FACTA UNIVERSITATIS (NIŠ)
Series: Electronics and Energetics vol. 14, No. 3, December 2001, 399-409

A FAST ALGORITHM FOR LINEAR CONVOLUTION OF DISCRETE TIME SIGNALS

Zdenka Babić and Danilo P. Mandić

Abstract: A novel, computationally efficient algorithm for linear convolution is proposed. This algorithm uses an N point instead of the usual $(2N-1)$ point circular convolution to produce a linear convolution of two N point discrete time sequences. To achieve this, a scaling factor is introduced which enables the extraction of the term representing linear convolution from any algorithm that computes circular convolution. The proposed algorithm is just as accurate as standard linear convolution provided that the chosen circular convolution algorithm does not introduce round-off errors. The analysis is supported by simulation examples for several typical application cases.

Key words: Linear convolution, circular convolution, DSP algorithms, FFT.

1. Introduction

Convolution is at the very core of digital signal processing. Presently, there is an ever increasing number of applications that require convolution of some kind. Many DSP applications require a task being completed in the minimal time. To this cause, ongoing research into DSP hardware concentrates on providing faster real-time algorithms. The aim of this paper is to propose a novel method of linear convolution that significantly reduces the computational cost of traditional linear convolution. Many convolution algorithms have been developed to suit a particular application or available hardware. A convenient way to compute linear convolution of two N point

Manuscript received November 12, 2001. A version of this paper was presented at the fifth IEEE Conference on Telecomunication in Modern Satellite, Cable and Broadcasting Services, TELSIKS 2001, September 19-21, 2001, Niš, Serbia.

Z. Babic is with Faculty of Electrical Engineering University of Banjaluka Patre 5, 78000 Banja Luka, Bosnia and Herzegovina (e-mail: zdenka@etf-bl.rstel.net). D. Mandic is with School of Information Systems University of East Anglia Norwich, NR4 7TJ Norfolk, United Kingdom (e-mail: d.mandic@uea.ac.uk) .

sequences is to employ circular convolution, i.e. using properties of the discrete Fourier transform (DFT) [1], [2], [3] or number theoretic transform (NTT) [4], [5], [6]. All these methods require the use of $(2N-1)$ point circular convolution to compute N point linear convolution of discrete time sequences.

To improve the speed of this operation, other methods such as the right-angle circular convolution (RCC) have been proposed [7], [8] which makes it possible to compute a linear convolution of two discrete time sequences using an N point RCC instead of standard $(2N-1)$ point circular convolution. To compute RCC, the modified Fermat number transform (FNT) [7] was used, whereby some adjustments had to be made to split the N[A numbers from the computation of RCC into $(2N-1)$ numbers required by linear convolution calculation methods.

Another approach was presented in [9], where N point linear convolution was obtained by using a four N point FNT and three N point inverse FNT (IFNT), with an additional expense of $3N$ additions and $3N$ multiplications.

To calculate a linear convolution of two N point sequences, we propose a novel algorithm, which uses circular convolution in N points, instead of $(2N-1)$ points, as required by the standard algorithm. The proposed algorithm does not impose any limits on the method for calculation of circular convolution if the assumption of zero round-off errors is upheld. This algorithm assumes that signals to be convolved are represented with a relatively small number of bits, such as in the case after the A/D conversion. Processor word lengths, however, tend to be much longer than the data word after A/D conversion. Furthermore, in image processing, the pixel value is often represented by eight bits, whereas the convolution kernel and the edge detection operators use four or even two bits. Therefore there is a potential redundancy in accuracy of data processing with respect to data length. This redundancy can be used to develop efficient algorithms to speed up the calculation of linear convolution [10].

