COMPUTATION OF INDIVIDUAL HARMONICS IN DIGITAL RELAYING

José A. Rosendo Macías
rosendo@esi.us.es

Antonio Gómez Expósito
age@us.es

Department of Electrical Engineering
University of Sevilla
Sevilla, Spain

Abstract - In this paper, a comparison of existing methods for harmonic calculation in digital relaying is presented. Pros and cons of recursive, approximate non-recursive and exact non-recursive algorithms are discussed. A compromise between speed and numerical robustness is obtained by a recently introduced modification of the pruned short-time discrete Fourier transform.

Keywords - Harmonic computation, spectral analysis, discrete Fourier transform, digital relaying.

1 INTRODUCTION

DIGITAL relays extract the information required to perform their function from the N waveform samples currently included in a sliding rectangular window [1]. Most times, one or several complex harmonics, or simply their magnitudes, are needed, for which the *discrete Fourier transform* (DFT) is the basic computational tool. This type of DFT is known in the literature as the short-time DFT [2],[3] but, also, as the moving-window, sliding or running DFT. When the full spectrum is required, the standard approach consists of using the well-known fast Fourier transform (FFT), whose complexity in the general case is $N \cdot \log_2 N$. However, the FFT is not competitive when a single or a few unrelated harmonics are needed, as in digital protections. In such cases, the DFT has been obtained traditionally by different methods: either directly from its definition or, indirectly, by previously computing less expensive transforms (e.g. Walsh, Haar or rectangular transforms), and then obtaining approximate Fourier harmonics [4], [5], [6], [7]. It is also possible to use recursive algorithms [8], [9], whose computational cost is independent of N in the moving-window case. However, the long-term response of these recursive filters may not be acceptable in certain cases.

In the last few years, the following improvements in the field of non-recursive algorithms have arisen which are not being used in digital relaying:

- There is a non-recursive way of computing the sliding FFT, whose complexity is O(N) [10].
- From the above method, a pruned algorithm can be implemented to obtain an individual running harmonic by means of O($\log_2 N$) additions and O($\log_2 N$) multiplications [11].
- Very recently, a new modification, based on the sliding DFT frequency-shifting property, has been proposed by which the cost of computing a single harmonic reduces to O($\log_2 N$) additions and only 2 complex multiplications [12].

In this paper, those recent developments will be reviewed and compared with classical approaches for computing running harmonics. The comparisons will be based on a window with $N = 16$ samples, assuming that only the fundamental harmonic is required.

2 DIRECT METHOD FOR HARMONIC CALCULATION

The short-time DFT of the N samples embraced by a sliding rectangular window at instant n can be expressed as [1],[2]:

$$X_n(k) = \sum_{i=0}^{N-1} x(i+n-N+1) e^{-j\frac{2\pi k}{N}i} \quad (1)$$

Denoting the vector of N DFT components as X_n, and the vector of the last N samples by x_n, yields:

$$X_n = G_F x_n \quad (2)$$

where the elements of matrix G_F are given by,

$$G_F(k,i) = e^{-j\frac{2\pi k}{N}i}$$

When the full spectrum is required, using (2) involves O(N^2) operations. A cheaper alternative is the so-called fast Fourier transform (FFT), whose complexity in the general case is $N \cdot M$ ($M = \log_2 N$). However, the FFT is not competitive when a single or a few individual harmonics are needed, as happens in digital relaying.

In such cases, each harmonic is obtained directly from (1) or by more elaborated algorithms, like that of Goertzel [13], requiring always O(N) computations.

3 APPROXIMATE NON-RECURSIVE ALGORITHMS

The DFT components can be also obtained indirectly, by previously computing less expensive transforms, and then obtaining approximate Fourier harmonics [6], [7]. The total number of operations required by any of these methods to obtain a generic harmonic is again of O(N).

3.1 DFT based on the Walsh transform

The Walsh transform can be expressed in matrix form as

$$W_n = G_W x_n$$

where, except for a scaling factor,

$$G_W(k,i) = \prod_{u=0}^{i-1} (-1)^{b_u(i) b_{i-1-u}(k)}$$

and $b_k(z)$ stands for the k-th bit of number z.

