Lietuvos matematikos rinkinys ISSN 0132-2818
Proc. of the Lithuanian Mathematical Society, Ser. A Vol. **57**, 2016
DOI: 10.15388/LMR.A.2016.18 pages 97–102

Revised fast convolution

Rimantas Pupeikis

Institute of Mathematics and Informatics, Vilnius University
Akademijos str. 4, LT-08663 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 a convolution as well. The idea is that fast Fourier transform (FFT) algorithm, used to calculate output frequency samples (f.s.), should not be recalculated with every new input sample. It is needed just to modify the convolution algorithm, when the new input sample replaces the old one. An example of computation of the convolution with ordinary and modified 8-point Fourier code matrix is presented.
Keywords: LTI system, DFT, IDFT, FFT, convolution.

1 Introduction

Convolution is a mathematical tool in digital signal and image processing [3, 2]. It is used in filtering, correlation, compression and in many other applications [3]. Although the concept of convolution is not new, the efficient computation of convolution is still an open topic [3]. As the burden of data is constantly increasing, there appears request for fast manipulation with large data. In wireless sensor networks, where a new set of input samples simultaneously replaces previous one, it is non-effective to recalculate convolution each time, even with FFT procedures [1], when only small part of new samples differs from previous one. It is needed to modify discrete FT (DFT) in order to recalculate on-line only some products of the convolution with the respective samples replaced [4].

2 Statement of the problem

Suppose that $x(n)$ is an arbitrary input of a LTI system having a kernel $h(n)$, too. The output $y(n)$ of the system is the linear convolution of the form

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

Here $\star$ is the asterisk of the convolution. The DFT of (1) is known as a fast convolution (Fig. 1) [3]:

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

where

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

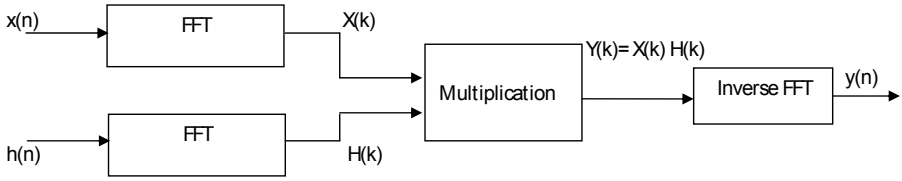

Fig. 1. Fast convolution. Signals: $\mathbf{x}(n)$, $\mathbf{h}(n)$, $\mathbf{y}(n)$. DFTs: $\mathbf{X}(k)$, $\mathbf{H}(k)$, $\mathbf{Y}(k)$.

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.

In a convolution scheme we consider a discrete-time finite duration real-valued signal $x(n)$ of length L (i.e., $x(n) = 0$ for $n < 0$ and $n \geqslant L$) that has the Fourier transform

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

where j is the imaginary unit. When we sample $X(\omega)$ at equally spaced frequencies $\omega_k = 2\pi k/N$, $\forall\, k \in \overline{0, N-1}$, with $N \geqslant L$, the resultant samples are as follows [3]:

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

For convenience, the upper index in the sum has been increased from $L-1$ to $N-1$ since $x(n) = 0$ for $n \geqslant L$. 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.

Assume that at any moment t_i the network of sensors is simultaneously evaluated the set of f.s. $X(k)$ by processing the signal samples $x(n)$. At time moment t_{i+1} the new set of current samples $x(n)$ enter memory replacing the previous samples. For the moment it is determined that most of signal's samples, indeed, are approximately equal to the previous ones. Only small part of current samples is different.

The aim of the paper is to update on-line the fast and the linear convolutions.

3 Recursive updating

It is not efficient to recalculate the basic spectrum samples $X(k)$ anew even using FFT algorithms, if only one new signal sample $x(i)$ or even a small portion of new samples emerges replacing previous samples. Therefore, we will use the solution of a convolution problem, applying the recursive DFT algorithm.

At any time moment t a signal $y(n)$ can be recovered from frequency samples $Y(k)$, $\forall\, k \in \overline{0, N-1}$ by the IDFT (inverse DFT) [1, 3]:

$$y_t(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y\left(\frac{2\pi k}{N}\right) e^{j2\pi kn/N}, \quad \forall\, n \in \overline{0, N-1}. \tag{5}$$

At $t+1$ time moment in sensor network some new samples of signal $x(n)$ emerge and replace the previous ones. In such a case, (5) can be rewritten as follows

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

In (6):

$$\Delta Y(k) \equiv \Delta Y\left(\frac{2\pi k}{N}\right) \equiv \Delta X\left(\frac{2\pi k}{N}\right)H\left(\frac{2\pi k}{N}\right) \equiv \Delta X(k)H(k). \tag{7}$$

In the matrix form $\mathbf{\Delta}X(k)$ is the $(N \times 1)$ correction vector in the frequency-domain of the form

$$\begin{bmatrix} \Delta X(0) \\ \Delta X(1) \\ \Delta X(2) \\ \vdots \\ \underbrace{\Delta X(N-1)}_{corrections} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N & W_N^2 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \dots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta x(1) \\ \Delta x(2) \\ \vdots \\ \underbrace{\Delta x(N-1)}_{corrections} \end{bmatrix}, \tag{8}$$

which is related with the time domain $(N \times 1)$ correction vector $\mathbf{\Delta x}(k)$ by $(N \times N)$ Fourier code matrix $\mathbf{W}_N$ with twiddle factors: $W_N, W_N^2, \dots, W_N^{N-1}, \dots$.

