Electrical Power Quality and Utilisation, Journal Vol. XIV, No. 2, 2008

A Hilbert Transform Based Algorithm for Detection of a Complex Envelope of a Power Grid Signals – an Implementation

Andrzej WETULA
AGH University of Science and Technology, POLAND

Summary: **The paper presents an algorithm for detection of a complex envelope of power grid signals. The algorithm is based on a Hilbert transform. It was prepared for an analysis of a low-frequency disturbances, but can also be used as a compex envelope source for other applications. An implementation of the algorithm in a digital signal processor was made to proof an ability of a real-time operation. Analyses with various numerical representations and input signal quantization were done. Execution times on a DSP were measured. Tests proved that the presented algorithm is able to analyze power grid signals in a real-time with satisfying performance and uncertainty.**

Key words:
power quality, measurement, Hilbert transformation, implementation, DSP

1. INTRODUCTION

Low-frequency voltage disturbances are considered a separate kind of voltage disturbances in a power grid. They are interpreted as a modulation, where the first harmonic is a carrier. Due to their effects on loads, the low-frequency voltage disturbances analysis is done with algorithms developed solely for this purpose. The most popular algorithm is the one used in a flickermeter [2]. However, many other algorithms are proposed, either for a disturbances demodulation and measurement [6, 4] or for their source detection [3]. Most of proposed algorithms neglect a signal phase modulation. This kind of a modulation is present in a power grid signals and is related to a grid structure itself. Phase variability of a power grid signals, especially voltage, is often very small. Despite that, in author's opinion, it should not be neglected as this simplifies a low-frequency disturbances model to a point at which it may be hard to determine their source. Also, an algorithm developed for low-frequency disturbances analysis may be useful for a load variation analysis if it is made versatile enough. Such an algorithm and its implementation is presented in this paper.

2. AN ALGORITHM

The described algorith was originally desinged to provide a complex envelope signal for a low-frequency disturbances analysis. Another application is a load and source impedance estimation of a Thevenin model of a power grid. Using the described algorithm as a source of complex envelopes of voltage and current, measurement of a load impedance can be done directly by a realisation of its definition. The algorithm was designed to meet following requirements:

— low uncertainty, even at a cost of a higher delay and numerical complexity,
— independence of any power grid models; only a signal model is considered,
— constant group delay to avoid phase problems,
— versatility, allowing to use an algorithm as an envelope source in various applications.

3.1. Model of a low-frequency disturbances as a complex modulation

An algorithm can't be designed without consideration of an analyzed signal model. In this case, a signal is a power grid voltage, or, less frequently, current featuring low-frequency disturbances. This kind of disturbances is mostly generated by a load changes, with a non-zero source impedance. A power grid model that can be used to illustrate a low-frequency analysis generation and propagation is shown in Figure 1.

This is an impedance-based model. As an impedance is defined for stationary systems only, using it to describe a non-stationary system causes errors. However, as shown in [5], such errors are not significant as long as parameters change slowly. In a presented model, $\underline{Z_t}$ and $\underline{u}$. are source impedance and source voltage, respectively. They are considered stationary. In order to use a complex (impedance) notation, a voltage has to be considered sinusoidal. In a case of a non-sinusoidal source voltage, a separate analysis for each harmonic can be made if necessary. $\underline{Z_l}$ represents a stationary (disturbed) load, and $\underline{Z_d}$ represents a non-stationary (disturbing) load. An equation relating a load voltage $u_l(t)$ with a disturbing impedance is:

$$\underline{u_l} = \underline{u_t}\,\frac{\underline{Z_d Z_l}}{\underline{Z_d Z_l} + \underline{Z_d Z_t} + \underline{Z_l Z_t}} \tag{1}$$

Above voltage depends on a source impedance value, and relation of disturbing to disturbed load value. A dependency