Computation of W_n is less expensive than that of the DFT, and its components can be subsequently used to obtain X_n from,

$$X_n = G_F x_n = G_F G_W^{-1} W_n$$

As the numerical values of the elements within each row of matrix $G_F G_W^{-1}$ greatly differ in size, approximate expressions can be obtained by neglecting the less significant ones. Denoting by $C_n(k)$ and $S_n(k)$ the real and imaginary components, respectively, of $X_n(k)$, the fundamental harmonic for $N = 16$ samples can be approximated by,

$$\begin{aligned} C_n(1) &= 0.125[W_n(1) + W_n(9)] + \\ &\quad +0.62 W_n(3) + 0.26 W_n(5) \\ S_n(1) &= -0.125[W_n(3) + W_n(11)] + \\ &\quad +0.628 W_n(1) - 0.26 W_n(7) \end{aligned}$$

3.2 DFT based on the Haar transform

The Haar transform is based on the set of rectangular and orthogonal Haar functions. In the time interval $0 \leq t \leq 1$, these functions can be expressed as [14],

$$\begin{aligned} H(0,t) &= 1 \quad 0 \leq t \leq 1 \\ H(1,t) &= \begin{cases} 1 & 0 \leq t \leq 1/2 \\ -1 & 1/2 \leq t \leq 1 \end{cases} \\ H(2^p+n,t) &= \begin{cases} \sqrt{2}^p & n/2^p \leq t \leq (n+1/2)/2^p \\ -\sqrt{2}^p & (n+1/2)/2^p \leq t \leq (n+1)/2^p \\ 0 & \text{elsewhere} \end{cases} \end{aligned}$$

where $p = 1, 2, \cdots$ and $n = 0, 1, 2, \cdots, 2^p - 1$.

Using those functions, except for the scaling factor, to build the rows of a matrix G_H, the Haar transform can be expressed in matrix form as,

$$H_n = G_H x_n$$

Neglecting small coefficients, as in the previous section, the DFT harmonics are approximated by

$$\begin{aligned} C_n(1) &= 0.628[H_n(2) - H_n(3)] + \\ &\quad +0.312[H_n(5) - H_n(7)] \\ S_n(1) &= 0.628 H_n(1) - 0.312[H_n(4) - H_n(6)] \end{aligned}$$

3.3 DFT based on the Rectangular transform

Real and imaginary parts of the rectangular transform are defined respectively by

$$C_R(k) = \sum_{i=0}^{N-1} x_i \mathrm{sgn}(\cos\theta k i)$$
$$S_R(k) = \sum_{i=0}^{N-1} x_i \mathrm{sgn}(\sin\theta k i)$$

where

$$\text{sgn(y)} = \begin{cases} y/|y| & y \neq 0 \\ 0 & y = 0 \end{cases}$$

Retaining only the most significant coefficients, the DFT fundamental harmonic components can be approximately obtained from:

$$\begin{aligned} C_n(1) &= 0.753 R_n(1) + 0.208 R_n(5) \\ S_n(1) &= 0.753 R_n(2) - 0.208 R_n(6) \end{aligned}$$

3.4 Custom FIR filters

The following approximate expressions have been also proposed in the literature [15] as a means of saving multiplications when computing the fundamental harmonic:

$$\begin{aligned} C_n(1) = \; & x(n-15) + x(n-14) + x(n-13) + x(n-12) \\ & -x(n-11) - x(n-10) - x(n-9) - x(n-8) \end{aligned}$$

$$
\begin{aligned}
& -x(n-7)-x(n-6)-x(n-5)-x(n-4)\\
& +x(n-3)+x(n-2)+x(n-1)+x(n)\\
S_n(1) = \; & x(n-15)+x(n-14)+x(n-13)+x(n-12)\\
& +x(n-11)+x(n-10)+x(n-9)+x(n-8)\\
& -x(n-7)-x(n-6)-x(n-5)-x(n-4)\\
& -x(n-3)-x(n-2)-x(n-1)-x(n)
\end{aligned}
$$

Similar expressions can be found for the 2nd and 5th harmonics.

4 EXACT NON-RECURSIVE ALGORITHMS

4.1 Fast short-time DFT

It has been shown that the short-time FFT can be obtained at the cost of O(N) operations by taking advantage of the similar contents of consecutive windows [10, 16]. This idea allows a single harmonic to be non-recursively and accurately computed by means of just O(M) mult/adds [11].

