On-line Approach for Fast Convolution over Sensor Networks

Dalius Navakauskas, Rimantas Pupeikis

Department of Electronic Systems, Faculty of Electronics of Vilnius Gediminas Technical University, Naugarduko 41, LT-03227 Vilnius, Lithuania

Abstract **– It is assumed that at some time moment, in wireless sensor networks the new set of current samples of input and system impulse response enter a digital memory replacing the previous samples. It is urgent for each new sample or for a small part of new samples to update a convolution as well. Therefore, a recursive fast convolution algorithm is proposed here to solve a linear filtering problem for a nonstationary system. The calculation operations are reduced because most columns of Fourier code matrices and respective rows of the right-hand side vectors were deleted for equal previous and current samples. An example with the ordinary and modified 8-point Fourier code matrices is presented for a nonstationary linear system. The amounts of operations, necessary for recursive and fast Fourier transform algorithms, are calculated. Results are discussed, and the conclusion is given.**

Keywords **– system, response, signal, convolution, filtering.**

1. Introduction

Convolution is a well-known mathematical tool in applied mathematics, physics as well as in digital sound signal and image processing [1-4]. It is used in filtering, correlation, compression, and in many other computer based applications, too [2, 6]. It is emphasized in [2] that though the concept of convolution is not new, the efficient computation of convolution is still an open topic. As the burden of data is constantly increasing, there arises a need for fast manipulation with large data [4]. Therefore, recursive convolution algorithms are worked out in order to significantly improve the efficiency, for example, artificial neural networks or loose coupled transmission lines (CLINP) [7]. Although more algorithms were proposed for the time-saving convolution computation, they are based, as a rule, on the ordinary signal observation schema that is appropriate, for example, for ordinary digital signal processing [8] or for system identification [9], but not for sensor networks [10]. In wireless sensor networks, where the new sets of input and impulse response samples ought to simultaneously replace the previous ones, it is not effective to recalculate the convolution each time, even with fast Fourier transform (FFT) procedures [1, 11, 12], when only a small part of new samples or even one sample differs from the previous one. The idea is that the fast Fourier transform algorithms, used to calculate frequency samples of input-output, and of a varying frequency response, too, should not be recalculated with each new current sample. It is necessary to modify discrete Fourier transform (DFT) in order to recalculate on-line only some convolution products with the respective samples replaced [12].

The rest of the paper is organized as follows. The next section introduces the statement of the problem. In Section 3 we propose the recursive algorithm to solve fast convolution problem on-line. In Section 4 the example is given. In Section 5 the total amount of computer operations for ordinary and recursive methods is calculated. Section 6 contains simulation results and discussion. In Section 7 the conclusion is given.

2. Statement of the problem

Suppose that $x(n), \forall n = \overline{0, N-1}$ is an arbitrary input of a dynamical system having the impulse response (kernel) $h(n), \forall n = \overline{0, N-1}$. The output $y(n), \forall n = \overline{0, N-1}$ of a linear system is a linear convolution of the form

DOI: 10.18421/TEM72-01
https://dx.doi.org/10.18421/TEM72-01

Corresponding author: Rimantas Pupeikis,
Faculty of Electronics of Vilnius Gediminas Technical University, Vilnius, Lithuania
Email: rimantas.pupeikis@vgtu.lt

Received: 18 November 2017.
Accepted: 26 February 2018.
Published: 25 May 2018.

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) = x(n) * h(n), \quad (1)$$

assuming that discrete-time finite duration real-valued signals $x(n), y(n)$ and $h(n), \forall n = \overline{0, N-1}$ are of length L (i.e., $x(n) = 0, y(n) = 0$ and the kernel $h(n) = 0$ for $n < 0,$ and $n \geq L$), $*$ is convolution's asterisk. DFT of (1) is known as a fast convolution – [11]:

$$Y[k] = H[k]X[k] \; \forall k = \overline{0, N-1}. \quad (2)$$

In (2):

$$Y[k] = Y\left[\frac{2\pi k}{N}\right], \; H[k] = H\left[\frac{2\pi k}{N}\right], \quad X[k] = X\left[\frac{2\pi k}{N}\right] \quad (3)$$