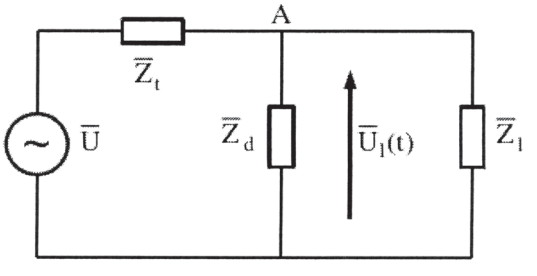


Fig. 1. A simplified power grid model used to illustrate low-frequency disturbances generation

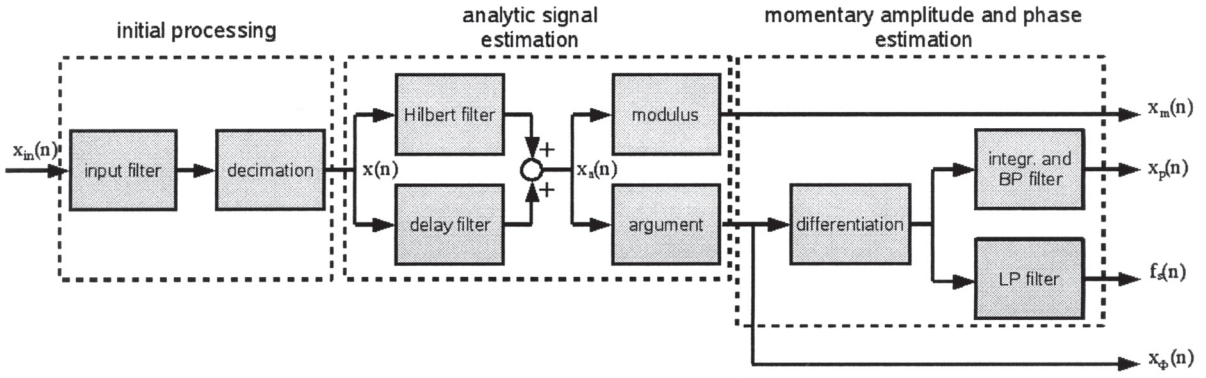


Fig. 2. Block schematic of the described algorithm

is non-linear, eg. sinusoidal load change will cause non-sinusoidal voltage change. Because all values in equation 1 are complex, low-frequency disturbances should be modelled as a complex modulation, not as an amplitude modulation. In fact, a pure amplitude modulation only happens in special cases. Based on an above model of a low-frequency disturbances generation, a model of a signal with such disturbances is:

$$x(t) = x_m(t)\sin\left(2\pi f_s(t)t + x_p(t)\right) \tag{2}$$

$$x_m(t) = X_m\left(1 + x_{mv}(t)\right) \tag{3}$$

In an above model a system frequency $f_s(t)$ is also considered variable. A phase modulation is very small, and can be easily masked by a system frequency change. Thus, a system frequency should also be considered variable, and a method of separating momentary phase and system frequency signals should be included in a demodulation algorithm.

3.2. Description of the algorithm

The algorithm is based on a Hilbert transform, used to calculate an anlaytic signal from an input signal. If an input signal has a properly defined frequency band, an analytic signal components can be considered orthogonal components of an input signal. An input signal band should be limited to a single harmonic surroundings. It's center is a frequency of an analyzed harmonic (usually 50 or 60 Hz), and it's width is defined by an assumed low-frequency disturbances band. To obtain a narrow-band signal from an input signal, a filter or set of filers is used. After calculating an analytic signal, its components are used to calculate an immediate amplitude and a generalized phase by transforming to a radial coordinates system. An immediate amplitude is an estimate of an input signal amplitude; generalized phase icludes two components. One is an integrate of a slowly-changing system frequency, and the other is a quickly-changing momentary phase. These two components differ by frequency bands, so to estimate them a low- and highpass filtering is needed. Recapitulating, the algorithm includes three parts: a signal preparation, an analytic signal estimation, and an estimation of complex envelope elements. To keep a constant group delay requirement, all filters used are finite impulse response (FIR) filters. A block schematic of the algorithm is shown in Fgure 2.