In this section, the O(N) short-time FFT is first reviewed by means of an example, and then the single harmonic case is discussed.

Consider the FFT butterflies for $N = 8$ at instants $n = 7$ and $n = 8$, shown respectively in figures 1 and 2.

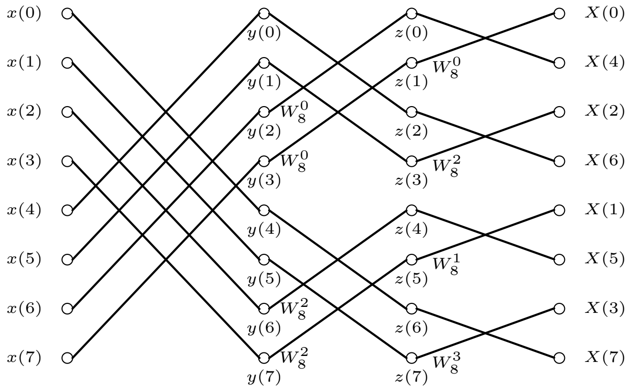

Figure 1: FFT butterfly structure for $N = 8$ at instant $n = 7$.

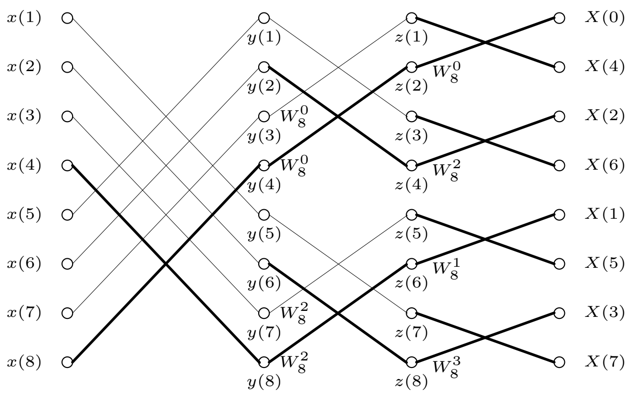

Figure 2: FFT applied to the same sequence of figure 1 one sample later ($n = 8$). Only butterflies shown in thicklines need re-evaluation.

When a new sample, in this case $x(8)$, enters the window, the oldest one, $x(0)$, is discarded. This is accomplished by merely shifting all the butterflies upwards, as shown in Figure 2. Note that, for convenience, all intermediate variables have been shifted in the same way as the input data.

Comparing both figures, by simple inspection, it can be concluded that many butterflies are simply shifted replicas of formerly computed butterflies. More specifically, only the lowest butterfly in the first layer, two butterflies in the second layer and four butterflies in the third layer are actually new. As a consequence, only 7 out of the 12 butterflies must be recomputed in the moving-window FFT case, provided certain intermediate variables are systematically stored.

Note that the set of updated butterflies forms a binary tree. For an arbitrary value of N, the number of such butterflies is clearly $N-1$, instead of the $N \cdot M/2$ that would be needed using a conventional in-place computation. Considering that each butterfly requires one complex multiplication and two complex additions, the total cost is then $N-1$ complex multiplications and $2N-2$ complex additions. These figures do not take advantage of the trivial values the twiddle factors take in certain butterflies (± 1, $\pm j$). Further savings are possible in the real-data case by exploiting the well-known symmetries implicit in the DFT. The price paid for the computational reduction is the extra memory required to store the $N \cdot M/2$ intermediate complex variables ($N/2$ per layer), in addition to the N harmonics.

From the above discussion it is also clear that only M half butterflies (one per layer) have to be updated when a single harmonic is needed, which involves M complex mult/adds in the general case. This set of butterflies constitutes a path in the tree linking the last sample with the desired harmonic. See [11] for further mathematical details.

4.2 Fast shifted short-time DFT

The technique discussed in this section is new in the power system arena.

As shown in [12], the short-time DFT satisfies the frequency-shifting property, which can be stated as follows: modulating a signal with the complex exponential in the discrete-time domain translates into a shift in the frequency domain plus a rotation.

