A COMPUTATIONALLY EFFICIENT COEFFICIENT UPDATE TECHNIQUE FOR LAGRANGE FRACTIONAL DELAY FILTERS

Azadeh Haghparast and Vesa Välimäki

Helsinki University of Technology (TKK)
Department of Signal Processing and Acoustics
P.O. Box 3000, FI-02015 TKK, Espoo, Finland
E-mails: azadeh.haghparast@tkk.fi, vesa.valimaki@tkk.fi

ABSTRACT

A new algorithm for coefficient update of the Lagrange fractional delay FIR filter is proposed, which reduces the computational complexity dramatically. It is based on rearranging the polynomial terms of the Lagrange interpolation formula and computing the common product terms only once. Reordering the Lagrange interpolation formula yields two other methods for updating the coefficients, the direct and the division-based methods. The division-based method uses only one division per coefficient. The two latter methods reduce the computational load, although they are not as efficient as the new algorithm. Finally, the superiority of the direct form FIR implementation of the Lagrange fractional delay filter and the new coefficient update method over other existing methods is demonstrated in an audio signal processing application.

Index Terms—Acoustic signal processing, delay filters, FIR digital filters, interpolation, polynomials

1. INTRODUCTION

The Lagrange fractional delay (FD) filter is one type of variable fractional delay digital filter, which can be applied as a band-limited interpolator for sampling rate conversion and fine-tuning the sampling instances. These operations are used in a vast range of applications, such as synchronization in digital communications, music synthesis [1, 2], and sound reproduction based on wave field synthesis [3].

A direct form FIR filter structure can be used to implement the Lagrange interpolation. It is well known that the coefficients of an *M*th-order Lagrange FD filter are obtained from

$$h(n,D) = \prod_{\substack{k=0 \\ k\neq n}}^{M} \frac{D-k}{n-k}, \text{ for } n = 0,1,2,...,M \ , \qquad (1)$$

where D is a real number representing the delay from the beginning of the impulse response (n = 0) [1]. Being dependent on the fractional delay, the coefficients of the Lagrange FD filter have to be updated as the delay changes. Coefficient update becomes demanding if the fractional delay changes frequently.

Different methods have been developed to reduce the computational complexity of the Lagrange filtering process. One approach is to implement the Lagrange filter using the Farrow structure or modified Farrow structures [2, 4, 5, 6]. An alternative to the Farrow structure was presented in [7, 8], which is based on the discrete time Taylor series expansion.

In this paper, a new algorithm for updating the coefficients of the Lagrange FD FIR filter is presented. This algorithm automates the rearrangement of the polynomial terms and computes the common product terms of the coefficients once, thus reducing the computational load. Another efficient scheme for the calculation of the coefficients of the Lagrange interpolator was proposed in [9].

Two other techniques for coefficient update of the Lagrange FD filter are discussed in this paper. The simplest technique is to directly use the Lagrange interpolation formula with a slight manipulation. Also, a division-based method is introduced in which all the polynomial terms are multiplied first. Then, each coefficient is obtained by one division. The three mentioned methods are compared in terms of the number of additions and multiplications.

This paper is structured as follows. Section 2 describes different methods for updating the coefficients of the Lagrange FD FIR filter. Section 3 compares the presented methods. In Section 4, the advantage of the direct form FIR implementation of the Lagrange interpolation over other implementation methods is discussed in an example application. Finally, Section 5 concludes this paper.

2. COEFFICIENT UPDATE TECHNIQUES

In this section, three methods for updating the coefficients of the Lagrange FD FIR filter are presented.

2.1. Direct Method

A straightforward technique to calculate the coefficients of the Lagrange FD filter is to use the closed-form formula [2]. According to (1), the computation of each coefficient of an Mth-order Lagrange filter requires $2M$ additions, $M - 1$ multiplications, and M divisions. To compute all the coefficients, the computational load increases $M + 1$ times, resulting in $2M(M + 1)$ additions, $(M + 1)(M - 1)$ multiplications, and $M(M + 1)$ divisions.

