Recursive updating of linear convolution

Rimantas Pupeikis
Institute of Data Science and Digital Technologies
of Faculty of Mathematics and Informatics of Vilnius University,
Akademijos 4, LT-2600, Vilnius, Lithuania
email:rimantas.pupeikis@mif.vu.lt

January 16, 2020

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 from the begining with every new input sample. It is necessary just to modify the algorithm, when the new input sample renew the set of previous samples. The recursive algorithm is worked out for such a purpose. Example of recursive computation of the convolution is presented.

Keywords: linear convolution; signal; online procedure; discrete-time.

1 Introduction

The efficient computation of convolution is still an open problem [1]. In real-time applications [2], where each time a new input sample appear, is non-effective to recalculate a linear convolution, repeating the whole proce-

dure from the begining. It is necessary here to meet the requirements for the fast manipulation with increasing real data in time [3]. This problem could be solved by working out the original online procedure which does not require to process the whole set of data each time [4]. Therefore, it is needed to modify convolution algorithm in order to recalculate only some products recursively. To the best of the author's knowledge, based on the view of the recent scientific production there exists, any creation except this one, in which the convolution sum could be updated, reacting to the new observation samples coming in.

The text begins with an introduction. In Section II the statement of the problem is presented. Section III considers the adaptation of the linear convolution to the presence of new data. Section IV includes example, results and discussion. Section V presents conclusion.

2 Problem statement

Suppose that $x(n) \equiv x(nT_s)$ is an N point input signal of a LTI causal system running from 0 to N$-$1, and $h(n) \equiv h(nT_s)$ is an M point kernel (impulse response) running from 0 to M$-$1, where T_s is a sampling period. The output $y(n) \equiv y(nT_s)$ 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 [1]

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

For simplicity in (1) and elsewhere else it is assumed that T_s=1.

As it is noted in [1], 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. It is typical offline $y(n)$ calculation procedure, firstly, until time moment t_{i+1} a scope of measurement samples $h(0), h(1), ..., h(M-1)$, as well as $x(0), x(1), ..., x(N-1)$ is obtained and, secondly, all these samples are processed at the same time [2]. 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 $n \geq N$) 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 are valid [1]. To summarize, 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) integer, 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 $n = d$.

Assume that at any moment t_i the network of sensors is evaluated the set of output samples $y(n)$ by a batch processing of available samples of signal $x(n)$ and $h(n)$. At a time moment t_{i+1} a new sample $x(N)$ enters processor's input, extending the length of previous sequence $x(n)$.

The objective of the paper is to update the linear convolution online by appearing new sample $x(N)$, as well as samples $x(N+1)$, ..., $x(K-1)$,

$x(K)$, observed at sequential time moments $t_{i+1}, t_{i+2}, ..., t_{i+k-1}, t_{i+k}$, without repeating the whole calculations from the begining.

3 Convolution recursions

It is not efficient to recalculate samples $y(n)$ anew using the ordinary convolution algorithm (1), if only one new signal sample $x(N)$ or even a small portion of new samples $x(N)$, $x(N+1)$, ... , $x(N+m)$ emerges extending the length of previous set of $x(n)$. For simplicity, we assume that initially M=N. At time moment t_i there appears the sample $x(N)$. We will update samples $y(N, t_{i-1})$, $y(N+1, t_{i-1})$, $\cdots$, $y(N+M-2, t_{i-1})$, $y(N+M-1, t_{i-1})$ and calculate a new sample $y(N+M, t_i)$ by the adaptive algorithm. In such a case, terms are as follows:

$y(0, t_i) \equiv y(0, t_{i-1})$, $y(1, t_i) \equiv y(1, t_{i-1})$, $\cdots$, $y(N-1, t_i) \equiv y(N-1, t_{i-1})$, as

$y(N, t_i) = y(N, t_{i-1}) + h(0)x(N)$,

$y(N+1, t_i) = y(N+1, t_{i-1}) + h(1)x(N)$,

$\vdots$

$y(N+M-1, t_i) = y(N+M-1, t_{i-1}) + h(M-2)x(N)$

$y(N+M, t_i) = h(M-1)x(N)$

Then, at a time moment t_{i+1} there appeares $x(N+1)$ and expressions obtain the following form:

$y(0, t_{i+1}) \equiv y(0, t_i)$, $y(1, t_{i+1}) \equiv y(1, t_i)$, $\cdots$, $y(N, t_{i+1}) \equiv y(N, t_i)$,

as $y(N+1, t_{i+1}) = y(N+1, t_i) + h(0)x(N+1)$,

$$y(N+2,t_{i+1}) = y(N+2,t_i) + h(1)x(N+1),$$
$$\vdots$$
$$y(N+M,t_{i+1}) = y(N+M,t_i) + h(M-2)x(N+1),$$
$$y(N+M+1,t_{i+1}) = h(M-1)x(N+1).$$