Mathematically, let us define the following modulated sequence,

$$y(i) = x(i) \cdot e^{-j\frac{2\pi p}{N} i} \tag{3}$$

whose short-time DFT can be expressed as

$$Y_n(k) = e^{-j\frac{2\pi p}{N}(n+1)} \cdot \sum_{i=0}^{N-1} x(i+n-N+1)e^{-j\frac{2\pi(k+p)}{N}i} \tag{4}$$

The right-most term in the above equation can be now identified as the $(k+p)$-th harmonic of the original sequence, $x(i)$. Hence, the following relationship is obtained

$$Y_n(k) = e^{-j\frac{2\pi p}{N}(n+1)} \cdot X_n(k+p) \tag{5}$$

It is interesting to note that the harmonic rotation contained in (5) is a direct consequence of the window shift. This rotation disappears in the single window, finite-length sequence, leading to the well-known frequency-shifting property of the regular DFT.

The application of this property to reduce the cost of computing a generic harmonic is based on the observation that obtaining the zero-frequency harmonic by the fast procedure described in the former subsection requires no multiplications. This can be visualized in figure 2, by tracing the path leading to $X(0)$.

Therefore, the p-th harmonic of the input sequence, $x(i)$, can be obtained more efficiently from the zero-frequency harmonic of the modulated sequence, $y(i)$, by setting $k = 0$ in equation (5).

Formally, every time a new sample is available, the algorithm performs the following three steps:

1. Modulate the input signal

$$y(n) = x(n)e^{-j\frac{2\pi p}{N}n}$$

2. Update the path of butterflies required to obtain $Y_n(0)$.

3. Rotate the resulting harmonic $Y_n(0)$,

$$X_n(p) = e^{j\frac{2\pi p}{N}(n+1)} \cdot Y_n(0) \tag{6}$$

The computational cost of this algorithm reduces to M complex additions (step 2), two real multiplications (step 1) and a complex multiplication (step 3). Furthermore, since $|X_n(p)| = |Y_n(0)|$, the complex multiplication is not required if only the harmonic amplitude is of interest. This is the lowest computational effort reported up to date for a non-recursive algorithm.

As far as memory requirements is concerned, this algorithm needs N complex locations to store the intermediate variables.

5 RECURSIVE ALGORITHMS

The computational cost can be made independent of N in the moving-window case provided recursive algorithms are acceptable [8, 9, 17]. Note that despite these algorithms being recursive, they belong to the FIR class, like the other algorithms presented in the paper.

Two different categories are possible, depending on whether the time origin moves with the window or remains fixed:

5.1 Class #1: shifting time origin, constant coefficients

By evaluating (1) at two consecutive instants and substracting, the following algorithm can be obtained

$$X_n(k) = [X_{n-1}(k) + x(n) - x(n-N)]e^{-j\frac{2\pi k}{N}}$$

Assuming real data, the above complex expression can be written in terms of its real and imaginary parts as follows:

$$\begin{aligned} C_n(k) &= (C_{n-1}(k) + \Delta x_n)\cos\theta_k - S_{n-1}(k)\sin\theta_k \\ S_n(k) &= (C_{n-1}(k) + \Delta x_n)\sin\theta_k + S_{n-1}(k)\cos\theta_k \end{aligned}$$

where $\Delta x_n = x(n) - x(n-N)$.

5.2 Class #2: fixed time origin, varying coefficients

The DFT can be also defined with respect to a static time origin, leading to periodically varying coefficients:

$$X_n^F(k) = \sum_{i=0}^{N-1} x(n-N+1+i)e^{-j\frac{2\pi k}{N}(n-N+1+i)}$$

The transform, defined in this way, can be related to the standard definition by the expression

$$X_n^F(k) = X_n(k)e^{-j\theta_k(n-N+1)}$$

which means that fixed time origin harmonics are just rotated versions of its shifting time origin counterparts.

A recursive expression for this transform can be obtained as

$$X_n^F(k) = X_{n-1}^F(k) + [x(n) - x(n-N)]e^{-j\frac{2\pi k}{N}}$$

which is valid even for complex data [8].

