2

A Proposed Model-Based Adaptive System for DFT Coefficients Estimation Using SIMULINK

Omar Mustaf
Gulf University
Kingdom of Bahrain

1. Introduction

Spectral Analysis is of great importance in signal processing applications in general, and in the analysis and the performance evaluation of communications systems specifically. On the other hand the procedure of calculating/estimating the spectrum itself is also important, i.e. whether it's simple or complicated, especially when taking into account the limitations of onboard space in the parallel computation and VLSI implementation. Therefore, in response to the above, Widrow et al. proposed an adaptive method for estimating the frequency content of a signal through demonstrating a relationship between the Discrete Fourier Transform (DFT) and the Least Mean Square (LMS) algorithm, where the DFT coefficients are estimated by a new means using the LMS algorithm (Widrow et al., 1987). That was the original attempt of relating the DFT to the LMS adaptation rule. The main features of such a spectrum analysis are simplicity, adaptability, and suitability with parallel computations and VLSI implementation. This is owing to the nature of the LMS algorithm, which lends itself to this type of implementation.

Later on, Mccgee showed that Widrow's spectrum analyzer could be used as a recursive estimator for the sake of solving the exponentially-weighted least squares estimation and a filter bank model was deduced. The fundamental outcome of that work was that the LMS algorithm could act as a bank of filters with two modes of operation, which means when the LMS learning rate is chosen to be ½, the filter poles are located at the origin. This means equivalently that the LMS effective transfer functions are FIR filters; otherwise, the equivalent filter is IIR (Mccgee, 1989).

The Widrow's principal relation between the LMS and the DFT was then extended to the 2-Dimentional (2-D) case by Liu and Bruton, which directly resulted in the 2-D LMS spectrum analyzer. Here it was shown that the 2-D LMS algorithm has the advantage that allows concurrent computations of what was called an updating matrix and therefore the potential for very fast parallel computations of the 2-D DFT; however, there was no statement about how fast it was (Liu and Bruton, 1993).

Two years later, a generalization of the above mentioned 2-D LMS spectrum analyzer was demonstrated by (Ogunfunmi and Au, 1995). It was achieved by successfully extending the relation to other 2-D discrete orthogonal transforms. The simulations of that work showed the same results when imposing a frame of a test signal of size 32 by 32 to both a 2-D DFT

LMS based algorithm and a 2-D spectrum analyzer for the 2-D discrete cosine transform. The same results coincidence was demonstrated when comparing the estimated spectrum of the 2-D DFT LMS based with that of the 2-D discrete Hartly transform (DHT).

Beaufays and Widrow came back in 1995 to compare the LMS spectrum analyzer with the straightforward, non-adaptive implementation of the recursive DFT, and the robustness of the LMS spectrum analyzer to the propagation of round-off errors was demonstrated. Also, they showed that this property is not shared by other recursive DFT algorithms (Beaufays and Widrow, 1995).

In 1999 Alvarez et al. proposed the analog version of the LMS spectrum analyzer, where the coefficients are adapted independently by analog LMS structure which can be implemented in the VLSI technology. The main advantage gained from the work is that since the adaptation is carried out in the continuous time domain, real time computing speed was achieved (Alvarez et al., 1999).

This chapter aims mainly to propose a model-based simulation design procedure for realizing the relationship between the DFT and the LMS adaptation algorithm, using a very powerful simulation tool, namely SIMULINK, which runs under the MATLAB package. The proposed design is supposed to perform a spectral estimation using Widrow's adaptive LMS spectrum analyzer (Widrow et al., 1987). Therefore, this work attempts to provide a design procedure for that theoretical work. Specifically, this chapter proposes a model-based design procedure with simulation results to show the importance of this relation, which can be considered as an adaptive spectrum analyzer with wide range of applications in the design of modern digital communications transceivers and specifically the transform domain equalizers.

The simulation methodology is based on the SIMULINK environment and it adopts a block by block SIMULINK design procedure by which each *system unit* (system unit means, for example, phasor generator unit, LMS adaptation process unit) of the designed model is created at the block level and then assembled together properly to form the intended system model of Widrow's adaptive spectrum analyzer model.