2. Basics of Fast Convolution

The traditional approach of fast signal convolution involves a multiplication in the frequency domain, due to the many established algorithms for fast computation of Fourier type transforms (FFT) [1], [2],[3]. Although these methods may seem mathematically involved, for long block sizes they often prove much faster for convolution than time domain methods. FFT algorithms produce a periodic output, with the same period as the input sequences. Convolution calculated this way is periodic and is called circular

convolution. For practical applications, however, we need the linear, not circular convolution of time domain sequences. Therefore, if we convolve aperiodic finite duration signals, we desire an aperiodic finite duration convolution of these signals. Methods to obtain a linear convolution from its circular counterparts are part of standard DSP courses and they rely on so called zero-filling [1], [2], [3], [4]. In general, the overlap that appears in the transform domain by using circular convolution to obtain the linear one can be avoided if the period N_{FFT} of the employed FFT is chosen to be

$$N_{FFT} \geq N_1 + N_2 - 1 \tag{1}$$

where N_1 and N_2 are the lengths of signals being convolved. In this case, input signals are filled with zeros up to a length N_{FFT} which is the smallest power of M for radix-M FFT methods, such that (1) holds. The inverse FFT returns a convolved sequence for which the length is $(N_1 + N_2 - 1)$. This is exactly the length of the standard linear convolution followed by $[N_{FFT} - (N_1 + N_2 - 1)]$ zeros in the returned signal. While standard linear convolution in the time domain requires approximately $N_1 N_2$ multiplications and additions, fast convolution methods require only two N_{FFT} point FFTs and one N_{FFT} point IFFT, which sums up to $3N_{FFT} \log_2 N_{FFT}$ operations for a radix-2 FFT algorithm. This means, that counting only multiplications, for $N_1 = N_2 = N = N_{FFT}/2$ the computational advantage of fast convolution, as compared to direct convolution, is

$$\frac{N^2}{3 \times 2 \log_2(2N) + 2N} \tag{2}$$

Fast convolution algorithms for practical applications employed in hardware may have execution speed advantage more than twice as much as this. For instance, for processing of real signals, there is no need to calculate the whole complex valued FFT, which speeds up the execution.

3. A Novel Algorithm for Linear Convolution by Means of Circular Convolution

To introduce a new algorithm let us first find the way to express circular convolution from linear convolution and vice versa. A circular convolution of two N point causal discrete time domain sequences $\boldsymbol{x} = \{x(0), x(1), \ldots, x(N-1)\}$ and $\boldsymbol{h} = \{h(0), h(1), \ldots, h(N-1)\}$ can be expressed as

$$y_c(n) = \sum_{k=0}^{N-1} x(\langle\, k \,\rangle_N) h(\langle\, n-k \,\rangle_N), \qquad n = 0, 1, \ldots, N-1 \tag{3}$$

where $\langle \cdot \rangle_N$ denotes the mod$N$ operator.

Based on the properties of the operator $\langle \cdot \rangle_N$

$$\begin{aligned} &\langle k \rangle_N = k, && 0 \le k \le N-1, \\ &\langle n-k \rangle_N = n-k, && 0 \le n-k \le N-1, \\ &\langle n-k \rangle_N = N+n-k, && n-k \le 0, \end{aligned} \tag{4}$$

to separate circular convolution (3) into two partial sums we write

$$\begin{aligned} y_c(n) &= \sum_{\substack{k=0 \\ 0 \le n-k \le N-1}}^{N-1} x(\langle k \rangle_N) h(\langle n-k \rangle_N) + \sum_{\substack{k=0 \\ n-k<0}}^{N-1} x(\langle k \rangle_N) h(\langle n-k \rangle_N) \\ &= \sum_{\substack{k=0 \\ n=0,\dots,N-1}}^{N-1} x(k) h(n-k) + \sum_{\substack{k=0 \\ n=0,\dots,N-2}}^{N-1} x(k) h(N+n-k) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad n = 0, 1, \dots, N-1 \end{aligned} \tag{5}$$

The lower index terms in sums (5) denote the range of values for which the sums are valid. If the index constraints are not upheld, assume zero value. The first term in (5) represents a linear convolution of sequences $\boldsymbol{x}$ and $\boldsymbol{h}$, for $n = 0, 1.\dots, N-1$, whereas the second term in (5) comprises the values of linear convolution between $\boldsymbol{x}$ and $\boldsymbol{h}$, for $n = N, N+1, \dots, 2N-2$. Technically, this gives an opportunity to calculate an N point circular convolution of two N point sequences from their linear convolution whereby an extra addition

$$\begin{aligned} &y_c(n) = y_l(n) + y_l(N+n), && n = 0, 1, \dots, N-2 \\ &y_c(N-1) = y_l(N-1) \end{aligned} \tag{6}$$

is necessary.