For the real data case, the following expressions are obtained:

$$\begin{aligned} C_n(k) &= C_{n-1}(k) + \Delta x_n \cos(n\theta_k) \\ S_n(k) &= S_{n-1}(k) - \Delta x_n \sin(n\theta_k) \end{aligned}$$

Note that N memory locations are also needed by recursive algorithms to obtain Δx_n

6 COMPARISON

Digital relays frequently rely on the fundamental component of electric signals, so obtaining $X_n(1)$ is of main interest. For comparison purposes, the computational effort for obtaining the squared magnitude of the fundamental component will be considered:

$$|X_n(1)|^2 = C_n^2(1) + S_n^2(1)$$

Table 1 compares the computational cost of all the methods presented in this paper. All of them have been optimized by saving common partial sums as intermediate variables whenever possible.

Method	Mult.	Addit.	Total
Conventional DFT	8	21	29
Walsh-based DFT	8	47	55
Haar-based DFT	6	27	33
Rectan.-based DFT	6	42	48
Custom FIR filters	2	17	19
Fast DFT	7	11	18
Fast shifted DFT	4	9	13
Recursive #1	6	6	12
Recursive #2	4	4	8

Table 1: Number of operations required to compute the fundamental harmonic ($N = 16$).

It is worth noting that the four approximate ways of obtaining the DFT components (based on Walsh, Haar and rectangular transforms, as well as on custom FIR filters) have imperfect frequency response (see the appendix).

On the other hand, recursive algorithms can lead to unacceptable long-term errors. As explained in [10], the error variance in the output of such recursive filters grows linearly with time. Other undesirable non-linear effects have been described in the literature regarding the long-term response of recursive structures [2], [17]. Such long-term errors can be illustrated by means of figure 3, where the response of recursive algorithms #1 and #2 are compared to that of the conventional DFT when the input signal is

$$500(2 - e^{-n/500})\left[\sin(\tfrac{2\pi n}{16}) + 0.2\sin(\tfrac{4\pi n}{16}) + \right. \\ \left. +0.15\sin(\tfrac{10\pi n}{16})\right]$$

This anomalous behaviour suggests that recursive algorithms should not be recommended for protection unless some sort of refreshing strategy is carried out.

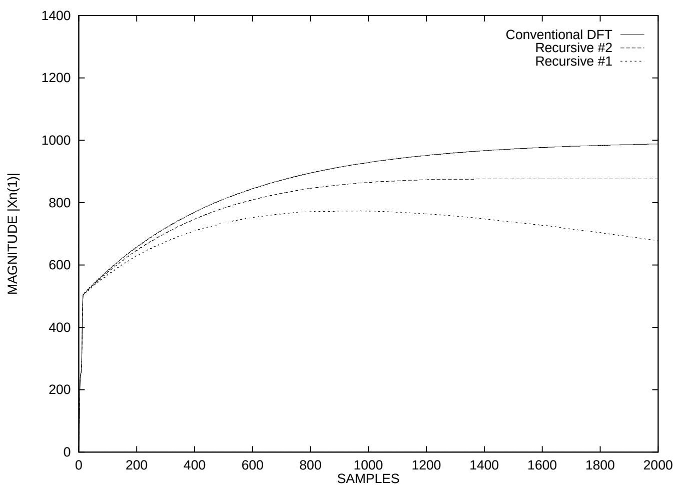

Figure 3: Long-term performance of recursive algorithms (16-bit integer arithmetic).

Since modern CPUs (RISC, Pentium, DSP) take approximately the same time to carry out one multiplication or one addition, the total number of arithmetic operations, rather than just multiplications, should be considered for comparison purposes. As can be seen, the exact non-recursive algorithm named fast shifted DFT compares well with all of the approximate techniques, including the customized FIR filters.

Regarding the recursive filters, it can be seen that the operation saving with respect to the fast shifted DFT method is not very significant for $N = 16$. Similar results can be obtained for larger windows, $N = 32$, $N = 64$, where the computational cost of this method increases only in 2 and 4 additions respectively.

Another important issue refers to the memory access and logical overhead which, depending on the processor architecture, could affect the above conclusions to a certain extent. Although both recursive and fast non-recursive methods need O(N) memory positions, the latter requires more memory accesses per sample. However, it is quite difficult to fully assess this factor, as it depends on compiler efficiency, amount of CPU registers, kind of cache memory, etc.