After finalizing the design, test signals will be imposed to the model for verifying the validity of the DFT coefficients that are estimated adaptively using the LMS algorithm. The simulation results will be showed and discussed clearly in the simulation results discussion section. In addition, new frequency contents will be added to the test input signal during the running mode to test the property of adaptability to sudden changes in the input signal, thus seeing how the system will follow up these changes.

As a future work, a generalized SIMULINK model design procedure for any specified *n-point* spectral estimation is suggested as a further development of this work. Also, there will be a suggestion of how to use the proposed model in the design and simulation of frequency domain equalizers as an application.

Finally, the chapter includes a brief introduction to using SIMULINK, which is important, especially for those who do not have access to or have limited information about SIMULINK. It acts as a *clear*, *easy*, *short and concentrated guide* to understand and/or develop the work presented in this chapter.

2. The mathematical background of the Adaptive Spectrum Analyzer

In this section the mathematical model of the adaptive spectrum analyzer, which was proposed first by Widrow et al. (Widrow et al., 1987), is presented. The main theory and the useful result of that work is given in this section without going further into the detailed mathematical derivations and proof, as it is out of the scope of this chapter. The objective of the chapter is to use this demonstration to propose a SIMULINK model that is corresponding to the adaptive LMS spectrum analyzer. Also, the proposed simulation model might be adopted later as a basic building block in the design of adaptive two-layer structures for fast filtering as it will be illustrated in the future work section at the end of the chapter. The Widrow et al. model will be abbreviated as WASA throughout the rest of the chapter, which stands for Widrow Adaptive Spectrum Analyzer.

The key idea of WASA is that for an arbitrary signal to be analyzed in the frequency domain, it's possible to choose a number of phasors, according to a certain criterion, and then weight these phasors with adaptable weights that might be adapted by iterating the well-known LMS adaptation role. Upon choosing a suitable value for the adaptation rate, these weighted phasors are equal to the DFT coefficients of the signal under consideration. Fig. 1 shows the complete schematic diagram of the WASA model.

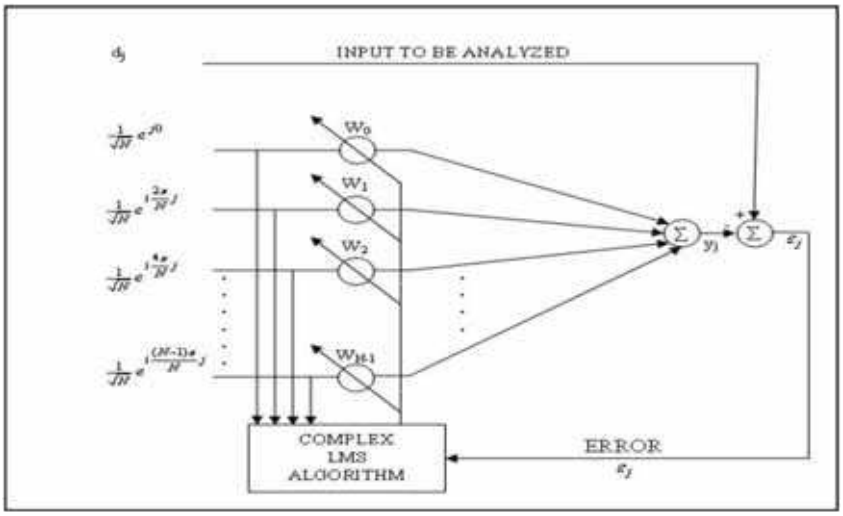


Fig. 1. Adaptive LMS spectrum Analyzer as proposed by (Widrow, et al., 1987)

The signal d_j is the input sampled version of the signal to be Fourier analyzed which is sampled at each time index j. The sampling time is T sec and the corresponding sampling frequency is $\Omega = 2\pi/T$ rad/sec. It's the objective of WASA to resolve d_j to its corresponding DFT coefficients. The left hand side of Fig. 1 shows a set of complex exponentials, which represent the time domain phasors that are supposed to equal the DFT coefficients of d_j after weighting them with an adaptable weight vector. There are N phasors, where N refers to the desired number of the DFT coefficients. The corresponding frequencies of these set of phasors span the frequency range from DC up to the sampling frequency Ω, including the

fundamental frequency Ω/N. The fundamental phasor can be expressed in terms of the time index j as:

$$e^{i\frac{\Omega}{N}jT} \tag{1}$$

Where $i=\sqrt{-1}$. Now, since $\Omega.T = 2\pi$, then this phasor becomes:

$$e^{i\frac{2\pi}{N}j} \tag{2}$$

The rest of the other phasors are the power of equation (2).