Notice that multipurpose and signal processors operate nowadays with a typically 64 bit processor word, whereas the data which come from an A/D converter are often represented by a 12-20 bit word. Separating the sums $y_l(n)$ and $y_l(N+n)$ from (6) (which are in fact linear convolutions) within the processor word, it would make it possible to calculate a linear convolution of N point sequences via a circular convolution in N points. Notice that the existing algorithms use a $(2N-1)$ point circular convolution for this purpose.

To separate $y_l(n)$ and $y_l(N+n)$ in (6), we introduce a scaling factor into input sequences as

$$\begin{aligned} x_1(n) &= x(n)s^n, \qquad n = 0, 1, \dots, N-1, \\ h_1(n) &= h(n)s^n, \qquad n = 0, 1, \dots, N-1. \end{aligned} \tag{7}$$

The circular convolution of $\boldsymbol{x}_1$ and $\boldsymbol{h}_1$ now becomes

$$y_{c1}(n) = s^n \sum_{\substack{k=0 \\ n=0,1,\dots,N-1}}^{N-1} x(k)h(n-k) + s^{N+n} \sum_{\substack{k=0 \\ n=0,1,\dots,N-2}}^{N-1} x(k)h(N+n-k), \qquad n = 0, 1, \dots, N-1 \tag{8}$$

and

$$\begin{aligned} y_{c1} &= s^n[y_l(n) + s^N y_l(N+n)], \quad n = 0, 1, \dots, N-2 \\ y_{c1} &= s^{N-1} y_l(N-1) \end{aligned} \tag{9}$$

Let us make a usual assumption that input sequences are bounded and can be expressed by a b-bit word. For N a power of two, representations of partial sums from (5) do not require more than $(2b+\log_2 N)$ bits. Therefore, $y_l(n)$ and $y_l(N+n)$ from (9) can be separated if s is chosen to be

$$s^N \geq 2^{2b+log_2 N} \tag{10}$$

In most practical applications $s = 2$ is a satisfactory choice.

As shown above, linear convolution of two N point sequences with bounded elements can be computed via the corresponding N point circular convolution instead of the standard fast convolution method, which requires a $(2N-1)$ point circular convolution. This result in the computational complexity for the proposed method being less than 50 % of that used in standard fast convolution. Any method for computation of circular convolution can be used to produce (8) as long as the final result is represented without round-off errors. For instance, methods that employ NTTs are suitable for this purpose. If a radix-2 FFT is employed for circular convolution, the advantage of the proposed method, in terms of computational complexity, as compared to direct linear convolution of two N point sequences, is approximately

$$\frac{N^2}{3 \times N \log_2 N + N} \tag{11}$$

Table 1 presents the reduction in computational complexity of the fast convolution and proposed method over the direct computation of convolution. Figure 1 shows the computational reduction plots of the proposed algorithm and the standard fast convolution algorithm. As seen, the proposed algorithm shows a consistent computational advantage over the standard algorithms. This is the case for any value of data length N (for small N, circular convolution does not have advantage over standard convolution - columns 1 and 2 in Table 1).

An insight in the equation (8) shows that for extracting the elements that represent linear convolution from an N point circular convolution, a choice of $s = 2$ does not require additional arithmetical operations, save for a linear shift.

Table 1. Showing the computational advantage of the fast convolution (2) and proposed method (11) for computation of linear convolution over the direct linear convolution in terms of computational speed.