Let us assume that only four values $\Delta x(0), \Delta x(2), \Delta x(4)$ and $\Delta x(N-1)$ in the right-hand side correction vector are not equal to zeros. Expression (8) obtains the form

$$\begin{bmatrix} \Delta X(0) \\ \Delta X(1) \\ \Delta X(2) \\ \vdots \\ \underbrace{\Delta X(N-1)}_{corrections} \end{bmatrix} == \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_N^2 & W_N^4 & W_N^{N-1} \\ 1 & W_N^4 & W_N^8 & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & W_N^{2(N-1)} & W_N^{4(N-1)} & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta x(2) \\ \Delta x(4) \\ \underbrace{\Delta x(N-1)}_{corrections} \end{bmatrix}, \tag{9}$$

because most columns of matrix $\mathbf{W}_N$ and respective rows of the right-hand side correction vector in (8) were deleted, and, then, compressed. Now, matrix $\mathbf{W}_N$ has the size $(N \times 4)$. The size of the right hand-side correction vector is (4×1). Thus, the calculation operations of $\Delta X(k)$, using compressed $\mathbf{W}_N$ and $\mathbf{\Delta x}(k)$ are reduced significantly. Afterwards, $\Delta Y(k)$ is calculated, using the same $\Delta X(k)$ and (7). Then, it is substituted in (6). We obtain

$$y_{t+1}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\left\{Y(k)e^{j2\pi kn/N} + \Delta Y(k)e^{j2\pi kn/N}\right\}. \tag{10}$$

In recursive form (10) is

$$y_{t+1}(n) = y_t(n) + \frac{1}{N}\sum_{k=0}^{N-1}\Delta Y(k)e^{j2\pi kn/N}. \tag{11}$$

Recursive formula (11) gives us possibility to update the output's samples on-line.

4 Example

Let us assume that the LTI system's discrete-time periodical input samples are: $x(n) = \{\ldots 24, 8, 12, 16, 20, 6, 10, 14, \ldots\}$. By inspection, the period of the input is $N = 8$. The DFT is computed by [4]

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ \underbrace{X(7)}_{current\ f.s.} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & W_8 & W_8^2 & \ldots & W_8^7 \\ 1 & W_8^2 & W_8^4 & \ldots & W_8^{14} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W_8^7 & W_8^{14} & \ldots & W_8^{49} \end{bmatrix} \begin{bmatrix} 24 \\ 8 \\ 12 \\ 16 \\ 20 \\ 6 \\ 10 \\ 14 \end{bmatrix} = \begin{bmatrix} 110 \\ 4 - 4.83j \\ 22 + 16j \\ 4 - 0.83j \\ 22 \\ 4 + 0.83j \\ 22 - 16j \\ 4 + 4.83j \end{bmatrix}, \tag{12}$$

using the known Fourier 'code' matrix with twiddle factors as follows:

$$W_8 = W_8^9 = W_8^{25} = W_8^{49} = a(1-j), \qquad W_8^2 = W_8^{10} = W_8^{18} = W_8^{42} = -j,$$
$$W_8^3 = W_8^{35} = b(1+j), \qquad W_8^4 = W_8^{12} = W_8^{20} = W_8^{28} = W_8^{36} = -1,$$
$$W_8^5 = W_8^{21} = b(1-j), \qquad W_8^6 = W_8^{14} = W_8^{30} = j, \qquad W_8^7 = W_8^{15} = a(1+j),$$
$$W_8^8 = W_8^{16} = W_8^{24} = 1.$$

Here $a = 0.7071$ and $b = -a$. Assume that the kernel $h(n) = \{\ldots 1, -0.85, 0.85, -0.7, 0.7, -0.25, 0.25, -0.1, \ldots\}$. It has the same period. The DFT of $h(n)$ is:

$$\begin{bmatrix} H(0) \\ H(1) \\ H(2) \\ H(3) \\ H(4) \\ H(5) \\ H(6) \\ \underbrace{H(7)}_{current\ f.s.} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & W_8 & W_8^2 & \ldots & W_8^7 \\ 1 & W_8^2 & W_8^4 & \ldots & W_8^{14} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W_8^7 & W_8^{14} & \ldots & W_8^{49} \end{bmatrix} \begin{bmatrix} 1 \\ -0.85 \\ .85 \\ -0.7 \\ 0.7 \\ -0.25 \\ 0.25 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.3 + 0.248j \\ 0.6 + 0.3j \\ 0.3 + 1.448j \\ 4.7 \\ 0.3 - 1.448j \\ 0.6 - 0.3j \\ 0.3 - 0.248j \end{bmatrix}. \tag{13}$$

Then, the f.s. $Y(k) = 100\{0.99, 0.024 - 0.0045j, 0.084 + 0.162j, 0.024 + 0.0555j, 1.034, 0.024 - 0.0555j, 0.084 - 0.162j, 0.024 + 0.0045j\}$ of the filter output $y(n)$ are obtained by (2). Afterwards, IDFT was computed by Matlab function: $y = \text{ifft}(Y, 8)$. We obtain eight discrete-time samples of the system output signal $y(n)$. They are: $28.6, -5.5, 24.7, 2.6, 26.2, -3.7, 21.7, 4.4$.