are DFTs of $y(n), h(n)$ and $x(n),$ respectively, assuming, for simplicity, that N is the total number of samples. The frequency samples (f.s.) of the respective signal are:

$$X\left[\frac{2\pi k}{N}\right] = \sum_{n=0}^{N-1} x(n)e^{-j2\pi k/N}, \quad (4)$$

$$Y\left[\frac{2\pi k}{N}\right] = \sum_{n=0}^{N-1} y(n)e^{-j2\pi k/N}, \quad (5)$$

$$H\left[\frac{2\pi k}{N}\right] = \sum_{n=0}^{N-1} h(n)e^{-j2\pi k/N} \quad (6)$$

$\forall n = \overline{0, N-1},$ respectively, when we sample $X[\omega]$, $Y[\omega]$, and $H[\omega]$ at equally spaced frequencies $\omega_k = 2\pi k/N \;\; \forall k = \overline{0, N-1}$ with $n \geq L.$ In (4) – (6), j is an imaginary unit. Note that, for convenience, the upper index in the sum has been increased from $L-1$ to $N-1,$ since $x(n) = 0$ for $n \geq L.$ The relations in (4) – (6) are used to transform the sequences $x(n)$, $y(n)$ and $h(n), \forall n = \overline{0, N-1}$ into f.s. $X[k], Y[k],$ and $H[k]$ of length N.

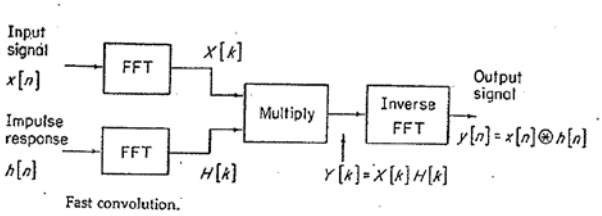

Figure 1 .Fast convolution.

Assume that at any moment t a sensors network is simultaneously evaluating the sets of f.s. $X[k]$ and $H[k]$ by processing the signal $x(n), h(n)$ samples, respectively. At the time moment $t + 1$ the new sets of current samples $x(n)$ and $h(n), \forall n = \overline{0, N-1}$ enter a digital memory replacing the previous samples. At the same moment, it is determined that most of the signal samples are approximately equal to the previous samples. Only particular – m_1 and m_2 parts of current samples $x(n),$ and $h(n)$ are different, respectively, and there exist m alterations that are not equal to zero. Here the alterations total the number: $m = m_1 + m_2.$ The aim of the paper is to update the fast convolution (2) on-line over the sensor network efficiently, i.e. without redundant complex multiples and adds (CMADs), when the previous samples of respective signals are partly replaced.

3. Recursive updating

It is known that the lowest time complexity can be achieved by using the recursive filtering [7, 9, 11, 12, 13]. Recursive filters are suitable for streaming architecture, such as field-programmable gate array (FPGA) [15]. Real-time techniques for implementing partitioned convolution are effective for processing audio signals [5], as well.

It is not correct in real time to recalculate the basic spectrum samples $X(k)$ and/or $H(k)$ anew even using FFT algorithms, if only some new samples of the signals $x(n),$ and $h(n)$ emerge replacing the previous ones. Therefore, we propose here also a recursive algorithm worked out to deal with a linear filtering problem for a nonstationary system. Note that, for the stationary system the recursive expressions are written in [13, 14].

At any time moment t a signal $y(n)$ can be recovered from the f.s. $Y[k]$ by the inverse discrete Fourier transform (IDFT) [11]:

$$y_t(n) = \frac{1}{N}\sum_{k=0}^{N-1} Y\left[\frac{2\pi k}{N}\right] e^{j2\pi kn/N} \forall n = \overline{0, N-1}. \quad (7)$$

At the time moment $t + 1$, m_1 new samples of signal $x(n), \forall n = \overline{0, N-1}$ and m_2 new samples of signal $h(n), \forall n = \overline{0, N-1}$ among N current samples, emerge in a sensor network and replace the previous ones. Then, (7) can be rewritten as follows

$$y_{t+1}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\left\{Y\left[\frac{2\pi k}{N}\right] + \Delta Y\left[\frac{2\pi k}{N}\right]\right\} e^{j2\pi kn/N}. \quad (8)$$