A slight manipulation of the Lagrange interpolation formula leads to the reduction of the computational cost and eliminates the need for the division operation. Equation (1) can be reordered as

$$h(n, D) = C_n \prod_{\substack{k=0 \\ k \neq n}}^{M} (D - k), \text{ for } n = 0, 1, 2, ..., M, \qquad (2)$$

where

$$C_n = \prod_{\substack{k=0 \\ k \neq n}}^{M} \frac{1}{n - k}, \text{ for } n = 0, 1, 2, ..., M. \qquad (3)$$

Constants C_n are independent of the fractional delay D. Hence, they can be calculated once when the filter order is determined and are used during the whole process.

Additionally, the terms $(D - k)$ are used repeatedly in the computation of the coefficients. Therefore, it is reasonable to compute them once when the fractional delay changes instead of $M + 1$ times for every coefficient. This way the computational load reduces to M additions and $M(M + 1)$ multiplications. This is called the direct method for updating the coefficients.

2.2. Division-Based Method

Another approach to update the coefficients of the Lagrange FD filter is to use the division technique by representing the Lagrange formula as below:

$$h(n, D) = C_n \frac{\prod_{k=0}^{M} (D - k)}{D - n}, \text{ for } n = 0, 1, 2, ..., M. \qquad (4)$$

Using the above formula, it is sufficient to multiply all the terms $(D - k)$ once and, then, divide the common term by $(D - n)$ for every coefficient. In this method, M additions, $2M + 1$ multiplications, and $M + 1$ divisions are required to compute the coefficients.

Many division algorithms have been developed so far [10, 11, 12]. The taxonomy of division algorithms along with their impact on system design is presented in [11]. Recently, a fast fixed-point division algorithm was introduced in [12], which uses the Newton-Raphson method to perform division. In this method, a 16-bit fixed-point division is performed by 4 multiplications and 3 additions with a precision of 13 bits. This precision is acceptable for many applications using Lagrange FD filters. However, a higher accuracy can be obtained using a more precise division algorithm [11].

Using the Newton-Raphson method for the division algorithm, updating the coefficients of the Lagrange filter requires $4M + 1$ additions and $6M + 5$ multiplications.

Algorithm L (Updating the coefficients of the Lagrange FD filter). Given the order of the Lagrange FD filter M and the fractional delay D, this algorithm evaluates the coefficients of the Lagrange FD filter $h(n)$ according to equation (2).

L1. Initialization:
1. Set $j = 0$.
2. Compute the $x_i = D - i$ terms for $i = 0, 1, \ldots, M$.
3. If M is odd $L_j = M + 1$, else $L_j = M + 2$.
4. Set all the values of array a_j of length L_j equal to 1.
5. Fill $M + 1$ first elements of the array a_j by x_i s.
6. Rearrange the elements of array a_j in such a way that every other element is swapped with its following element in the array. (see Fig. 2)

L2. Iteration: Do the following steps while $L_j > 2$:
1. Let $L_{j+1} = L_j/2$.
2. If L_{j+1} is odd, $L_{j+1} = L_j + 1$.
3. Set all the values of the array a_{j+1} of length L_{j+1} equal to 1.
4. Multiply every other element of the array a_j by its next element and fill the array a_{j+1} by the results.
5. $j = j + 1$.

L3. Iteration: Do the following steps while $j > 0$:
1. $j = j - 1$.
2. $N_{\text{Iter}} = (L_0/2^j) - 1$.
3. Iteration: $k = 0$ to N_{Iter}: $a_j(k) = a_{j+1}([k/2])\, a_j(k)$

L4. Iteration: $k = 0$ to $M + 1$
1. 1. $h(k) = a_0(k)\, C(k)$

Fig. 1. New polynomial rearrangement algorithm.