Assume that sensors send current portion of $x(n)$ samples, between which only samples $x(0), x(2), x(4)$ and $x(7)$ differ in values from previous ones. At the moment, they are: $x(0) = 20$, $x(2) = 15$, $x(4) = 25$, $x(7) = 10$. Thus, $\Delta x(0) = -4$, $\Delta x(2) = 3$,

$\Delta x(4) = 5$, $\Delta x(7) = -4$. Then, $\Delta X(k)$ is computed according to

$$\begin{bmatrix} \Delta X(0) \\ \Delta X(1) \\ \Delta X(2) \\ \Delta X(3) \\ \Delta X(4) \\ \Delta X(5) \\ \Delta X(6) \\ \Delta X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_8^2 & W_8^4 & W_8^7 \\ 1 & W_8^4 & W_8^8 & W_8^{14} \\ 1 & W_8^6 & W_8^{12} & W_8^{21} \\ 1 & W_8^8 & W_8^{16} & W_8^{28} \\ 1 & W_8^{10} & W_8^{20} & W_8^{35} \\ 1 & W_8^{12} & W_8^{24} & W_8^{42} \\ 1 & W_8^{14} & W_8^{28} & W_8^{49} \end{bmatrix} \begin{bmatrix} -4 \\ 3 \\ 5 \\ -4 \end{bmatrix} = \begin{bmatrix} 0 \\ -11.83 - 5.83j \\ -2 - 4j \\ -6.17 + 0.17j \\ 8 \\ -6.17 - 0.17j \\ -2 + 4j \\ -11.83 + 5.83j \end{bmatrix}, \qquad (14)$$

and multiplied by $H(k)$ from (13). Afterwards, using IDFT for their product we obtain the third term in recursive expression (11). It was added to $y_t(n)$, finally. We obtain the updated values of current $y_{t+1}(n)$ according to (11) as follows: $32.25, -7.05, 28.35, -0.45, 31.95, -10.05, 27.45, -3.45$. We calculate f.s. of current $Y(k)$: fft($[32.25, -7.05, 28.35, -0.45, 31.95, -10.05, 27.45, -3.45]$). Then, we checked up the recursive solution with Matlab: firstly, f.s. $X(k)$ were calculated by fft($[20, 8, 15, 16, 25, 6, 10, 10], 8$), secondly, f.s. $Y(k)$ were obtained as product of $X(k)$ and $H(k)$, thirdly, samples of $y(k)$ were determined by ifft. The results obtained by recursive (11) and ordinary methods are coincident.

The example shows us how effective can be the recursive approach. It allows us to reduce Fourier code matrix $\mathbf{W}_N$ in (9). It let us to cut the total number of CMADs (complex multiplications and additions) needed for calculations with complex valued f.s. In such a case, we spend here 16 CMADs. At last, it assures updating of linear (1) and fast (2) convolutions on-line.

5 Conclusions

The fast convolution have been proposed to recursively determine if one new signal sample or new small portion of samples emerge in the given period N of a realization $x(n)$ replacing the old one sample or old portion of samples, respectively. The number of operations for their speedy calculating is essentially reduced by the original recursive expression (11) in comparison with the ordinary FFT procedure used only in the case of fixed values of samples $x(n)$. The recursive algorithm could be effective in real-time applications for very large N.

References

[1] P. Duhamel and M. Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Sign. Proc.*, **19**:259–299, 1990.

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

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

[4] R. Pupeikis. Fast Fourier transform revisited. *Liet. matem. rink. Proc. LMS, ser. A*, **56**:113–118, 2015. DOI:10.15338/LMR.A.2015.20.

REZIUMĖ

Patikslinta greitoji sąsūka

R. Pupeikis

Tariama, kad taikant diskrečiąją Furjė transformaciją, signalo atskaitų apdorojimui skaitmeniškai, kai kurios jo atskaitos esti jutiklių, veikiančių laiku, pakeičiamos naujomis atskaitomis. Būtina kiekvienai naujai atsiunčiamai atskaitai skaičiuoti naują sistemos išėjimą. Siūloma neperskaičiuoti išėjimo spektrą naujai, o jį modifikuoti rekurentiškai, žymiai sutaupant aritmetinių operacijų skaičių. Pateiktas sąsūkos skaičiavimo įprastiniu ir rekurentiniu būdais diskrečiąja Furjė transformacija pavyzdys.

Raktiniai žodžiai: tiesinė sistema, DFT, IDFT, GFT, sąsūka.