Here:

$$\Delta Y\left[\frac{2\pi k}{N}\right] = \Delta H\left[\frac{2\pi k}{N}\right]\Delta X\left[\frac{2\pi k}{N}\right] + H\left[\frac{2\pi k}{N}\right]\Delta X\left[\frac{2\pi k}{N}\right] + \Delta H\left[\frac{2\pi k}{N}\right]\Delta X\left[\frac{2\pi k}{N}\right], \quad \forall k = \overline{0, N-1}, \quad (9)$$

where $\Delta X\left[\frac{2\pi k}{N}\right]$, and $\Delta H\left[\frac{2\pi k}{N}\right]$ are alterations of f.s. $X[k]$, and $H[k]$, respectively, at the moment $t + 1$; besides,

$$\begin{bmatrix} \Delta X(0) \\ \Delta X(1) \\ \Delta X(2) \\ \vdots \\ \Delta X(N-1) \end{bmatrix} = W_N \begin{bmatrix} \Delta x(0) \\ \Delta x(1) \\ \Delta x(2) \\ \vdots \\ \Delta x(N-1) \end{bmatrix}, \quad (10)$$

and

$$\begin{bmatrix} \Delta H(0) \\ \Delta H(1) \\ \Delta H(2) \\ \vdots \\ \Delta H(N-1) \end{bmatrix} = W_N \begin{bmatrix} \Delta h(0) \\ \Delta h(1) \\ \Delta h(2) \\ \vdots \\ \Delta h(N-1) \end{bmatrix}, \quad (11)$$

with

($N\times N$) Fourier code matrix

$$W_N = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & w_N & \dots & w_N^{N-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & w_{N-1}^N & \dots & w_N^{(N-1)(N-1)} \end{bmatrix} \quad (12)$$

having the respective twiddle factors:

$$w_N, w_N^2, \dots, w_N^{N-1}, \dots, w_N^{(N-1)(N-1)}.$$

Let us assume that, at the time moment t+1, only m_1 and m_2 values of current samples $x(n)$ and $h(n)$, among the right-hand side correction vectors in (10), (11), respectively, are not equal to zeros. Thus, the calculation operations of (10) and (11), needed to multiply W_N and respective correction vectors are significantly reduced because most columns of matrix W_N in (10), (11) and respective rows of the right-hand side correction vectors were deleted in the same expression. Note that, usually the sizes of matrices will be different due to the different number of alterations in both equations.

Afterwards, the value of the variable (9) is calculated. Then, it is substituted in (8), which, in recursive form, is

$$y_{t+1}(n) = y_t(n) + \frac{1}{N}\sum_{k=0}^{N-1}\Delta Y\left[\frac{2\pi k}{N}\right] e^{j2\pi kn/N} \quad (13)$$

or

$$y_{t+1}(n) = y_t(n) + \Delta y_{t+1}(n). \quad (14)$$

Here, at the time moment t+1, a correction is

$$\Delta y_{t+1}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\Delta Y\left[\frac{2\pi k}{N}\right] e^{j2\pi kn/N}. \quad (15)$$

Recursive formula (14) renders us a possibility to update a nonstationary system or filter on-line.

4. Example

Assume that input samples of nonstationary system are: $x(n)$= {… 24, 8, 12, 16, 20, 6, 10, 14, …}. The period of the input is N=8. DFT is computed by

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & w_8 & \dots & w_8^7 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & w_8^7 & \dots & w_8^{49} \end{bmatrix} \begin{bmatrix} 24 \\ 8 \\ 12 \\ 16 \\ 20 \\ 6 \\ 10 \\ 14 \end{bmatrix} = \begin{bmatrix} 110 \\ 4-4.83j \\ 22+16j \\ 4-0.83j \\ 22 \\ 4+0.83j \\ 22-16j \\ 4+4.83j \end{bmatrix},$$

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

$w_8 = w_8^9 = w_8^{25} = w_8^{49} = a(1-j)$, $w_8^2 = w_8^{10} = w_8^{18} = w_8^{42} = -j$, $w_8^3 = w_8^{35} = b(1+j)$, $w_8^4 = w_8^{12} = w_8^{20} = w_8^{28} = w_8^{36} = -1$, $w_8^5 = w_8^{21} = b(1-j)$, $w_8^6 = w_8^{14} = w_8^{30} = j$, $w_8^7 = w_8^{15} = a(1+j)$, $w_8^8 = w_8^{16} = w_8^{24} = 1$.