N	16	32	64	128	256	512	1024	2048	4096
Fast convolution	0.50	0.84	1.45	2.56	4.57	8.26	15.06	27.68	51.20
Proposed method	1.23	2.00	3.37	5.82	10.24	18. 28	33.03	60.24	110.70

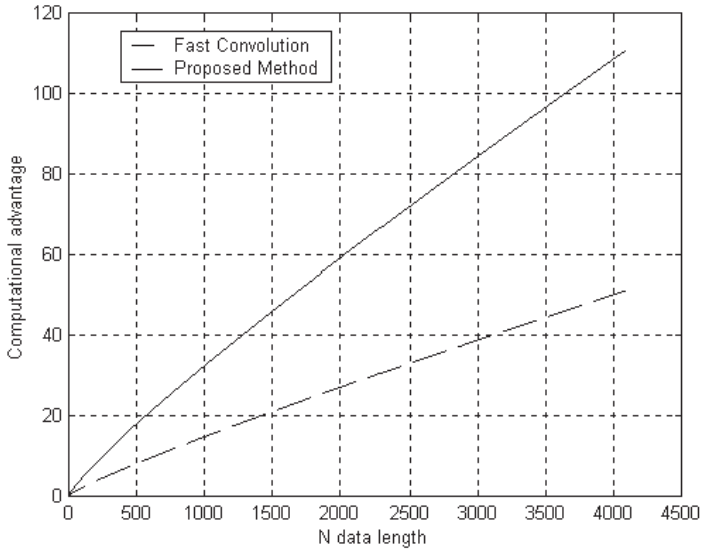

Fig. 1. Computational reduction plots.

4. Simulation Results

To support the analysis, simulations were undertaken for simple real sequences, and the results of the proposed and standard algorithms were compared with respect to accuracy and computational speed. The FFT and IFFT functions from MATLAB were used to compute N point circular convolutions. As expected, correct results were obtained if there were no round-off errors. If the size of the data and length of sequences were relatively small there were no scaling data to prevent overflow and N point circular convolution (8) of scaled sequences was computed exactly. In that case it was possible to separate two parts of linear convolution from (8) without errors (Example 1). If the data and sequence length increased, round-off errors occurred which then affected the first few samples of linear convolution (Example 2). In many DSP algorithms the first few samples of signals are discarded, thus reducing the sensitivity of the zero round-off error constraint. As b and N increased, the frequency and amplitude of round-off errors increased (Example 3).

Example 1

Convolution was performed for sequences with $N = 16$ and $b = 7$. The scaling factor was $s = 2$. Both the proposed and standard convolution algorithms produced the same results. Fig. 2 shows two input sequences (a) and (b), their linear convolution obtained with the standard fast convolution algorithm (circular convolution in $2N$ points via FFT) (c), and the linear convolution obtained with the proposed algorithm (N point circular convolution via FFT) (d). In this case, the proposed algorithm gave the correct result.

Example 2

Convolution was performed for sequences with $N = 32$ and $b = 7$. The scaling factor was $s = 2$. The round-off errors occurred in MATLAB if the FFT method was used for N point circular convolution of scaled sequences, because the magnitude of the DFT of a sequence tends to be significantly larger than the magnitude of the sequence itself. Some of the first elements of the result were incorrect. Fig. 3 shows two input sequences (a) and (b), their linear convolution obtained with the standard fast convolution algorithm (circular convolution in points via FFT) (c), and the linear convolution obtained with the proposed algorithm (N point circular convolution via FFT) (d) (the first few inaccurate samples of the resulting convolution