7 CONCLUSION

In this paper, a comparison of three groups of methods for harmonic calculation in digital relaying is provided. First, approximate non-recursive algorithms have been introduced, illustrating its non-exact frequency response.

A second group of exact non-recursive methods has them been presented. The most effective algorithm of this group has not been used so far in any power system related application.

The third class corresponds to recursive algorithms, which are slightly cheaper than the best non-recursive scheme. However, recursive structures are always prone to long-term output deviations.

It can be stated that non-recursive moving window DFT algorithms are currently the recommended choice for most real-time applications.

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support of the Spanish DGI under grant DPI2001-2367.

APPENDIX: FREQUENCY RESPONSE OF APPROXIMATE METHODS

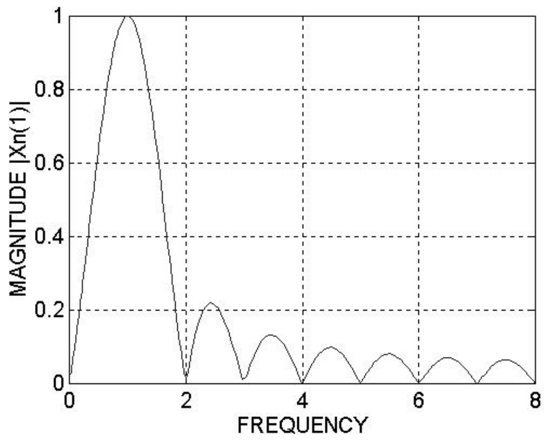

Figure 4: DFT spectrum for the fundamental component

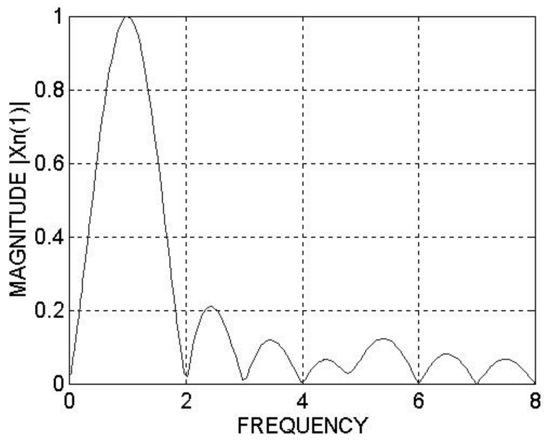

Figure 5: Walsh-based DFT spectrum for the fundamental component

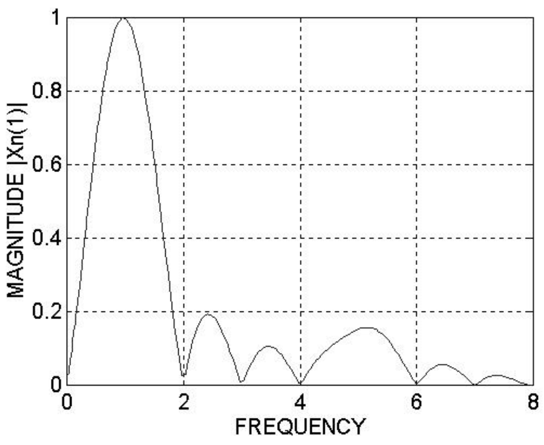

Figure 6: Haar-based DFT spectrum for the fundamental component

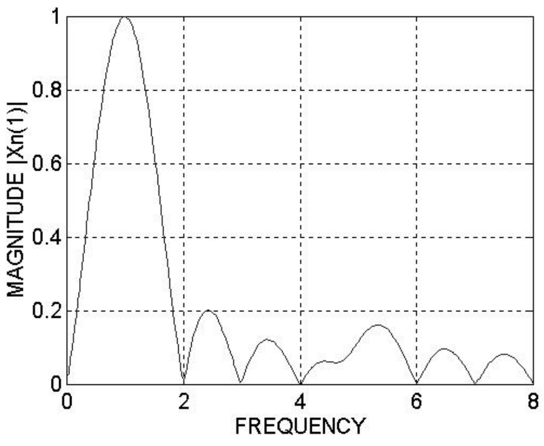

Figure 7: Rectangular-based DFT spectrum for the fundamental component

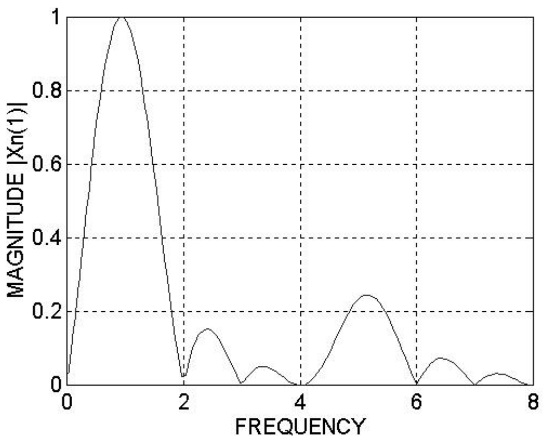

Figure 8: Custom FIR spectrum for the fundamental component

REFERENCES

[1] Phadke, Thorp, “Computer Relaying for Power Systems”, John Wiley & Sons Inc., 1988.

[2] A.V. Oppenheim, R.V. Schafer, “Discrete-Time Signal Processing”, New Jersey, Prentice Hall, 1989.

[3] J.S. Lim, A.V. Oppenheim, “Advanced Topics on Signal Processing”, New Jersey, Prentice Hall, 1988.

[4] M. A. Rahman, B. Jeyasurya, “A state-of-the-art review of transformer protection algorithms”, IEEE transac. on Power Delivery, Vol. 3, No. 2, pp. 534-544, April 1988.

[5] M. Habib, M. A. Marín, “A Comparative analysis of digital relaying algorithms for the differential protection of three phase transformers”, IEEE transac. on Power Systems, Vol. 3, No. 3, pp. 1378-1384, August 1988.