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

$$\begin{bmatrix} H(0) \\ H(1) \\ H(2) \\ H(3) \\ H(4) \\ H(5) \\ H(6) \\ H(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & w_8 & \dots & w_8^7 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & w_8^7 & \dots & w_8^{49} \end{bmatrix} \begin{bmatrix} 1 \\ -0.85 \\ 0.85 \\ -0.7 \\ 0.7 \\ -0.25 \\ 0.25 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.3+0.248j \\ 0.6+0.3j \\ 0.3+1.448j \\ 4.7 \\ 0.3-1.448j \\ 0.6-0.3j \\ 0.3-0.248j \end{bmatrix}.$$

Then, $H(k)\ X(k) = Y(k)$ is

$$\begin{bmatrix} 0.9 \\ 0.3+0.248j \\ 0.6+0.3j \\ 0.3+1.448j \\ 4.7 \\ 0.3-1.448j \\ 0.6-0.3j \\ 0.3-0.248j \end{bmatrix} \times \begin{bmatrix} 110 \\ 4-4.83j \\ 22+16j \\ 4-0.83j \\ 22 \\ 4+0.83j \\ 22-16j \\ 4+4.83j \end{bmatrix} = 100 \begin{bmatrix} 0.99 \\ 0.024-0.0045j \\ 0.084+0.162j \\ 0.024+0.0555j \\ 1.034 \\ 0.024-0.0555j \\ 0.084-0.162j \\ 0.024+0.0045j \end{bmatrix}.$$

Afterwards, IDFT was computed by the Matlab function: y=ifft(Y, 8). We obtain eight discrete-time samples of the system output $y(n)$. They are: $y(0)$ = 28.6, $y(1)$ = –5.5, $y(2)$ = 24.7, $y(3)$ = 2.6, $y(4)$ = 26.2, $y(5)$ = –3.7, $y(6)$ = 21.7, $y(7)$ = 4.4. Note that the given samples correspond to the signal y_t (n) in (14).

Assume that now sensors send the current portion of $x(n)$ samples, among which only the samples $x(0)$, $x(2)$, $x(4)$, and $x(7)$ differ in values from the previous ones. At the moment, they are: $x(0)$ = 20, $x(2)$ = 15, $x(4)$ = 25, $x(7)$ = 10. Thus, alterations are: $\Delta x(0) = -4$, $\Delta x(2) = 3$, $\Delta x(4) = 5$, $\Delta x(7) = -4$. At the same time moment there appear observation samples of $h(n)$: $h(0) = 1$, $h(1) = -0.5$, $h(2) = 0.85$, $h(3) = -0.5$, $h(4) = 0.7$, $h(5) = -0.4$, $h(6) = 0.25$, $h(7) = -0.1$.

In the case of the kernel h(n), the alterations are: $h(1) = 0.35$, $h(3) = 0.2$, $h(5) = -0.15$. The rest of the samples are unchangeable.

The alteration vector $\Delta \boldsymbol{X}(k)$ is:

$$\begin{bmatrix} \Delta X(0) \\ \Delta X(1) \\ \Delta X(2) \\ \Delta X(3) \\ \Delta X(4) \\ \Delta X(5) \\ \Delta X(6) \\ \Delta X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & w_8 & \dots & w_8^7 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & w_8^{14} & \dots & w_8^{49} \end{bmatrix} \begin{bmatrix} -4 \\ 3 \\ 5 \\ -4 \end{bmatrix} = \begin{bmatrix} 0 \\ -11.8+5.8j \\ -2+4j \\ -6.2-0.2j \\ 8 \\ -6.2+0.2j \\ -2-4j \\ -11.8-5.8j \end{bmatrix}.$$