In vector form these phasors might be written as:

$$X_j \cong \frac{1}{\sqrt{N}}\begin{bmatrix} 1 \\ e^{i\frac{2\pi}{N}j} \\ e^{i\frac{4\pi}{N}j} \\ . \\ . \\ e^{i\frac{2\pi(N-1)}{N}j} \end{bmatrix} \tag{3}$$

These phasors are to be weighted at each sampling time index j by the adaptable weight vector W_j, where

$$W_j \cong \begin{bmatrix} w_{j0} \\ w_{j1} \\ w_{j2} \\ . \\ . \\ w_{jN-1} \end{bmatrix} \tag{4}$$

The factor $\frac{1}{\sqrt{N}}$ is a normalization factor and it's used to simplify the analysis of the system of Fig. 1, because it pulls the power of the phasors' vector, $\overline{X}_j$ to unity, as it is shown in the main literature of (Widrow, et al, 1987). The weight vector components w_{j0}, w_{j1},…., w_{jN-1} are updated at each sampling time index j in accordance to the well-known LMS adaptation rule,

$$W_{j+1} = W_j + 2\mu\varepsilon_j\overline{X}_j \tag{5}$$

Where

$$\varepsilon_j = d_j - y_j \tag{6}$$

is the instantaneous error between the input signal d_j and the spectrum analyzer output, y_j which is given by:

$$y_j = X_j{}^T W_j \tag{7}$$

whereas, μ represents the LMS adaptation rate (speed) and $\overline{Xj}$ is the complex conjugate of X_j. (Widrow, et al., 1987) concluded that when the learning speed is set to 0.5, then the output of the spectrum analyzer of Fig.1 is equal to the DFT of the input signal d_{j-1}. Referring to Fig. 1, the spectrum analyzer output is:

$$\begin{pmatrix} LMS \\ spectrum \\ analyzer \\ output \\ vector \end{pmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & & & & & \\ & e^{i\frac{2\pi}{N}j} & & & & \\ & & e^{i\frac{2\pi(2)}{N}j} & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & & e^{i\frac{2\pi(N-1)}{N}j} \end{bmatrix} Wj \tag{8}$$

$$\therefore DFT[\,d_{j-1}\,] = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & & & & & \\ & e^{i\frac{2\pi}{N}j} & & & & \\ & & e^{i\frac{2\pi(2)}{N}j} & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & & e^{i\frac{2\pi(N-1)}{N}j} \end{bmatrix} Wj \tag{9}$$

This conclusion is demonstrated through the proposed SIMULINK model of this chapter through considering real time signals for the input d_j, and then it will be compared with that of the standard sliding DFT system in the simulation result section.

3. Guide for using SIMULINK

This is a brief guide to using SIMULINK regarding to the modelling and simulation of dynamic systems. It gives a general introduction to SIMULINK, its importance and some useful tips of modelling with SIMULINK. The guide provides sufficient knowledge to understand the proposed model of this chapter, especially for those who have a limited knowledge of SIMULIMK. A simple DSP example, namely a 'tapped delay line' of an arbitrary signal, is adopted in the end of the guide as a simple but a good practice of modelling with SIMULINK.

Finally, it is important here to indicate that the entire guide, and even the design of the proposed system of this chapter, assumes that the MATLAB is installed on a platform with a

Microsoft Windows operating system and not a Macintosh environment. Consequently, this may lead to some differences to those who are using a Macintosh operating system when dealing with the graphical user interfaces.

3.1 SIMULINK at a glance

SIMULINK which stands for SIMUlation and LINK, is a simulation tool that runs under the MATLAB package which stands for MATrices LABoratory, the main software by Mathworks. SIMULINK combines both the simplicity and the high capability of simulation and modelling of dynamic and embedded systems. SIMULINK can be used efficiently in the design, implementation, simulation and testing of a wide range of applications like communications, controls, signal processing, video processing and image processing. Although SIMULINK requires the installation of MATLAB on the platform of simulation, in general it doesn't require prior knowledge of programming using MATLAB, which is the case here in our proposed model-based system design; however, in other cases it does require such knowledge, but it is out of the scope of this chapter to go further in explaining such cases.