At last, at a time moment t_{i+k} there appeares a sample $x(K)$. Values of signal $y(n)$ are determined as follows:

$$y(0,t_{i+k}) \equiv y(0,t_{i+k-1}), y(1,t_{i+k}) \equiv y(1,t_{i+k-1}), \cdots,$$

$$y(K-1,t_{i+k}) \equiv y(K-1,t_{i+k-1}), \tag{2}$$

while values $y(K,t_{i+k}), y(K+1,t_{i+k}), \cdots y(K+M-2,t_{i+k})$ are recalculated:

$$y(K,t_{i+k}) = y(K,t_{i+k-1}) + h(0)x(K), \cdots$$

$$y(K+1,t_{i+k}) = y(K+1,t_{i+k-1}) + h(1)x(K),$$
$$y(K+M-2,t_{i+k}) = y(K+M-2,t_{i+k-1}) + h(M-2)x(K), \tag{3}$$

recursively.

The last $y(n)$ value is:

$$y(K+M-1,t_{i+k}) = h(M-1)x(K). \tag{4}$$

The main feature of this procedure is adaptation of $y(n)$ to the new information in current $x(n)$ observation at every sampling interval T_s. The number of operations for speedy $y(n)_{new}$ calculation could be essentially reduced because the procedure does not require to repeat operations for $y(n)_{new}$ sample values that start from 0 and finish at $N-1$, where N is a time-varying integer. Remaining meanings of updated $y(n)_{new}$ are determined recursively,

adding ordinary corrections to the respective sample values of the previous $y(n)$.

For stopping of recursive calculations it is important to work out an efficient stopping rule. The calculations can be finished, when the inequalities

$$y(n) = \sum_{j=0}^{M-2} \{y(K+j, t_{i+k}) - y(K+j, t_{i+k-1})\}^2 \leq \mu_1, \tag{5}$$

$$\{y(K+M-1, t_{i+1})\}^2 \leq \mu_2 \tag{6}$$

are satisfied for some time period. Here μ_1 and μ_2 are thresholds to be chosen beforehand.

It is needed to terminate calculations, when desired accuracy of signal $y(n)$, in respect of μ_1 and μ_2 is achieved. The time for redundant calculations will be consumed, on the contrary.

4 Example: calculations, results and discussion

The LTI system's discrete-time input samples are:

$x(n)= \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$

$= \{24, 8, 12, 16, 20, 6, 10, 14\}$.

By inspection, the input sequence is running from 0 to 7 samples. The kernel

$h(n)= \{h(0), h(1), h(2), h(3), h(4), h(5), h(6), h(7)\}$

$= \{1, -.85, .85, -.7, .7, -.25, .25, -1\}$.

The number of kernel samples M=N, initially. Signals $x(n)$ and $h(n)$ are shown in Fig.1a and 1b, respectively. The third signal

$y(n) =\{y(0), y(1), y(2), y(3), y(4), y(5), y(6), y(7), y(8), y(9),$

$y(10), y(11), y(12), y(13), y(14)\}$ = { 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}

is determined by Matlab convolving given $h(n)$ and $x(n)$ by (1). Its length is $M + N$-1. The signal $y(n)$ is represented in Fig.2a. Arithmetical operations needed to calculate signal $y(n)$ contain 72 multiplications and 56 additions. The new $x(n)$ value $x(8)$=22 appeares. Then, input samples are (see Fig.1c):

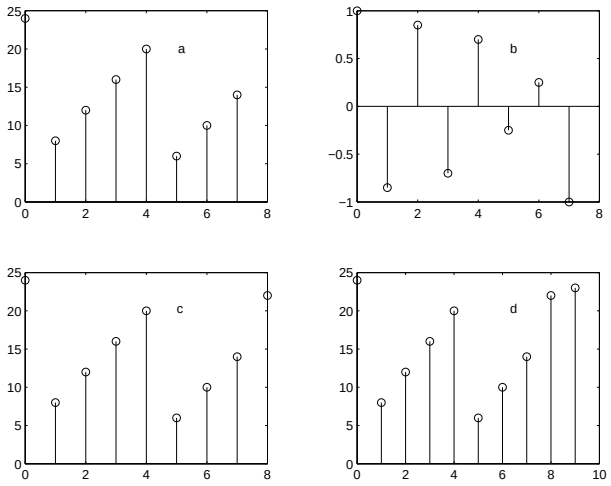

Figure 1: Signals: sequences $x(n)$ (a), $h(n)$ (b), $x(n)$ with $x(8)$=22 (c), and $x(n)$ with $x(9)$=23(d). Sample's number (axis X), sample's value (axis Y)