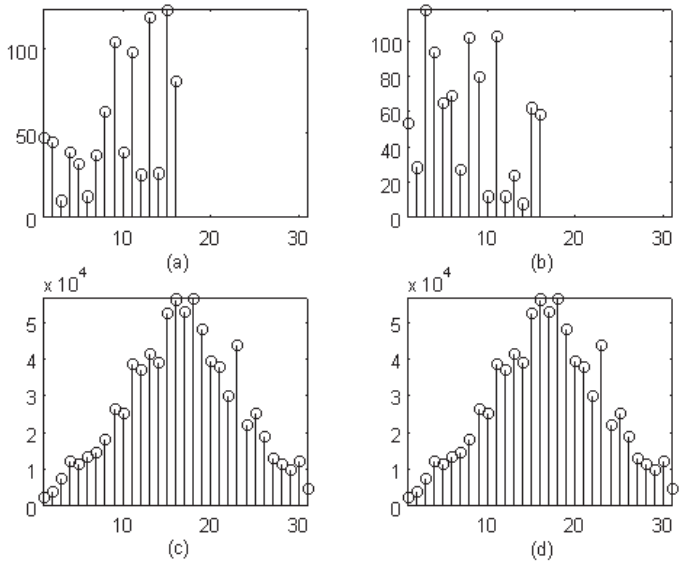

Fig. 2. Linear convolution of two 16-point 7-bit sequences.

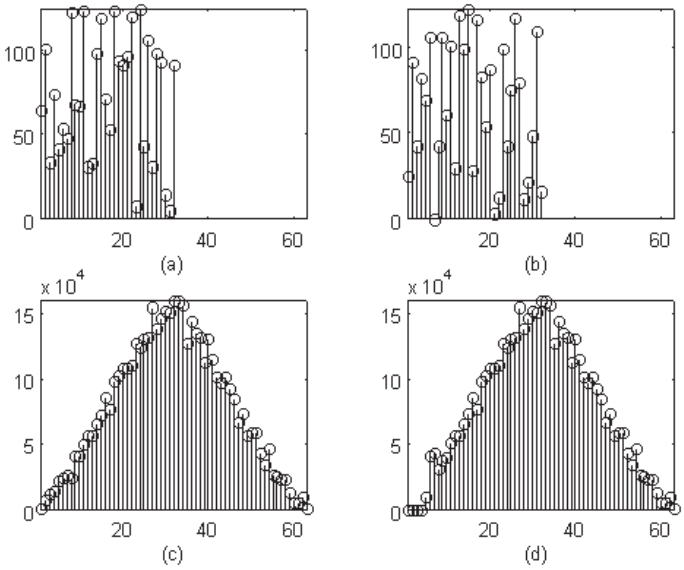

Fig. 3. Linear convolution of two 32-point 7-bit sequences.

sequence that are 10 % greater than the maximum value of the convolution are shown as zero elements).

Example 3

In this example, convolution was performed for sequences with $N = 64$ and $b = 7$. The scaling factor was . The round-off errors which occurred in MATLAB were greater than in the previous examples and therefore about a half of the elements of the convolution sequence were incorrect. Fig. 4 shows two input sequences (a) and (b), their linear convolution obtained with the standard fast convolution algorithm (circular convolution in points $2N$ via FFT) (c), and the linear convolution obtained with the proposed algorithm (N point circular convolution via FFT) (d) (the first few inaccurate samples of the resulting convolution sequence that are 10 % greater than the maximum value of the convolution are shown as zero elements).

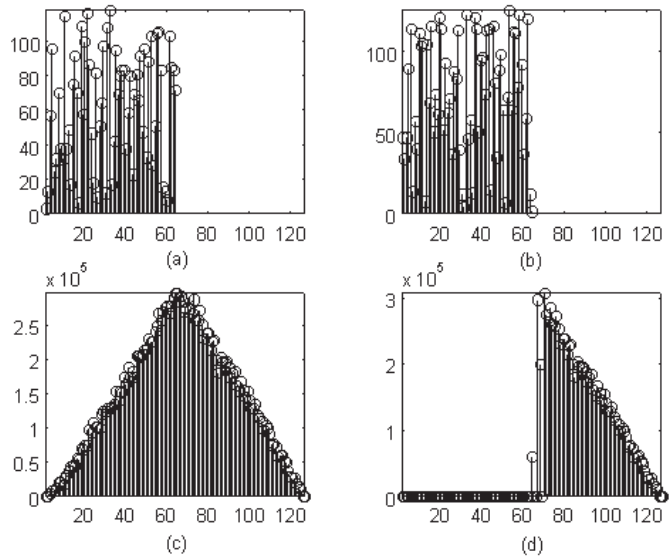

Fig. 4. Linear convolution of two 64-point 7-bit sequences.