2.3. New Polynomial Rearrangement Algorithm

The coefficients of a Lagrange FD filter of order M are Mth-order polynomials in D. Every two coefficients share $M - 1$ terms which are of the form $(D - i)$. Therefore, an efficient way to update the coefficients of the Lagrange FD filter is to calculate the common polynomial terms only once so that the overall computation is reduced. In Fig. 1, a new algorithm is introduced in which common terms are multiplied step by step to yield the coefficients.

A demonstration of the new polynomial rearrangement algorithm for a filter order $M = 6$ is shown in Fig. 2. The first step is to compute all the polynomial terms and put them in array a_0. The length of the array is selected to be $L_0 = 6 + 2 = 8$. The elements of the array are rearranged such that every other element is swapped with its following element. This is shown by crossed arrows in Fig. 2 (step L1). Note that the sixth element is set to 1. Indeed, at every stage of the algorithm any extra element of the array is set to 1.

Starting from the second step, the elements of the previous array are multiplied pair wise. The results constitute a new array whose elements are rearranged the same way as the first array in stage L1. For example, the result of the multiplication of the first two elements of the first array x_0x_1 will constitute the second element of the second array. This procedure continues until a two-element array remains (step L2). Therefore, this results in three arrays $a_0 = \{x_1, x_0, x_3, x_2, x_5, x_4, 1, x_6\}$, $a_1 = \{x_2x_3, x_0x_1, x_6, x_4x_5\}$, and $a_2 = \{ x_4x_5x_6, x_0x_1x_2x_3\}$.

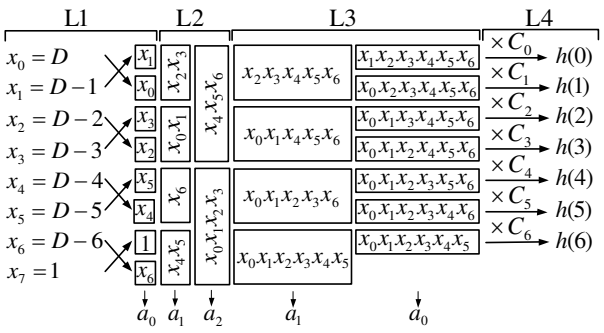

Fig 2. Block diagram of the novel polynomial rearrangement algorithm for filter order $M = 6$. Labels L1 to L4 refer to Fig. 1.

At this point, the algorithm takes a reverse path starting from the last array, i.e., a_2. Every element of the array is multiplied by two elements of the previous array. For instance, the first element of the array a_2 is multiplied by the first and second elements of the array a_1 and replaces them. Likewise, the second element of the array a_2 is multiplied by the third and fourth elements of the array a_1 and substitutes them (step L3). The array a_1 will, then, change to $a_1 = \{x_2x_3x_4x_5x_6x_7,\ x_0x_1x_4x_5x_6x_7,\ x_0x_1x_2x_3x_6x_7,\ x_0x_1x_2x_3x_4x_5\}$. This procedure is repeated for all arrays starting from a_2 until the array a_0, that is, the first array.

The last stage of the algorithm is to multiply the fixed coefficients C_n by the elements of array a_0 to yield the coefficients $h(n)$ of the Lagrange FD filter for the given fractional delay D (step L4). Using this algorithm, a 6th-order Lagrange filter requires 6 additions and 22 multiplications to update the coefficients. The number of additions and multiplications required in this algorithm is M and $4M - 2$, respectively, for an Mth-order filter.

3. COMPARISON OF METHODS

The computational costs of the methods mentioned above to update the coefficients of the Lagrange FD filter are compared in Fig. 3. Figure 3 (a) shows the number of multiplications required by each of the methods as a function of filter order. It can be seen clearly that as the order of the filter increases, the number of multiplications rockets up using the direct method. This is because the number of multiplications increases as $O(M^2)$. However, the order of increase for the division-based method and the polynomial rearrangement algorithm is $O(M)$, as the plots indicate. Figure 3 (a) also shows that the number of multiplications in the new rearrangement algorithm always remains below that of the direct and division methods.