2.2.1. Signal preparation

The signal preparation part includes an input filter and a decimation. In the second part of the algorithm an analytical signal amplitude and phase, also called general phase, are estimated. A deriverative of an analytical signal phase, a momentary frequency, is ambigous in any case when input signal was not narrowband [7, 8]. In order to avoid such an ambiguoity of a momentary frequency, a processed signal needs to be filtered using a narrowband input filter. Using a downsampling allows the rest of the algorithm to work with a fixed sampling frequency. This simplifies design, as for any given input sampling rate only a signal preparation part should be redesigned. In a case of a power grid signal, everything but a single harmonic and its surroundings are filtered out. The base harmonic is a most obvious signal component to analyze, although analysis of higher harmonics may yield iteresting results. Width of an input filter band is determined by an assumed band of a low-frequency disturbances. Such a band is defined in [2] as 50±35 Hz, or 15–85 Hz. This band is related to a flickermeter design with its corresponding load model. The described algorithm should be load model independent; however a low-frequency disturbances band is not precisely defined in any other way than in [2]. For this reason abovementioned frequency range wa choosen for an input filter.

An input filter design is sampling-frequency dependent. For this reason, a soultion presented below should only be considered an example. An algorithm is designed as an element of a power quality analyzer device. Common sampling frequencies in such devices are around 10–12 kHz. Designing FIR filter for a sampling frequency within this range with a cut-off frequency of around 100 Hz and an acceptable steepness of transition band yields an extremely long impulse response of around 2 thousand samples. It is better to divide an input filter and downsampling blocks into a series of consecutive filters and downasamplings by a lower factor. An example input block for a 10 kHz sampling frequency includes two filters. After each filter a signal is downsampled by 5. First filter is a 200 samples long, and a second one is 280 samples long. As a sampling frequency of

the second filter is 5 times lower than the first, it introduces most of an input block group delay, and is much stepper than the first filter. Both filters were designed using a window method, and for both of them a Blackman window was used for its low passband errors coupled with simplicity and good sidelobe attenuation.

2.2.2. Analytical signal estimation

An analytical signal is estimated in a time domain. A FIR Hilbert transformer is used to obtain an imaginary part of a signal, and a delay filter is used instead of a buffer delay to obtain a real part. Using filters in both signal paths allows an extra bandpass filtering of an analytical signal, increasing damping of an unwanted higher frequency components. Filters were designed to be type IV (Hilbert transformer) and type II (delay filter) FIR filters, with an impulse response length of 160 samples. Output signals of these filters are an analytical signal components; for a power grid signals they are orthogonal components. Before obtaining a complex envelope signal, a coordinate system need to be changed from cartesian to polar:

$$x_m(n) = \sqrt{x_r(n)^2 + x_i(n)^2} \tag{4}$$

$$x_{\ddot{o}}(n) = \arctan\left(\frac{x_i(n)}{x_r(n)}\right) \tag{5}$$

Where $x_m(n)$ is a momentary amplitude, $x_\Phi(n)$ is a general phase, and $x_r(n)$ and $x_i(n)$ are real and imaginary analytical signal components. It should be noted that above equations introduce a nonlinearity into an algorithm.

2.2.3. Complex envelope estimation

A complex envelope is composed of a momentary amplitude and a momentary phase. A momentary amplitude is an amplitude $x_m(n)$ of an analyitcal signal. A general phase of this signal includes two components: an integral of a system frequency and a momentary phase. To separate these signals, a frequency band separation is used. A system frequency changes are related to a regulation processes in power plants, and a phase changes are caused mostly by load variation. Thus, system frequency occupies low-frequency band of a momentary frequency (a general phase deriverative), while phase changes occupy higher-frequency band. A band separation algorithm includes a differentiate estimator and two filtering blocks. A differential is estimated using finite difference method, which proves precise enough in an analyzed 0–35 Hz band, while providing smallest possible group delay. An obtained momentary frequency signal is then split into two filters: one for a momentary phase estimation, and one for a system frequency estimation.

