, with code length estimates for TMS320C30 floating-point signal processor implementation.

II. APPROXIMATION USING FIR FILTERS
Let us consider the approximation of a noninteger delay $D$ by an FIR filter with the $z$-domain transfer function

$$H(z) = \sum_{n=0}^{N} h(n) z^{-n}$$

where $N$ is the filter order. Let us denote the fractional part of the desired overall delay $D$ by $r$. The first problem is how to choose the overall delay $D$ with respect to given $N$ and $r$. Recalling that the group delay of an $N$th-order symmetric filter is $N/2$, which is not integer for odd $N$, it is advisable to choose $D = r + N/2$ for even $N$ and $D = r + (N - 1)/2$ for odd $N$. So the largest values of the impulse response are at the center, which generally results in smallest approximation error. Using the classical Lagrange interpolation formula, the FIR filter coefficients are found by fitting the impulse response coefficients to pass through the given set of data values. 

$$H(z) \cong z^D, \quad D = 0 \ldots N$$

Hence, with integer values of the delay the approximation error is zero. The solution is (see, e.g., [Las88]):

$$h(n) = \frac{1}{\binom{N}{r}} \sum_{\substack{\ell=0 \ldots r \\ell \neq n}} D_{n}$$

Lagrange interpolation has several advantages: easy closed-form design, best possible response at low frequencies (due to maximally flat characteristics at zero frequency), and smooth and bounded magnitude response. For example, the maximum never exceeds unity, which is important in applications including feedback.

III. APPROXIMATION USING ALLPASS FILTERS
The $z$-domain transfer function of an $N$th-order allpass filter is of the form

$$H(z) = \frac{z^{-N}A(z^{-1})}{A(z)} = \frac{a_0 z^{N} + \cdots + a_{N-1} z + a_N}{1 + a_0 z + \cdots + a_{N-1} z}$$

where the numerator polynomial is a mirrored version of the denominator polynomial $A(z)$. Denoting the group delay of the denominator by $t_d(a)$, the group delay of the whole allpass filter is

$$t_d(a) = N + t_d(a)$$

(5)
Hence, the overall nominal delay is \( D = x + N \). Here we present a simple closed-form design method for the allpass filter approximating desired group delay. In 1971, Thiran proposed an analytic solution for an all-pole lowpass filter with a maximally flat group delay at zero frequency [Thi71]. Since the group delay of an allpass filter is twice that of the denominator, the solution can be expressed as

\[
\alpha_k = (-1)^i \binom{N}{k} \prod_{j=0}^{k} \frac{x + j}{x + j + k + 1} \quad k = 1, \ldots, N
\]

where \( x \) is the fractional part of the delay and \( \binom{N}{k} \) is a binomial coefficient. For example, the coefficient of the 1st-order allpass filter is \( \alpha_1 = -2(x + 1) \) and those of the 2nd-order filter are \( \alpha_2 = -2(x + 2) \) and \( \alpha_0 = x + 1 + 1(x + 1) \). Allpass filters have exactly unity magnitude response by definition and excellent group delay approximation even with low (2 to 5th-order) filter orders. The stability of the filter is always guaranteed for positive \( x \), as shown in [Thi71].

IV. TRANSIENT BEHAVIOUR

When tuning the delay continuously, there is the problem of severely disturbing glitches when a new integer delay is added or removed in the delay line. As discussed in [Smi71], these glitches can be practically eliminated in allpass filters. However, an even better performance can be obtained with FIR interpolators using special techniques, as will be shown in an extended version of this paper.

V. IMPLEMENTATION CONSIDERATIONS

We have estimated the costs of filter computations as well as coefficient updates for both FIR and allpass designs using formulas (3) and (6). The number of arithmetic operations as well as the estimated number of instructions for the TMS320C30 signal processor implementation are collected in Table I, assuming that only the changing parts of the coefficients are updated and the rest are stored to eliminate unnecessary computations. The allpass filter is assumed to be implemented with direct form I structure, involving \( 2N \) delays and \( 2N \) multiplications.

The implementation of the allpass filter is computationally slightly more expensive due to the required divisor in the coefficient update. However, usually much lower-order allpass filter is needed to meet the same delay specifications as that of an FIR filter. Furthermore, faster coefficient update can be obtained by polynomial approximation of the exact formulas, or by interpolating between stored sample values.

Table I. Comparison of Computational Costs of FIR and Allpass Implementations of Variable Delay

<table>
<thead>
<tr>
<th>Costs as a Function of Filter Order ( N )</th>
<th>FIR</th>
<th>Allpass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Computations: TMS320C30 Code</td>
<td>( N + 5 \text{ Instr.} )</td>
<td>( 2N / 2N \cdot 1 )</td>
</tr>
<tr>
<td>Coefficient Update: TMS320C30 Code</td>
<td>( N^2 + N / N + 1/2 )</td>
<td>( N^2 - 2N - 1 / 2N - 1 / 1 )</td>
</tr>
<tr>
<td>Memory: Coefficients / Delays</td>
<td>( N + 1 / N )</td>
<td>( N / 2N )</td>
</tr>
</tbody>
</table>

VI. REFERENCES