Fig. 3 (b) indicates the number of additions for different filter orders. The number of additions in the direct form and the rearrangement algorithm are both equal to the filter order M. For the division-based method, the increase in the number of additions is faster than the two other cases.

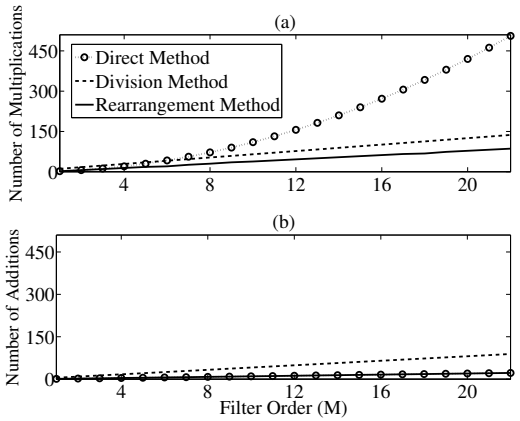

Fig. 3. Comparison of the computational cost of coefficient update.

The number of operations for three different filter orders M = 10, 20, and 50 using the direct method, the division-based method, and the rearrangement algorithm are compared in Table 1. As can be seen, the division-based method reduces the computational cost compared to the direct method. However, the proposed polynomial rearrangement algorithm requires the least number of multiplications and additions to update the coefficients of the Lagrange FD FIR filter compared with the other methods.

Table 1. Comparison of the computational cost for M = 10, 20, and 50. Number of multiplications (Mul), additions (Add), and total number of operations (Tot) are shown. The smallest total number of operations is underlined in each case.

	$M = 10$			$M = 20$			$M = 50$		
	Mul	Add	Tot	Mul	Add	Tot	Mul	Add	Tot
Direct Method	110	10	**120**	420	20	**440**	2550	50	**2600**
Division Method	65	41	**106**	125	81	**206**	305	201	**506**
Rearran-gement	38	10	**48**	78	20	**98**	198	50	**248**

4. DISCUSSION

Besides the direct form FIR filtering structure to implement the Lagrange FD filter, two general methods can be found in the literature. One of them is the Farrow structure [4] in which the Lagrange filter can be restructured to $M + 1$ filters with constant coefficients and M variable multipliers D. The computational cost to compute every output sample of the Farrow structure is $M^2 + M$ multiplications and M^2 additions. The Farrow structure can be modified such that its computational complexity is reduced [2, 5, 6]. One of the modifications was proposed in [5], which results in fewer multiplications, i.e., $M(M + 1)/2$ multiplications. The second filtering structure for the Lagrange FD filter was proposed in [7, 8]. This structure, which the authors of this paper call the Taylor structure, decreases the

computational complexity to $3M - 2$ additions and $3M - 1$ multiplications per output sample for an Mth-order Lagrange FD filter.

If a direct form FIR filtering structure is used, $M + 1$ multiplications and M additions should be added to the computational cost of updating the coefficients due to the filtering process. The supremacy of the direct form FIR filtering structure becomes apparent in the applications for which the fractional delay does not change very often, such as sound synthesis. While the Farrow or Taylor methods continuously update the filtering structure, in the direct form FIR implementation the coefficients are updated only when the fractional delay changes. For example, in audio applications the control rate is around 100 Hz. Therefore, with a sampling frequency of 48 kHz, the fractional delay may change, requiring coefficient update, every 480 samples. In Fig. 4, the average computational cost of the three mentioned methods over an interval of 10 ms with a sampling frequency of 48 kHz is shown as a function of filter order M. As can be seen, the direct form FIR implementation of Lagrange interpolation using the new rearrangement algorithm is more efficient than the other techniques.