$$x(n) = \{24, 8, 12, 16, 20, 6, 10, 14, 22\}. \tag{7}$$

The length of $h(n)$ is the same (see Fig.1b). After recalculation of $y(n)$ we have: $y(n)_{new1}$={24, -12.4, 25.6, -4.2, 27.8, -6.2, 23.1, -17.2, 19.4, -22.6, 3.4, -26.6, 8.4, -12, -8.5, -22}. The updated signal $y(n)_{new1}$ is represented in Fig.2b. On close inspection, we notice that there exist here important equalities: $y(0)_{new1}$=$y(0)$, $y(1)_{new1}$=$y(1)$, $y(2)_{new1}$=$y(2)$, $y(3)_{new1}$=$y(3)$, $y(4)_{new1}$=$y(4)$, $y(5)_{new1}$=$y(5)$, $y(6)_{new1}$=$y(6)$, $y(7)_{new1}$=$y(7)$. Therefore, it is not needed to

calculate $y(n)_{new1}$ sample values for n less or equal than $N-1$. Remaining terms are calculated recursively:

$y(8)_{new1}$ =$y(8) + x(8) \cdot h(0)$ =-2.6+22·1=19.4;
$y(9)_{new1}$ =$y(9) + x(8) \cdot h(1)$ =-3.9+22·(−0.85)=-22.6;
$y(10)_{new1}$ =$y(10) + x(8) \cdot h(2)$ =-15.3+22·0.85=3.4;
$y(11)_{new1}$ =$y(11) + x(8) \cdot h(3)$ =-11.2+22·(−0.7)=-26.6;
$y(12)_{new1}$ =$y(12) + x(8) \cdot h(4)$ =-7+22·0.7=8.4;
$y(13)_{new1}$ =$y(13) + x(8) \cdot h(5)$ =-6.5+22·(−0.25)=-12;
$y(14)_{new1}$ =$y(14) + x(8) \cdot h(6)$ =-14+22·0.25=-8.5;
$y(15)_{new1}$ =$x(8) \cdot h(7) = 22 \cdot (-1)$=-22;

In such a case, recursive updating of signal $y(n)$ required 8 multiplications and 7 additions, i.e. 15 elementary arithmetical operations, in total. Then, after appearing $x(n)$ sample $x(9)$=23 insert its value in the signal (see Fig.1d)

$$x(n) = \{24, 8, 12, 16, 20, 6, 10, 14, 22, 23\}. \tag{8}$$

Now, instead of doing redundant calculations, equate current terms of signal $y(n)_{new2}$ and previous ones $y(n)_{new1}$ as follows:

$$y(k)_{new2} = y(k)_{new1} \forall\ k \in \overline{0, 8}. \tag{9}$$

Remaining $y(n)$ values are calculated recursively, too, as follows

$y(9)_{new2}$ =$y(9)_{new1} + x(9) \cdot h(0)$ =-22.6+23·1=0.4;
$y(10)_{new2}$ =$y(10)_{new1} + x(9) \cdot h(1)$ =3.4+23·(−0.85)=-16.15;

$y(11)_{new2}$ =$y(11)_{new1} + x(9) \cdot h(2)$ =-26.6+23·0.85=-7.05;

$y(12)_{new2}$ =$y(12)_{new1} + x(9) \cdot h(3)$ =8.4+23·(−0.7)=-7.7;

$y(13)_{new2}$ =$y(13)_{new1} + x(9) \cdot h(4)$ =-12+23·0.7=4.1;

$y(14)_{new2}$ =$y(14)_{new1} + x(9) \cdot h(5)$ =-8.5+23·(−0.25)=-14.25;

$y(15)_{new2}$ =$y(15)_{new1} + x(9) \cdot h(6)$ =-22+23·0.25=-16.25;

$y(16)_{new2}$ =$x(9) \cdot h(7)$ = 23·(−1)=-23;

We obtain: $y(n)_{new2}$: {24, -12.4, 25.6, -4.2, 27.8, -6.2, 23.1, -17.2, 19.4, 0.4, -16.15, -7.05, -7.7, 4.1, -14.25, -16.25, -23} (see Fig.2c).

Signal $y(n)$ is again updated recursively using the same number, i.e. 15 arithmetical operations. We obtain 158 operations for $y(n)_{new2}$, in total.

