Lietuvos matematikos rinkinys, Vol. 60, 2019
Proc. of the Lithuanian Mathematical Society, Ser. A
https://doi.org/10.15388/LMR.A.2019.14961

ISSN 0132-2818
eISSN 2335-898X
pages 33–38

Revised linear convolution

Rimantas Pupeikis

Institute of Data Science and Digital Technologies, Vilnius University
Akademijos st. 4, LT-08412 Vilnius, Lithuania
E-mail: Rimantas.Pupeikis@mii.vu.lt

Abstract. It is assumed that linear time-invariant (LTI) system input signal samples are updated by a sensor in real time. It is urgent for every new input sample or for small part of new samples to update an ordinary convolution as well. The idea is that well-known convolution sum algorithm, used to calculate output signal, should not be recalculated with every new input sample. It is necessary just to modify the algorithm, when the new input sample renew the set of previous samples. Approaches in time and frequency domains are analyzed. An example of computation of the convolution in time area is presented.

Keywords: LTI system, convolution sum, signal, impulse response.

1 Introduction

Convolution is a mathematical way of combining two signals to form a third one [4]. It is the single most important technique in digital signal processing [4]. Convolution is used in filtering, correlation, compression and in many other applications, as well in mathematics of many fields, such as probability and statistics [4, 2]. Although the concept of convolution is not new, the efficient computation of convolution is still an open topic because with data increasing, there appears demand for fast manipulation with large data [1]. In real-time applications, where a new input sample simultaneously replaces previous one, it is non-effective to recalculate convolution each time. It is needed to modify its algorithm in order to recalculate on-line only some products.

2 Statement of the problem

Suppose that $x(n)$ is an N point input signal of a LTI system running from 0 to $N-1$, and $h(n)$ is an M point impulse response (kernel) running from 0 to $M-1$. The output $y(n)$ of the system is an $N+M-1$ point signal running from 0 to $N+M-2$, given by the linear convolution of the form [4]

$$y(n) = x(n) \star h(n), \tag{1}$$

where $\star$ is the asterisk symbol of the convolution. As it is noted in [4], the convolution is important because it relates the three signals of interest: the input signal $x(n)$, the

output signal $y(n)$, and the impulse response $h(n)$. This equation allows each sample in the output signal $y(n)$ to be calculated independently of all other samples in the output signal.

Thus, in a convolution scheme we consider a discrete-time finite duration real-valued signal $x(n)$ of length N (i.e., $x(n) = 0$ for $n < 0$ and $n \geqslant N$) that is processed by LTI system having impulse response $h(n)$ of length M in order to obtain the signal $y(n)$ of length $N + M - 1$. If the input to a causal linear time-invariant system is a causal sequence (i.e., if $x(n) = 0$ for $n < 0$), the limits on the convolution sum formula are further restricted. In this case the two equivalent forms of the convolution formula become [4]

$$y(n) = \sum_{j=0}^{M-1} h(j)x(n-j) = \sum_{j=0}^{M-1} x(j)h(n-j). \tag{2}$$

To summarize, in total, the process of computing the convolution between $x(n)$ and $h(n)$ involves the following four steps [2]:

1. *Folding.* Fold $h(n)$ about $n = 0$ to obtain $h(-n)$;
2. *Shifting.* Shift $h(-n)$ by d to the right (left) if d is positive (negative), to obtain $h(d-n)$;
3. *Multiplication.* Multiply $x(n)$ by $h(d-n)$ to obtain the product sequence $x(n)h(d-n)$;
4. *Summation.* Sum all the values of the product sequence to obtain the value of the output at time $n = d$. Here d is an integer.

Assume that at any moment t_i the network of sensors is simultaneously evaluated the set of output samples $y(n)$ by processing the signal samples $x(n)$. At time moment t_{i+1} the new sample $x(n)_{new}$ enter processor memory and shift $x(n)$ samples by one sample to the right, extending the length of previous $x(n)$ samples till $N + 1$. The aim of the paper is to create the procedure in order to update the signal $y(n)$ in the presence of $x(0)_{new}$, $x(0)_{new+1}$, ..., appearing at time moments $t_{i+1}, t_{i+2}, \ldots$.