A system frequency signal is not needed for load variability analysis, nor for an impedance estimation, and it's block is optional. However, a system frequency is a good indicator of a global state of a power system, as shown in [12]. Designing a system frequency filter may be done by assuming a definition from [1]. This standard defines a system frequency as a tenth of a number of whole cycles in a ten second long part of an analyzed signal. It is an equivalent of a 10 second long running average of a momentary frequency. A running average can be also interpreted as a filtering with a rectangular time window. Thus, it provides all advantages and drawbacks of such a window. The main lobe is the narrowest possible, at a cost of a low sidelobe attenuation. This may cause a significant leak of a high-frequency components into a system frequency signal. A possible solution is replacing rectangural time window with another time window. A non-rectangular time window provides much higher sidelobe attenuation compared to rectangular window, at a cost of widening the main lobe. Overall, a system frequency estimation block includes two filters with a downsampling by 10 between them. First filter is a 120 samples long lowpass filter, providing around 100 dB attenuation at 20 Hz. Second filter, 10 s or 400 samples long, is a hiperbolic time window as defined in [9]:

$$w_{h(n)} = \left(1 - 2.25\tanh\left(0.18\pi n\right)^2\right)^4 \tag{6}$$

Such a window has a main sidelobe 1.6 times wider than a rectangular window, with a sidelobe attenuation 2.3 times higer. Using it results in a system frequency band wider than the one derived from a definition in [1], but allows for a much better band separation between system frequency and a momentary phase.

A momentary frequency signal is a differential of a phase signal. Thus, to obtain a momentary phase signal one must not only correctly separate its band, but also reverse a differentiation operation. A band of a momentary phase is not directly defined in any standard, so it can be defined freely by an algorithm author. The most obvious approach is using a frequency range defined in a flickermeter standard [2] as 0.05–35 Hz (modulation frequency). The higher cut-off frequency is the same as a cut-off frequency of an input and Hilbert filters. The lower cut-off frequency is much lower than the system frequency band boundary presented above. As a system frequency band cannot be made much narrower without using a longer time window, a lower cut-off frequency of a momentary phase needs to be raised to over 0.2 Hz. Overall, path including a differentiator and a momentary frequency filter should have a gain of 1 between 0.2 and 35 Hz. A steep lower transition band is needed to avoid a system frequency leaks into a momentary phase signal. Considering an above requirements, a momentary frequency filter was designed using a frequency sampling method [13]. A desired amplitude-frequency response of a momentary frequency filter was set to a reciprocal of a differentiator response in a desired passband and to zero at the other frequencies. A type IV FIR filter was used in order to compensate for a $\pi/2$ phase shift of a differentiator.

3. AN IMPLEMENTATION

An algorithm was implemented in order to show it's usefullness in an on-line measurement equipment. To find out if a real-time operation in an embedded device is possible, following questions need to be answered:

1. Is the algorithm not too complex for state-of-the-art embedded systems?
2. Will it work using a single-precision floating point numbers ?

In order to answer above quaestions, the algorithm was implemented on a TMS320C6713 digital signal processor (DSP) from Texas Instruments. It is a 32-bit floating-point DSP, equipped with two serial ports, universal memory controller and a multi-channel DMA controller. A DSP used in an implementation was a part of a DSK6713 starter kit, which featured 16 MB of a random access memory, 2 MB of a flash memory for a non-volatile storage, an audio codec, an expansion connectors, and a USB debug probe. A Code Composer Studio development environment was provided with a starter-kit. It allowed for a mixed C/embeddec C++ and assembly programming and provided a real-time kernel and libraries with a set of drivers for DSP peripherials.