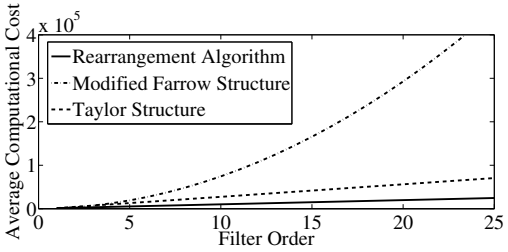

Fig. 4. Average computational cost over a time interval of 0.01 seconds with a sampling frequency of 48 kHz for different methods of implementing Lagrange FD filter.

Another advantage of the proposed algorithm is that it can be used for updating the coefficients of the truncated Lagrange FD filter, which was introduced recently [13]. The truncated Lagrange FD filter cannot be implemented using the Taylor structure.

5. CONCLUSION

A new algorithm for computationally efficient coefficient update of the Lagrange FD FIR filter was presented. The direct form FIR structure is the most efficient form for the implementation of the Lagrange FD filter in audio signal processing applications, such as music synthesis and wave-field synthesis, since the fractional delay does not change at every sampling interval. The proposed rearrangement algorithm requires a smaller number of operations compared to other coefficient update techniques, like the direct and the division-based methods.

6. ACKNOWLEDGMENT

This work has been supported by the Foundation of Technology (TES). Special thanks go to Dr. Balazs Bank and Mr. David Yeh for their useful comments on this work.

7. REFERENCES

[1] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay – Tools for fractional delay filter design," *IEEE Signal Process. Mag.,* vol. 13, no. 1, pp. 30-60, Jan. 1996.

[2] V. Välimäki, *Discrete-Time Modeling of Acoustic Tubes Using Fractional Delay Filters*, Doctoral thesis, Lab. of Acoustics and Audio Signal Processing, TKK, Espoo, Finland, Dec. 1995.

[3] A. Franck, A. Gräfe, T. Korn, and M. Strauβ, "Reproduction of moving sound sources by wave field synthesis: an analysis of artifacts," in *Proc. AES 32nd Int. Conf.*, pp. 188-196, Copenhagen, Denmark, Sept. 2007.

[4] C. W. Farrow, "A continuously variable digital delay element," in *Proc. IEEE Int. Symp. Circ. Sys.,* vol. 3, pp. 2641-2645, Espoo, Finland, Jun. 1988.

[5] J. Vesma and T. Saramäki, "Optimization and efficient implementation of FIR filters with adjustable fractional delay," in *Proc. IEEE Int. Symp. Circ. Sys.,* vol. 4, pp. 2256–2259, Hong Kong, Jun. 1997.

[6] T.-B. Deng, "Coefficient-symmetries for implementing arbitrary-order Lagrange-type variable fractional-delay digital filters," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4078-4090, Aug. 2007.

[7] S. Tassart and P. Depalle, "Analytical approximations of fractional delays: Lagrange interpolators and allpass filters," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.,* vol. 1, pp. 455-458, Apr. 1997.

[8] C. Candan, "An efficient filtering structure for Lagrange interpolation," *IEEE Signal Process. Letters,* vol. 14, no. 1, pp. 17–19, Jan. 2007.

[9] P. Murphy, A. Krukowski, and A. Tarczynski, "An efficient fractional sample delayer for digital beam steering," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.,* vol. 3, p. 2245-2248, Apr. 1997.

[10] D. A. Patterson and J. L. Hennessy, "Arithmetic for computers" in *Computer Organization & Design*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1997.

[11] S. F. Oberman and M. J. Flynn, "Division algorithms and implementations," *IEEE Trans. Computers*, vol. 46, no. 8, pp. 833–854, Aug. 1997.

[12] N. M. Nenadic and S. B. Mladenovic, "Fast division on fixed-point DSP processors using Newton-Raphson method," in *Proc. of the Int. Conf. on Computer as a Tool*, EUROCON 2005, vol. 1, pp. 705–708, Belgrade, Nov. 2005.

[13] V. Välimäki and A. Haghparast, "Fractional filter design based on truncated Lagrange interpolation," *IEEE Signal Process. Letters*, vol. 14, no. 11, pp. 816-819, Nov. 2007.