Suppose now, that all signal $x(n)$ values, including $x(8)$ and $x(9)$ are stored

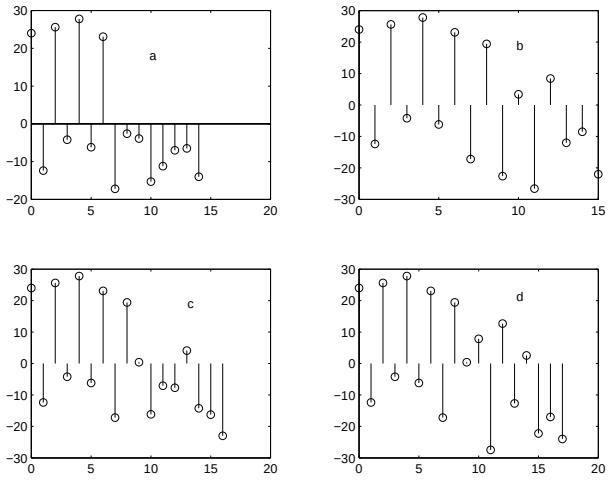

Figure 2: Results: $y(n)$ (a), $y(n)_{new1}$ (b), $y(n)_{new2}$ (c), and $y(n)_{new3}$ for $x(n)_{new3}$=24 (d)

in the procesor's memory. Afterwards, the batch data processing, as usually, is used by ordinary offline convolution formula (1). In such a case, signal $x(n)$= {24, 8, 12, 16, 20, 6, 10, 14, 22, 23}, while $h(n)$= {1, -.85, .85, -.7, .7,

-.25, .25, -1}. Signal $\hat{y}(n)$ was calculated by Matlab programm. Its values coincide with respective values of signal $y(n)_{new2}$.

However, to calculate signal $y(n)_{new2}$ offline is needed to use 100 multiplications and 81 additions. Thus, the number of operations can be decreased if the recursive procedure to determine $y(n)$ by means of online convolution is applied. Samples (Fig.1a,1b) represent initial input $x(n)$ (Fig.1a) and kernel $h(n)$ (Fig.1b), respectively, used in (1) to determine sequence $y(n)$ (Fig.2a). Samples, shown in (Fig.1c) and (Fig. 1d) correspond to discrete-time sequences $x(n)$ (7) and (8). Samples (Fig.2), varying values of which corespond to the distinct signals $x(n)$ (see Fig.1), respectively, are determined by ordinary convolution (1). Then, they were checked by recursive procedure (2) – (4). Results, obtained twofold, are coincident. The discrete-time curve (Fig.2d) correspond to signal $y(n)$ calculated for input $x(n) = \{24, 8, 12, 16, 20, 6, 10, 14, 22, 23, 24\}$.

Recursive convolution procedure (2) – (4) is proposed to update the ordinary convolution (1), when at each sampling period T_s the new input signal's $x(n)$ sample emerges in the given realization $x(n)$. Recursive procedure borrows the calculation results of the ordinary one but only for the first current recursion, continuing previous calculations (1) by equations (2) – (4). In each current recursion only M multiplications and M-1 adds are needed. The preference of the recursive procedure against the ordinary one consists as well as in the avoiding repeating calculations: each recursion starts not from the very begining as it could be using ordinary convolution (1), but by immediate processing current $x(n)$ sample.

5 Conclusion

The recursive convolution technique could be effective in real-time applications for very large N and relatively small kernel $h(n)$. It could be further accelerated by resolving the available procedure (2) – (4) into elementary recursive subprocedures, working in parallel. In addition, the effective online stopping rule (5), (6) allows to stop calculations, avoiding redundant mathematical operations.

R. Pupeikis (*The DMSTI MIF VU, Vilnius, LT*)

E-mail: rimantas.pupeikis@mif.vu.lt

References

[1] Smith,S. Digital signal processing: a practical guide for engineers and scientists, 2013, Elsevier Science.

[2] Proakis, J.G. and Manolakis D.G. Digital signal processing: principles, algorithms and applications, 1999, Prentice Hall, London.

[3] K. Pavel and D. Svoboda, "Algorithms for efficient computation of convolution", Chapter 8 of the book: Design and Architectures for Digital Signal Processing, edited by G. Ruiz and J.A. Michell, 2013. doi: 10.5772/51942.

[4] Pupeikis, R.: Revised linear convolution. *Lietuvos matematikos rinkinys*, Proc. of the Lith.math.society, ser.A, **60**, 2019, p.33 – 38, doi:10.15388/LMR.A.2019 14961