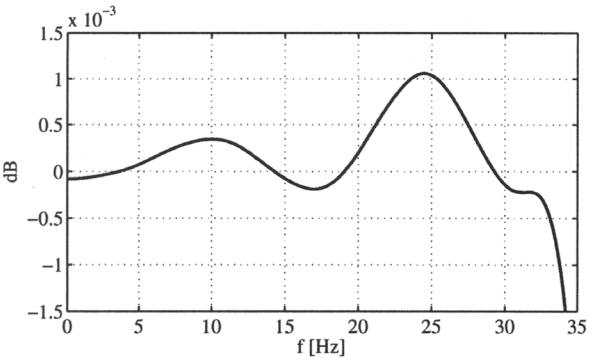
An implementation was made in a C language, using an authomatic code optimizations and functions from an optimized mathematical library, called fastRTS. It could be made faster by optimizing code manually and rewriting at least part of it in an assembly language, but it would require a longer development time. As only an amount of needed computational throughoutput and rounding errors were to be tested, an implementation didn't use an A/D converter. Instead, an input signal was read from and results were put into an on-board memory.

3.1. Numerical representation errors

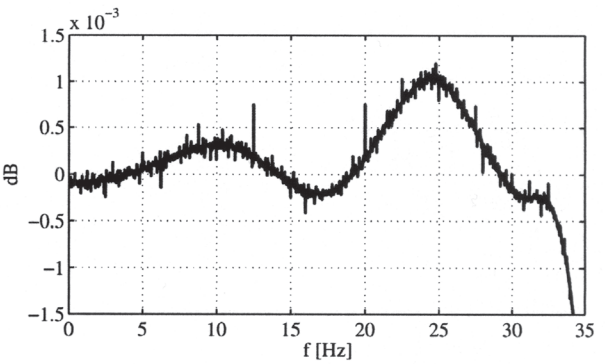
In a theoretical description, an algorithm operates on real numbers. However, digital computers use fractional approximations of real numbers to represent data. This causes rounding errors, depending on chosen word length and representation. An algorithm was first implemented in Matlab, which uses 64-bit floating point representation, also called double precision representation. Floating-point digital signal processors, including TMS320C6713, use 32-bit representation, called single precision representation. Thus, an errors caused by limiting word length should be calculated. This can be done by comparing results of analysis carried out using different numerical representations. Only floating-point representations were tested. It should be noted that signals acquired using an A/D converter are seldom represented by more than 16-bit fixed-point numbers, even if an algorithm itself works with floating-point numbers. Thus, an algorithm should also be tested for sensitivity to an input signal quantization. In order to compare an algorithm uncertainty with different numerical representations, two kinds of model experiments were conducted. First one included calculating frequency responses of an amplitude and phase demodulation. Such responses show attenuation of a sinusoidal modulation signal as a function of a modulation frequency, with carrier frequency and modulation depths being parameters. They allow to estimate maximum demodulation uncertainty for a sinusoidal modulation. Because of a wideband character of a power grid signal modulation, this uncertainty is only a rough estimate of a total uncertainty of the algorithm. However, it is suffcient as a measure of the algorithm performance with different numerical representations. Second experiment included analysing test signal with an algorithm implemented in a Matlab environment and in hardware, and presenting differences between obtained signals. This was done to proof a possibility of a DSP implementation and to illustrate an amount of uncertainty introduced by such an implementation.