[6] B. Jeyasurya, M.A. Rahman, “Application of Walsh Functions for Microprocessor-Based Transformer Protection”, IEEE Trans. on Electromagnetic Compatibility, Vol. EMC-27, No. 4, November 1985.

[7] D B Fakruddin, K Parthasarathy, L Jenkins, “Application of Haar functions for transmission line transformer differential protecction”, Electrical Power & Energy Systems, Vol. 6, No. 3, pp. 169-180, 1984.

[8] J.H. Halberstein, “Recursive, complex Fourier analysis for real-time applications”, Proc. IEEE, Vol. 54, no. 6, p. 903, Jun. 1966.

[9] Jorge L. Aravena, “Recursive Moving Window DFT Algorithm”, IEEE Trans. on Computers, Vol. 39, No. 1, pp. 145-148, January 1990.

[10] M. Covell, J. Richardson, “A new, efficient structure for the short-time Fourier transform, with an application in code-division sonar imaging”, Proceedings, ICASSP’91 (Toronto), vol. 3, pp. 2041,2044.

[11] A. Gomez Expósito, J.A. Rosendo Macías, “Fast harmonic computation for digital relaying”, IEEE trans. on Power Delivery, vol. 14, No. 4, pp. 1263-1268, Oct. 1999.

[12] A. Gomez Expósito, J.A. Rosendo Macías, “Fast non-recursive computation of individual running harmonics”, IEEE trans. on Circuits and Systems –Part 2, vol. 47, No. 8, pp. 779-782, Aug. 2000.

[13] G. Goertzel, “An algorithm for the Evaluation of Finite Trigonometric Series”, Amer. Math. Monthly 65, pp. 34-35, 1968.

[14] R.C. González, R.E. Woods, “Digital Image Processing”, Addison-Wesley Publishing Company, 1992.

[15] R.R. Larson, A.J. Flechsig, E.O. Schweitzer, “An efficient inrush current detection algorithm for digital computer relay protection of transformers”, Paper A77-510-1, IEEE Power Engineering Society Summer Meeting, 1977.

[16] B. Farhang-Boroujeny, Y.C. Lim, “A comment on the computational complexity of sliding FFT”, IEEE trans. on Circuits and Systems – Part 2, vol. 39, pp. 875-876, Dec. 1992.

[17] A. Gómez Expósito, J.A. Rosendo Macías, J.L. Ruiz Macías, “Discrete Fourier Transform Computation for Digital Relaying”, Electric Power & Energy Systems, Vol. 16, No. 4, pp. 229-233, 1994.