Then, the alteration $\Delta \boldsymbol{H}(k)$ is computed according to

$$\begin{bmatrix} \Delta H(0) \\ \Delta H(1) \\ \Delta H(2) \\ \Delta H(3) \\ \Delta H(4) \\ \Delta H(5) \\ \Delta H(6) \\ \Delta H(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ w_8 & w_8^3 & w_8^5 \\ w_8^2 & w_8^6 & w_8^{10} \\ w_8^3 & w_8^9 & w_8^{15} \\ w_8^4 & w_8^{12} & w_8^{20} \\ w_8^5 & w_8^{15} & w_8^{25} \\ w_8^6 & w_8^{18} & w_8^{30} \\ w_8^7 & w_8^{21} & w_8^{35} \end{bmatrix} \begin{bmatrix} 0.35 \\ 0.2 \\ -0.15 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.21-0.49j \\ 0+0j \\ -0.21-0.49j \\ -0.4 \\ -0.21+0.49j \\ 0-0j \\ 0.21+0.49j \end{bmatrix},$$

Multiplying the respective sequences, we obtain:

$$\boldsymbol{H}(k)\,\Delta\boldsymbol{X}(k) = \begin{bmatrix} \Delta Y'(0) \\ \Delta Y'(1) \\ \Delta Y'(2) \\ \Delta Y'(3) \\ \Delta Y'(4) \\ \Delta Y'(5) \\ \Delta Y'(6) \\ \Delta Y'(7) \end{bmatrix} = \begin{bmatrix} 0 \\ -2.1-4.68j \\ -3j \\ -2.1-8.88j \\ 37.6 \\ -2.1+8.88j \\ 3j \\ -2.1+4.68j \end{bmatrix},$$

$$\Delta\boldsymbol{H}(k)\,\boldsymbol{X}(k) = \begin{bmatrix} \Delta H(0)X(0) \\ \Delta H(1)X(1) \\ \Delta H(2)X(2) \\ \Delta H(3)X(3) \\ \Delta H(4)X(4) \\ \Delta H(5)X(5) \\ \Delta H(6)X(6) \\ \Delta H(7)X(7) \end{bmatrix} = \begin{bmatrix} 44 \\ -1.54-3j \\ -0+0j \\ -1.25-1.8j \\ -8.8 \\ -1.25+1.8j \\ -0-0j \\ -1.54+3j \end{bmatrix},$$

and

$$\Delta\boldsymbol{H}(k)\,\Delta\boldsymbol{X}(k) = \begin{bmatrix} \Delta H(0)\Delta X(0) \\ \Delta H(1)\Delta X(1) \\ \Delta H(2)\Delta X(2) \\ \Delta H(3)\Delta X(3) \\ \Delta H(4)\Delta X(4) \\ \Delta H(5)\Delta X(5) \\ \Delta H(6)\Delta X(6) \\ \Delta H(7)\Delta X(7) \end{bmatrix} = \begin{bmatrix} 0 \\ -5.39+4.62j \\ -0j \\ 1.39+3.02j \\ -3.2 \\ 1.39-3.02j \\ 0j \\ -5.39-4.62j \end{bmatrix}.$$

Adding $\boldsymbol{H}(k)\,\Delta\boldsymbol{X}(k)$, $\Delta\boldsymbol{H}(k)\,\boldsymbol{X}(k)$ and $\Delta\boldsymbol{H}(k)\,\Delta\boldsymbol{X}(k)$ we determine $\Delta\boldsymbol{Y}(k)$ of form (9). Then, applying IDFT to (9), we calculate correction's value (15) in recursive expression (14). Finally, we determine

$$y_{t+1}(n): \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \\ y(4) \\ y(5) \\ y(6) \\ y(7) \end{bmatrix} = \begin{bmatrix} 34.55 \\ -1.8 \\ 32.25 \\ 7.3 \\ 37.65 \\ -1.3 \\ 31.55 \\ 2.8 \end{bmatrix}_{t+1}.$$

Concerning the operations number, we can also say that N is too small to show here the efficiency of recursive expression (14) against the ordinary FFT.