SIMULINK, as stated earlier, works under the MATLAB package, so it requires that MATLAB is installed on the simulation platform, normally a personal computer (pc), and the installation of SIMULINK choice should be selected during the installation process. Thus, the first step is running MATLAB. MATLAB can be run by double-clicking on the MATLAB icon on the desktop, and then SIMULINK is easily opened by either clicking on the SIMULINK icon in the MATLAB toolbar or by writing SIMULINK in what is called the *command window*.

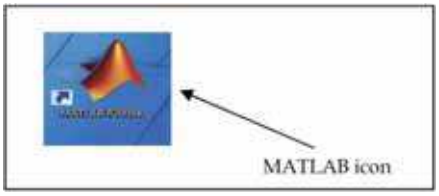


Fig. 2. Opening a MATLAB session

SIMULINK can be opened in different ways. The simplest way is by clicking on the SIMULINK icon in the MATLAB toolbar as illustrated in Fig. 3.

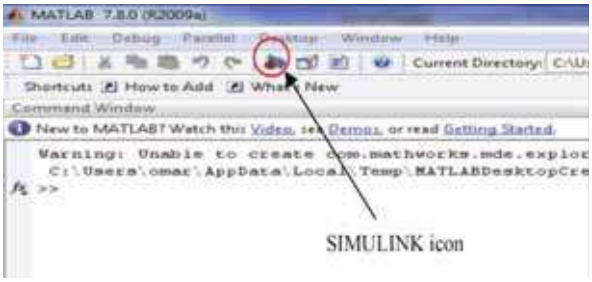


Fig. 3. Opening SIMULINK

Fig. 4 shows the SIMULINK Library Browser window that appears after clicking on the SIMULINK icon. This is where you can find all the components that you may add to your model. Components are known as blocks in the SIMULINK context. Once the SIMULINK library browser is opened, a modeling environment has to be opened. This is called a model file, which is opened from the SIMULINK library browser toolbar by clicking on the new model icon as illustrated in Fig. 4. Fig. 5 shows a blank new model file.

Fig. 4. The SIMULINK Library Browser

In this model file you can create, connect all of your system components, simulate, change simulation parameters and finally view and/or print the results. The model file has to be saved with a suitable name and it will be saved in a default folder, namely MATLAB, which is created automatically during the installation period. Fig. 6 clarifies how to save a SIMULINK model.