3 Two approaches for updating the convolution

It is not efficient to recalculate samples $y(n)$ anew using the ordinary convolution algorithm (1), if only one new signal sample $x(n)_{new}$ or even a small portion of new samples emerges replacing previous samples in the set of $x(n)$. Therefore, we will seek to solve of a linear convolution problem in time domain, applying the procedure as follows:

$$\begin{aligned}
y(0)_{new} &= h(0)x(0)_{new},\\
y(1)_{new} &= h(0)x(0) + h(1)x(0)_{new},\\
y(2)_{new} &= h(0)x(1) + h(1)x(0) + h(2)x(0)_{new},\\
&\vdots
\end{aligned}$$

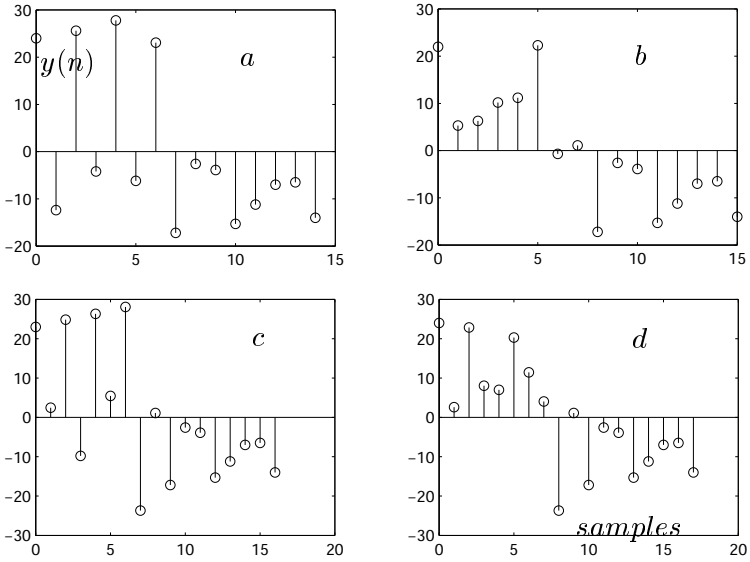

Fig. 1. Example results: (a) $y(n)$, (b) $y(n)_{new}$, (c) $y(n)_{new1}$, and (d) $y(n)_{new2}$ for $x(n)_{new2} = 24$.

$$
\begin{aligned}
&y(M-1)_{new} = h(0)x(N) + h(1)x(N-1) + \cdots + h(M-1)x(N+1-M),\\
&y(M)_{new} = y(M-1),\\
&y(M+1)_{new} = y(M), \ldots,\\
&y(N+M)_{new} = y(N+M-1)
\end{aligned}
$$

if the new sample $x(n)_{new}$ is apppearing at a time moment t_{i+1}, and

$$
\begin{aligned}
&y(0)_{new1} = h(0)x(0)_{new1},\\
&y(1)_{new1} = h(0)x(0)_{new} + h(1)x(0)_{new1},\\
&y(2)_{new1} = h(0)x(1) + h(1)x(0)_{new} + h(2)x(0)_{new1},\\
&\qquad\vdots\\
&y(M-1)_{new1} = h(0)x(N+1) + h(1)x(N) + \cdots + h(M-1)x(N+2-M),\\
&y(M)_{new1} = y(M-1)_{new},\\
&y(M+1)_{new1} = y(M)_{new}, \ldots,\\
&y(N+M+1)_{new1} = y(N+M)_{new}
\end{aligned}
$$

if the sample $x(n)_{new1}$ is entering the processor memory at t_{i+2}.

Let us assume that the LTI system's discrete-time input samples are: $x(n) = \{24, 8, 12, 16, 20, 6, 10, 14\}$. By inspection, the total number of the input is $N = 8$. Assume that the kernel $h(n) = \{1, -.85, .85, -.7, .7, -.25, .25, -1\}$. It has $M = N$. The output $y(n)$ samples are obtained by the convolution formula as follows: $y(n) = \{24, -12.4, 25.6, -4.2, 27.8, -6.2, 23.1, -17.2, -2.6, -3.9, -15.3, -11.2, -7, -6.5, -14\}$ (see, Fig. 1a). There appeares $x(0)_{new} = 22$. Then input samples are: $x(n) = \{22, 24, 8, 12, 16, 20, 6, 10, 14\}$. The length of $x(n)$ is 9, while the length of $h(n)$ left the same. After $y(n)$ recalculation we have: $y(n)_{new} = \{22, 5.3, 6.3, 10.2, 11.2, 22.3, -0.7,$