3.1.1. Frequency responses for 32- and 64-bit representation

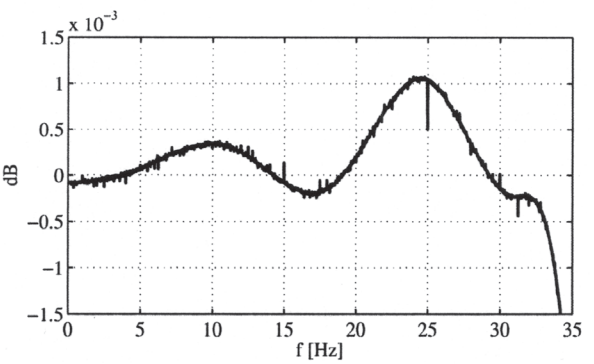
Frequency responses of a momentary amplitude, momentary phase and system frequency estimators were calculated for a 32-bit and 64-bit floating-point representations. Separately, responses were calculated for 64-bit floating-point algorithm anaysing an input signal quantized with a 16-bit quantizer. Differences between 64- and 32-bit frequency responses for phase and amplitude estimators are of an order of 10−3 dB, lower than passband error of calculated frequency responses. This leads to a



(a) 64-bit floating point



(b) 32-bit floating point



(c) 64-bit floating point with 16-bit quantized input signal

Fig. 3. Frequency responses of a momentary amplitude estimation algorithm for various numerical representations

conclusion that 32-bit numerical representation is not introducing significant uncertainty. Similar situation occurs for an input signal quantized with a 16-bit quantizer, with differences being slightly less, but of the same order of magnitude as in 32-bit representation. Example frequency responses for an amplitude estimation path are presented in Figure 3.

3.1.2 A test signals analysis

Second expreiment included analysis of a test signal with the algorithm implemented in a DSP and comparing obtained results with ones calculated with the same algoritm in Matlab. Both implementations were using 32-bit floating point numbers, and a test signal was:

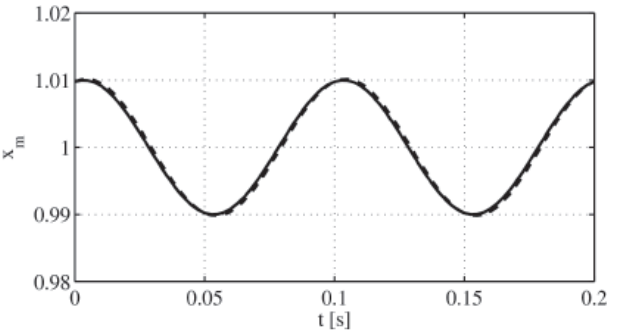
$$y(n) = K_m(n)\sin\left(2\pi\frac{50}{f_s}n + \frac{\pi}{180}\cos\left(2\pi\frac{10}{f_s}n\right)\right) \quad (7)$$

$$K_m(n) = \left(1 + 0.01\sin\left(2\pi\frac{10}{f_s}n\right)\right) \quad (8)$$

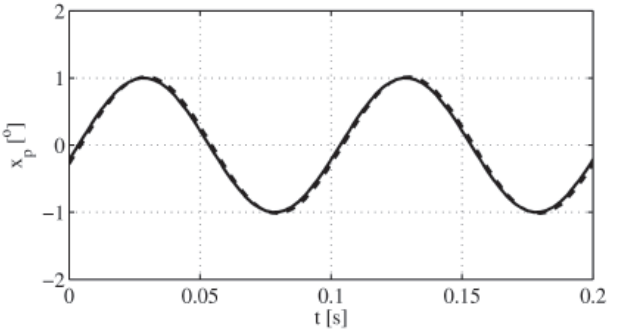
with a sampling rate f_s of 10 kHz. Obtained amplitude and phase envelopes are shown in Fgure 4. A system frequency signal was also calculated. Most of differences between results obtained from a DSP and Matlab implementation are a result of a small phase shift in a DSP implementation. It is probably caused by a different rounding of filter coeffcients. Phase shift is similar for phase and amplitude signals, and without it differences between signals are around 1% for a momentary amplitude and around 2% for a momentary phase. Differences for a system frequency are around 0.0001 Hz for a 50 Hz, which makes 0.0002%. Obtained results show a possibility of a DSP implementation of the described algorithm.