5. Operation numbers

Operation numbers of different fast convolution algorithms are given in Table 1, in which the first, the second and the third columns correspond to values: N, N^2 and $N\log_2 N$, respectively. It is known, that to calculate convolution sum (1) directly in the time domain we need N^2 operations approximately. The total amount of computer operations required to calculate $X(k)$, $Y(k)$, $H(k)$, and product $H(k)X(k)$, using FFT algorithms, consists of $3N\log_2 N$ CMADs operations and N products (the fourth column in table) (Fig.1), i.e.

$$O = 3N\log_2 N + N_{\text{products}}. \tag{16}$$

The recursive expression (14), proposed here, required approximately $3N\log_2 N$ CMADs operations, $3N$ products, and $3N$ adds

$$O_1 = 3(N\log_2 N + N_{\text{products}} + N_{\text{adds}}) \tag{17}$$

if data in $x(n)$ and $h(n)$ change all their values, i.e. $m = N$ (the fifth column in table, corresponding to eq.(16)), completely. On the other hand, $O_1 = 0$ if the comparators do not fix that both signals change their values in the current iteration. In such a case, $y_{t+1}(n) = y_t(n)$.

Table 1. Operation numbers of fast convolution algorithms

1	2	3	4	5	6
2^1	4	2	8	18	18
2^2	16	8	28	48	40
2^3	64	24	80	120	88
2^4	256	64	208	288	192
2^5	1024	160	512	672	416
2^6	4096	384	1216	1536	896
2^7	16384	896	2816	3456	1920
2^8	65536	2048	6400	7680	4096
2^9	262144	4608	14336	16896	8704
2^{10}	1048576	10240	31744	36864	18432
2^{11}	4194304	22528	69632	79872	38912
2^{12}	16777216	49152	151552	172032	81920

Let us assume now that at the time moment $t+1$ each of signals $x(n)$ and $h(n)$ changes only one value, i.e. $m_1 = m_2 = 1$. Then, (17) takes the shape

$$O_1 = N \log_2 N + 5N_{\text{products}} + 3N_{\text{adds}}. \quad (18)$$

One can define that, at the time moment t+1, value O_1 (the sixth column in table, corresponding to eq. (18)) reduces considerably, especially with gross N. On the other hand, there exist maximal alteration number $m = m_{\max}$, for which inequality

$$O_1 \leq O \quad (19)$$

is not valid.

In such a case, the ordinary FFT gains an advantage over recursive expression (14).

6. Results and discussion

The results of simulation of varying operation numbers, dependent on values of 2^q are given in Fig. 2. Here the curves are shown, the values of which correspond to the – base 10 logarithms of needed operation numbers (axis Y) depending on respective power indexes q (axis X), and fixed m meanings. The curves 1, 2, 3, 5 belong to the recursion (14), while the curve 4 belongs to ordinary fast convolution, when O is of the form (16). The values m = 2, 5, 9, 10 correspond to curves 1, 2, 3, 5, respectively. For example, the curve 5 corresponds to the total number of alterations m = 10, i.e. in each set of signal current samples there are only five samples that do not coincide with the respective previous samples. In such a case, operation O_1 varying values (curve 5) are bigger than O values (curve 4). On the other hand, O_1 that still assures eq. (19), could be for m less or equal to nine (curves 1 – 3 in Fig.2).

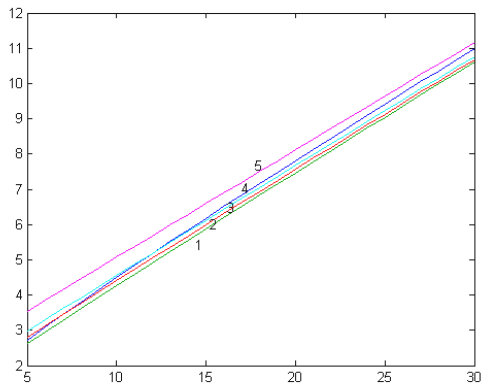

Figure 2. 10 logarithms of O and O_1 (axis Y) depending on power index q of base 2 (axis X).