5. Discussion

MATLAB was used to demonstrate accuracy of the proposed algorithm. For correct implementation of the proposed algorithm in MATLAB, or some other DSP tool which introduces round-off errors, the order of convolution

and word length have to be carefuly chosen to ensure the desired accuracy in circular convolution. Here, due to limitations of MATLAB, only simple, yet illustrative examples were chosen to support the analysis. In general, if the n-th bit of the circular convolution sequence is due to an error, that results in the bits $0, 1, \ldots, n$ of the linear convolution to be incorrect, as shown in the examples.

To make use of all the advantages of the proposed algorithm, a careful implementation of circular convolution, by exact methods in machine code or by special-purpose hardware is necessary. From (7) it can be seen that, if $s = 2$, for scaling of input sequences, only shift operations are needed to calculate the result in hardware, which is a great benefit. Extracting $(2N-1)$ elements of the linear convolution from the N point circular convolution (8) requires only $(2N-1)$ shift operations and $(2N-1)$ partitioning operators on a number (no subtraction if $s = 2$). Therefore, the proposed algorithm can be easily implemented either in hardware or software.

6. Conclusions

An algorithm for computing of N point linear convolution using N point circular convolution for real time signal processing has been proposed. It has been derived based upon redundancy of representation of real world input data in a signal processing system. This new algorithm has been shown to enable exact calculation of linear convolution and to introduce considerable improvement in computational complexity, provided that circular convolution does not introduce round-off errors. It is also applicable for a wide range of DSP methods, which introduce round-off errors but neglect the first few samples of linear convolution. Simulation examples support the analysis.

REFERENCES

1. L.R. Rabiner and B. Gold: *Theory and Application of Digital Signal Processing.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.
2. J.G. Proakis and D.G. Monolakis: *Digital Signal Processing.* Prentice-Hall International, Inc., 1996.
3. P.A. Lynn and W. Fuerst: *Introductory Digital Signal Processing with Computer Applications.* John Wiley & Sons, 1996.
4. J.H. McClellan and C.M. Rader: *Number Theory in Digital Signal Processing.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1979.
5. H. Krishna, B. Krishna, K.Y. Lin and J.D. Sun: *Computational Number theory and Digital Signal Processing.* CRC Press, Inc., 1994.
6. X. Ran and K.J. Ray Liu: *Fast algorithms for 2-D circular convolutions and Number theoretic transforms based on polynomial transforms over finite rings.* IEEE Trans. on Signal Processing, vol. 43, No. 3, 1995, pp. 569-578.

7. W. Li: *The modified Fermat number transform and its application.* In Proc. IEEE International Symposium on Communication and Systems 1990, pp. 2365-2368.

8. W. Li and A.M. Peterson: *FIR filtering by the modified Fermat number transform.* IEEE Trans. on Acoustics, Speech and Signal Processing, vol. 38, No. 9, 1990, pp. 1641-1645.

9. W. Shu and Y. Tianren: *Algorithm for linear convolution using number theoretic transform.* Electronic Letters, vol.24, No 5, 1988, pp. 249-250.

10. Z. Babic: *An efficient convolution algorithm over finite integer rings with arbitrary choosen level of parallelism.* In Proc. The 6th IEEE International Workshop on Intelligent Signal Processing and Communication Systems 1998, vol. II, pp. 807-811.

11. Z. Babic and D.P. Mandic: *A fast algorithm for linear convolution of discrete Time signals.* In Proc. The 5th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service 2001, vol. II, pp. 595-598.