3.2. Computational complexity

To allow a real-time operation, an implemented algorithm needs to perform analysis for each input signal sample in time less than a sampling time. A too complex algorithm is unable to perform a real-time analysis on embedded devices, even if it was able to work off-line with pre-registered signals. In order to calculate a numerical complexity, a number of operations needed for a single iteration was calculated. An algorithm includes only digital FIR filters and a coordinate change. A FIR filter implemented in time domain requires one multiply and accumulate per sample per coeffcient plus data load/store operations. Amount of processor cycles needed for a specific FIR implementation depends on a processor used and code optimizations. For example, a TMS320C67x DSP Library functions [11] require a number of cycles roughly proportional to half of a number of filter coeffcients per input sample. However, a mentioned library is written in a highly optimized assembly language, and able to perform only block filtering. With an unoptimized FIR filter a number of cycles needed may be even a few times higher. A coordinate change requires one square root, two multiplications, one sum, one division and an arcus tangent estimation per an algorithm iteration. It's time complexity depends on a chosen estimation



(a) amplitude envelope



(b) phase envelope

Fig. 4. Amplitude and phase envelopes of a test signal demodulated with a Matlab implementation (continous line) and with a DSP implementation (dashed line) of the described algorithm

algorithms and processor capabilities. A fastRTS library functions for TMS320C67x [10] take around 50 cycles for square root and around 100 cycles for arcus tangent using a single precision representation. It should be noted that an exact amount of calculations needed depends on input values. Concluding, evaluation of a precise amount of calculation needed for a single algorithm iteration is theoretiaclly simple. Practically, a precise amount of computing power needed highly depends on code optimization and number of additional operations, such as memory load/store, branches and task switching. Thus, a theoretical calculations should be only used for a rough estimation of a complexity of an algorithm implemented in a certain hardware. They can be used to find out if there is a possibility of a real-time operation, and how much of code optimization may be needed. An experiment conducted either with a real device or a cycle-accurate simulator is a more reliable method of verifying an algorithm.

3.2.1. Theoretical complexity of the described algorithm

If all output signals are needed, the algorithm includes total of seven FIR filters working with a total of four sampling frequencies, a deriverative estimator and a coordinate change operation. For a presented implementation, sampling frequencies are 10000, 2000, 400 and 40 Hz. Filters lengths vary between 120 and 1000 samples. A total number of multiply and accumulate operations needed is 3152000 per second, with 1 or 2 such operations per cycle for the used DSP. To siplify, one multiply and accumulate operation per cycle was counted, overrating a number of cycles needed. An

amount of calculations required for a coordinate change and a differentiation should be added to a number of calculations needed for filters. It increases a total number of cycles per second to over 3.2 milion. Theoretically, the algorithm should be easily implementable in a DSP capable of around 1000 milion operations per second. However, a calculated number is an amount of operations needed solely for calculations. An input signal is read from a dynamic RAM, which requires a large number of cycles per each read or write. Also, organizing circular buffers and controlling program execution causes a large overhead.

3.2.2. DSP load analysis

A cycle-accurate simulator was not provided with the starter kit used for an implementation. This left only one option of measuring an algorithm execution speed: executing it in a hardware and measuring an analysis time. An analysis time of a signal of a certain length was measured, followed by measuring time of a single algorithm iteration. A tested signal was 100 thousand samples long, an equivalent to seconds at 10 kHz sampling frequency. Obtained results are presented in Table 1. Even without opitmizations the algorithm is able to perform a real-time analysis, as analyzing a 10 second long signal takes slightly less than ten seconds. An automatic code optimization provides a large gain in effectiveness, reducing an analysis time by over a half. Comparing a measured and theoretical execution speed shows a possibility of making an implementation much faster by manual code optimizations. It should be noted, however, that only an analysis of a few signals in parallel is possible. As power quality analysis for a three-phase system requires analysing of a six signals, the used DSP does not seem sufficient.