7. Conclusion

A new recursive linear filtering approach, based on – the fast convolution, is developed for the wireless sensor networks. The number of CMADs needed for a speedy convolution calculation, can be significantly reduced by the original recursive algorithm (14), the number of operations of which tends from 0 to $3(N \log_2 N + N_{\text{products}} + N_{\text{adds}})$. It depends not only on value N, but also on the number of alteration of the current signal samples from their previous values. The larger N and the smaller m, the less operations are needed for recursive calculations (the sixth column in table), comparing with the fast convolution based on FFT (the fourth column in the table).

Note that we calculate all N+1 Fourier coefficients, using recursive expression (14), while, really, it is necessary to compute only N/2 +1 coefficients, because all samples in period, current and previous, are real-valued. This fact allows us to decrease calculating time sufficiently, and to increase the value of $m_{\max}$, on the contrary. Nevertheless, there exist such total number of alterations $m_{\max}$, for which nonequality (19) is not valid. In such a case, the ordinary fast convolution based on FFT will be preferable against the recursive relashionship (14). Thus, the results, given in Table 1 and Fig.2, show us that it is needed in sensor networks to implement the hybrid linear filtering schema, based on ordinary and simple recursive algorithms. Then, ordinary fast Fourier algorithm will generate output samples, when the total number of alterations is more or equal to $m_{\max}$, and the recursive one, on the contrary.

Acknowledgements

The authors thank the Vilnius Gediminas Technical University for financial support of their scientific researches of the project No 335 TMT 07T.

References

[1]. Duhamel, P., & Vetterli, M. (1990). Fast Fourier transforms: a tutorial review and a state of the art. *Signal processing*, *19*(4), 259-299.

[2]. Karas, P., & Svoboda, D. (2013). Algorithms for efficient computation of convolution. Chapter 8 in *Design and architectures for digital signal processing*, 179 - 208.

[3]. Gonzales, R.C., & Woods, R.E. (2002). *Digital image processing*, 2-nd Edition, Prentice Hall PTR.

[4]. Svoboda, D. (2011, September). Efficient computation of convolution of huge images. In *International Conference on Image Analysis and Processing* (pp. 453-462). Springer, Berlin, Heidelberg.

[5]. Battenberg, E., & Avizienis, R. (2011, September). Implementing real-time partitioned convolution algorithms on conventional operating systems. In *Proceedings of the 14th International Conference on Digital Audio Effects*. Paris, France.

[6]. Salomon, D. (2007). *Data Compression: The Complete Reference*. Springer.

[7]. Nguyen, T. V. (1994). Recursive convolution and discrete time domain simulation of lossy coupled transmission lines. *IEEE transactions on computer-aided design of integrated circuits and systems*, *13*(10), 1301-1305.

[8]. Deziel, J., P. (2000). *Applied introduction to digital signal processing*. Prentice Hall PTR.

[9]. Åström, K.J., & Eykhoff, P. (1971). System identification a survey. *Automatica*, *7*(2), 123 – 162.

[10]. Becker, M. (2014). *Services in Wireless Sensor Networks: Modelling and Optimisation for the Efficient Discovery of Services*. Springer Science & Business Media.

[11]. Proakis, J.G. & Manolakis, D.,G. (2008). *Digital signal processing, principles, algorithms, and applications.* Prentice Hall. New Jersey.

[12]. Pupeikis, R. (2015). Fast Fourier transform revisited. *Lietuvos matem. rink. Proc. LMS*, ser. A, 56, 113-118, doi:10.15338/LMR.A.2015.20.

[13]. Pupeikis, R. (2016, April). Revised fast 2D linear filtering. In *Electrical, Electronic and Information Sciences (eStream), 2016 Open Conference of* (pp. 1-5). IEEE.

[14]. Kazlauskas, K., & Pupeikis, R. (2016). *Advanced methods for short signal spectrum estimation*. Shaker. Aachen.

[15]. Hwang, J. K., & Li, Y. P. (2010). Efficient recursive IDFT scheme for complex-valued signals in tap-selective maximum-likelihood channel estimation. *Journal of Signal Processing Systems*, *60*(1), 71-80.