$1.1, -17.2, -2.6, -3.9, -15.3, -11.2, -7, -6.5, -14\}$. Note that there exist equalities as follows: $y(M+1)_{new} = y(M)$, $y(M+2)_{new} = y(M+1)$, $y(M+3)_{new} = y(M+2)$, $y(M+4)_{new} = y(M+3)$, $y(M+5)_{new} = y(M+4)$, $y(M+6)_{new} = y(M+5)$, $y(M+7)_{new} = y(M+6)$, $y(M+8)_{new} = y(M+7)$ (Fig. 1b). Thus, there is not necessary to calculate $y(n)_{new}$ values for n more than M. It is needed only to rewrite the values of previous $y(n)$ from M. Now, after entering $x(0)_{new1} = 23$ and $x(n) = \{23, 22, 24, 8, 12, 16, 20, 6, 10, 14\}$, by given above procedure we obtain $y(n)_{new1}$: $\{23, 2.45, 24.85, -9.8, 26.3, 5.45, 28.05, -23.7, 1.1, -17.2, -2.6, -3.9, -15.3, -11.2, -7, -6.5, -14\}$. The $y(n)_{new1}$ values that begin from $M+1$ are determined without calculation, i.e., we equate $y(M+1)_{new1} = y(M)_{new}$, $y(M+2)_{new1} = y(M+1)_{new}$, $\ldots$, $y(M+M+1)_{new1} = y(N+M)_{new}$ (Fig. 1c). The continuation of results is shown in Fig. 1d for $x(n)_{new2} = 24$. There exist, nearby, the other approach, named the fast convolution [2], that is based on the frequency samples (f.s.), used to solve this problem. The idea is that fast Fourier transform (FFT) algorithm, used to calculate output f.s., should not be recalculated with every new input sample if it is possible. It is needed just to modify the convolution algorithm, when the new input sample replaces the previous one. In a convolution scheme we consider a discrete-time finite duration real-valued signal $x(n)$ of length N that has the Fourier transform

$$X(\omega) = \sum_{n=0}^{N-1} x(n) e^{-j\omega n}, \quad \forall\, \omega \in \overline{0, 2\pi}, \tag{3}$$

where j is the imaginary unit. When we sample $X(\omega)$ at equally spaced frequencies $\omega_k = 2\pi k/N$, $\forall\, k \in \overline{0, N-1}$, the resultant samples are [2]:

$$X(k) \equiv X\left(\frac{2\pi k}{N}\right) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad \forall\, k \in \overline{0, N-1}. \tag{4}$$

The relation in (4) is called DFT of $x(n)$. It is used for transforming the sequence $x(n)$ into f.s. $X(k)$ of length N. The DFT of (1) is known as a fast convolution [2]:

$$Y(k) \equiv Y\left(\frac{2\pi k}{N}\right) = X\left(\frac{2\pi k}{N}\right) H\left(\frac{2\pi k}{N}\right), \quad \forall\, k \in \overline{0, N-1}, \tag{5}$$

where

$$H(k) \equiv H\left(\frac{2\pi k}{N}\right), \qquad X(k) \equiv X\left(\frac{2\pi k}{N}\right)$$

are DFTs of $h(n)$ and $x(n)$, respectively, N is the total number of samples of the basic real-valued signals $x(n)$, $y(n)$ and $h(n)$, $\forall\, n \in \overline{0, N-1}$ under consideration.