4. CONCLUSION

Presented results allow to conclude that the described algorithm is able to operate in a real-time in an embedded system. Uncertainties introduced by a numerical representation and an input signal quantization are not significant. An algorithm executes properly on a floating-point DSP. However, analysis of a single input signal took a significant amount of a DSP resources. For an impedance estimation, a two signals need to be analyzed, and a three-phase system requires parallel analysis of six signals. An obtained complex envelope signal is often used for a further analysis, which take an additional resources. For this reason, a DSP used is not suffcient for a multi-channel implementation. Such an analysis can be made possible by using an embedded PC platform or a multiprocessor DSP system. Implementing the algorithm to work in a fixed-point device would allow to execute it on a faster, fixed-point DSP or in a FPGA device.

Table 1. Execution times of a non-optimized and automatically optimized algorithm in a TMS320C6713 DSP running at 225 MHz

Optimized	One cycle	10 s single
no	2.43 ms	9.7 s
auto	1.089 ms	4.32 s

4.1. Further research

Planned further research include:

— implementing the algorithm for a multi-channel analysis,
— a fixed-point implementation of the algorithm,
— using the algorithm in a distributed measurement system for disturbances propagation analysis.

Of the above, implementing an algorithm in a distributed system is the most interesting, though the most complicated one. In a distributed measurement system for power quality analysis multiple data acquisition systems with very precise measurement synchronization are required. Building such system with a careful choice of measurement points may allow for a detection of low-frequency disturbances sources.

REFERENCES

1. IEC standard: Electromagnetic compatibility (emc), part 4–30: Testing and measurement techniques — power quality measurement methods, basic emc publication, 2000.
2. IEC Standard: Electromagnetic compatibility (emc) — part 4: Testing and measurement techniques — section 15: Flickermeter - functional and design specifications, 2003.
3. Axelberg P.G.V., Bollen M.H.J.: *An algorithm for determining the direction to a flicker source*. IEEE Transactions on Power Delivery, 2006, 21(2):755–760.
4. Bień A., Duda K., Szyper M., Wetula A., Zieliński T.P., Rozkrut A.: *The new measure of low-frequency energy disturbances in power system*. Metrology and Measurement Systems, 2005, XI(2).
5. Bień A., Szyper M., Wetula A.: *Model studies on signals measured with light flicker severity meter* (in polish). Proceedings of XV Symposium on Modelling and Simulation of Measurement Systems, pages 193–201, Krynica, Poland, september 2005.
6. Feilat E.A.: *Detection of voltage envelope using prony analysis — hilbert transform method*. IEEE Transactions on Power Delivery, 2006, 21(4): 2091–2093.
7. Gabor D.: *Theory of communication*. Procedings of the IEE, 1946, 93: 429–457.
8. Huang N.E., Shen Z., Long S.R., Wu M.C., Shih H.H., Zheng Q., Yen N., Chao Tung C., Liu H.H.: *The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis*. Proc. of the Royal Soc. London A, 1998, 454: 903–995.
9. Szyper M.: *New time domain windows*. Electronics Letters, 1995, 31(9).
10. Texas Instruments Inc., TMS320C67x FastRTS Library Programmer's Reference (spru100a.pdf), 2002.
11. Texas Instruments Inc., TMS320C67x DSP Library Programmer's Reference (spru657b.pdf), 2006.
12. Welfonder E.: *Least-cost dynamic interaction of power plants and power systems*. Control Eng. Practice, 1997, 5(9): 1203–1216.
13. Zieliński T.P.: *Od teorii do cyfrowego przetwarzania sygnałów*. 2002, Publisher: Nakładem Wydziału EAIiE AGH, Kraków.

Andrzej Wetula, Ph.D.
is a research and teaching assistant in a Department of Measurement and Instrumentation, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, AGH University of Science and Technology, Krakow, Poland. His main research interest is analysis of a low-frequency disturbances in a power grid. Other research interests include distributed measurement systems.
e-mail address: wetula@agh.edu.pl
research team web page: www.poweragh.xt.pl.