It is used for transforming the sequence $x(n)$ into f.s. $X(k)$ of length N. At any time moment t a signal $y(n)$ can be recovered from f.s. $Y(k)$, $\forall\, k \in \overline{0, N-1}$ by the IDFT (inverse DFT) [4, 2]:

$$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y\left(\frac{2\pi k}{N}\right) e^{j2\pi kn/N}, \quad \forall\, n \in \overline{0, N-1}. \tag{6}$$

At t_i time moment in sensor network some new sample $x(0)_{new}$ of signal $x(n)$ emerges and replaces the previous $x(0)$ by shifting the whole set of samples. In such a case

(4) can be rewritten as follows

$$X(k) \equiv X\left(\frac{2\pi k}{N}\right)_{new} = x(0)_{new} + \sum_{n=1}^{N} \tilde{x}(n) e^{-j2\pi kn/N}, \quad \forall\, k \in \overline{0, N-1}. \tag{7}$$

Here $\tilde{x}(1) = x(0), \tilde{x}(2) = x(1), \ldots, \tilde{x}(N) = x(N-1)$ are shifted previous samples of $x(n)$. The fast convolution for shifted input $x(n)$ is

$$Y(k) \equiv Y\left(\frac{2\pi k}{N}\right)_{new} = X\left(\frac{2\pi k}{N}\right)_{new} H\left(\frac{2\pi k}{N}\right), \quad \forall\, k \in \overline{0, N-1}. \tag{8}$$

In such a case, (6) can be rewritten as follows

$$y_{t_i}(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y\left(\frac{2\pi k}{N}\right)_{new} e^{j2\pi kn/N}, \quad \forall\, n \in \overline{0, N-1}. \tag{9}$$

Thus, in order to obtain the updated output $y_{t_i}(n)$ it is necessary to fulfill all restricted calculations once more. It should be emphasized that such a way, as compared with the technique given above, is non-effective one.

4 Conclusions

The linear convolution (2) have been proposed to determine by simple above presented procedure if one new signal sample or new small portion of samples emerge in the given period N of a realization $x(n)$ by shifting previous samples. The number of operations for their speedy calculation is essentially reduced because the procedure does not require to repeat calculations for $y(n)_{new}$ values that start from $M+1$ and finish at $N+M+1+j$ (Fig. 1). In such a case, it is necessary only to use sequential previous $y(n)$ values that start from $M+j-1$ and finish at $N+M+j-2$. Here $j = 1, 2, \ldots$. The algorithm proposed here could be effective in real-time applications for very large N. The fast convolution approach, based on the formula (5) is not effective because it requires the recalculation of $y(n)$ each time, when $x(n)_{new_j}$ enters the memory of processor. On the other hand, the fast convolution could be effective, when new current sample or small set of new samples replace the previous samples in $x(n)$ realization without its shift [3].

References

[1] K. Pavel and D. Svoboda. *Algorithms for efficient computation of convolution*. Available from Internet: http://dx.doi.org/10.5772/51942.

[2] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing, Principles, Algorithms, and Applications*. Prentice Hall, New Jersey, 2008.

[3] R. Pupeikis. Revised fast convolution. *Liet. matem. rink. Proc. LMS, Ser. A*, **57**:97–102, 2016. Doi:10.15388/LMR.A.2016.18.

[4] S.W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Second edition. California Technical Publishing, P.O. Box 502407, San Diego, 1999. Available at the Website: DSPguide.com.

REZIUMĖ

Patikslinta tiesinė sąsūka

R. Pupeikis

Tariama, kad taikant diskrečiąją tiesinę sąsūką, įėjimo signalo $x(n)$ atskaitų apdorojimui skaitmeniškai, kai kurios jo atskaitos esti jutiklių, veikiančių realiu laiku, pakeičiamos naujomis atskaitomis, perstūmiant visą skaičių seką. Būtina, kiekvienai naujai atsiunčiamai atskaitai skaičiuoti naują sistemos išėjimą pagal sąsūkos išraišką (1). Siūloma patikslinto išėjimo $y(n)$ reikšmes nuo $(M+1)$-osios reikšmės neperskaičiuoti naujai, o jas gauti perslinkus per vieną atskaitą ankstesnės $y(n)$ sekos stebėjimus. Taip taupomas laikas, skirtas $y(n)$ atskaitoms skaičiuoti. Straipsnyje taip pat analizuojamas greitosios sąsūkos metodas šiam atskaitų gavimo būdui. Parodyta, kad lyginant jį su darbe pasiūlytu algoritmu išaiškėja greitosios sąsūkos metodo neefektyvumas. Pateikti eksperimento rezultatai (1 pav.).

Raktiniai žodžiai: LTI sistema, kompozicijos suma, signalas, impulsine reakcija.