Fig. 5. New SIMULINK model file

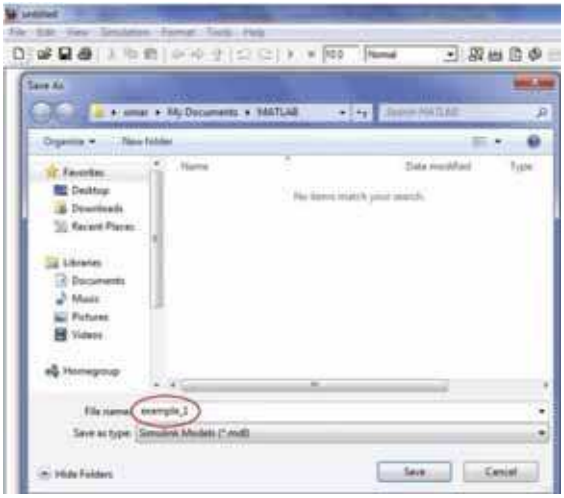


Fig. 6. A standard Windows operation to save SIMULINK model files

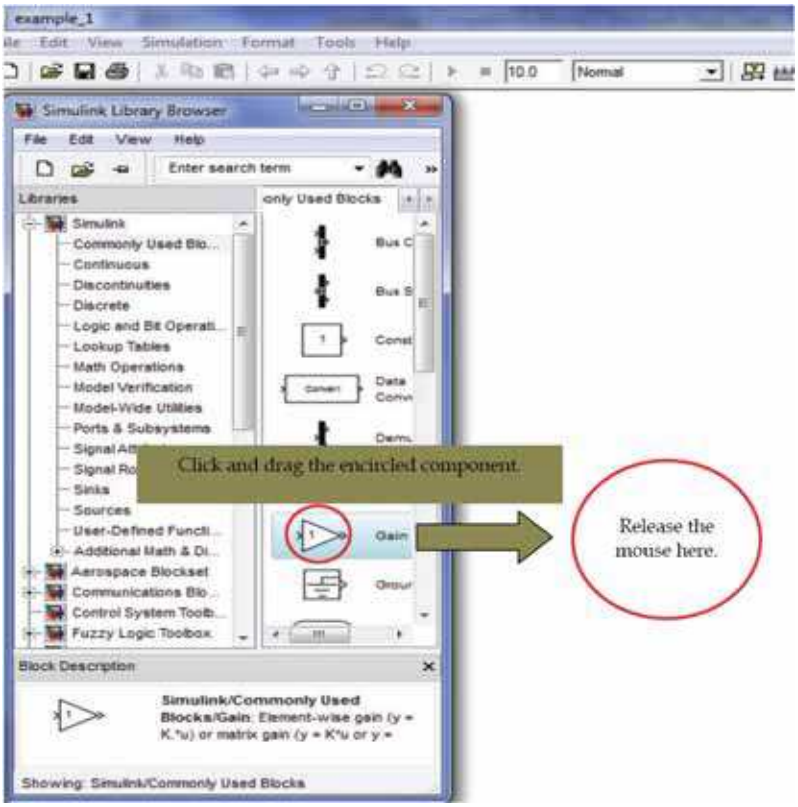


Fig. 7. Inserting blocks in a SIMULINK model

Now, the entire components/blocks that you may wish to add to your model are available at the SIMULINK LIBRARY BROWSER. The next step is inserting and connecting the blocks of the model. A block can be inserted from the library browser very easily by left-clicking on the block, holding down the button, dragging it and then releasing the mouse in the model file. The procedure of adding a block to a model is illustrated in Fig. 7. It is useful here to indicate that a brief explanation about the mathematical operation of the selected block appears at the bottom of the library browser, as is illustrated in Fig. 7.

Similarly, the entire model's blocks might be inserted in the same way. At this point, after inserting all of the required model components, you may rearrange the locations of the components within the same model file in such a way that best matches the schematic diagram of the simulated system. The next important step is linking the blocks of the model. Fig. 8 shows how to link two blocks together automatically, while Fig. 9 shows how to make a branch line.

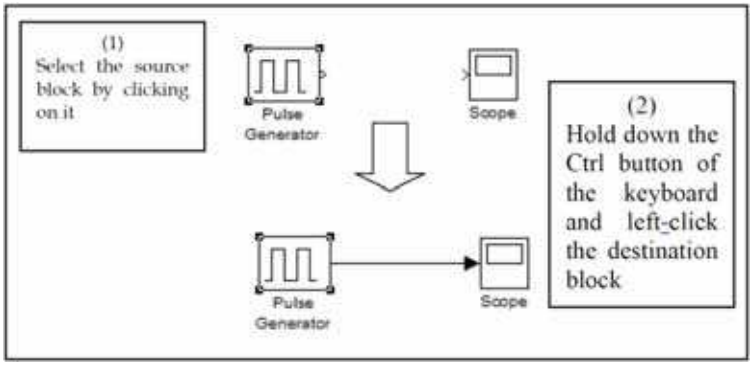


Fig. 8. Linking blocks in the model file

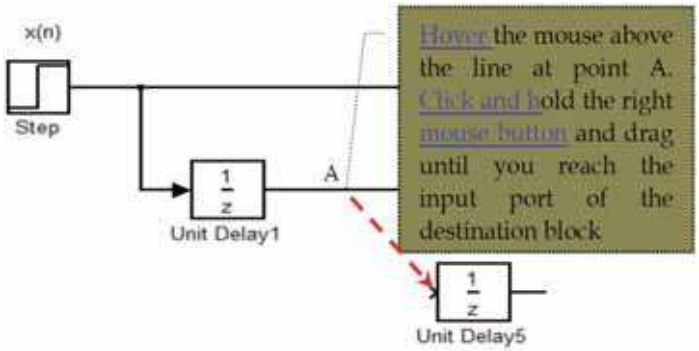


Fig. 9. Making a branch line

Each SIMULINK block has its own simulation parameters, and these parameters can be reached by double-clicking on the block and then it can be changed according to the simulation requirements. As an example, Fig. 10 shows how to change the gain value of a gain block.

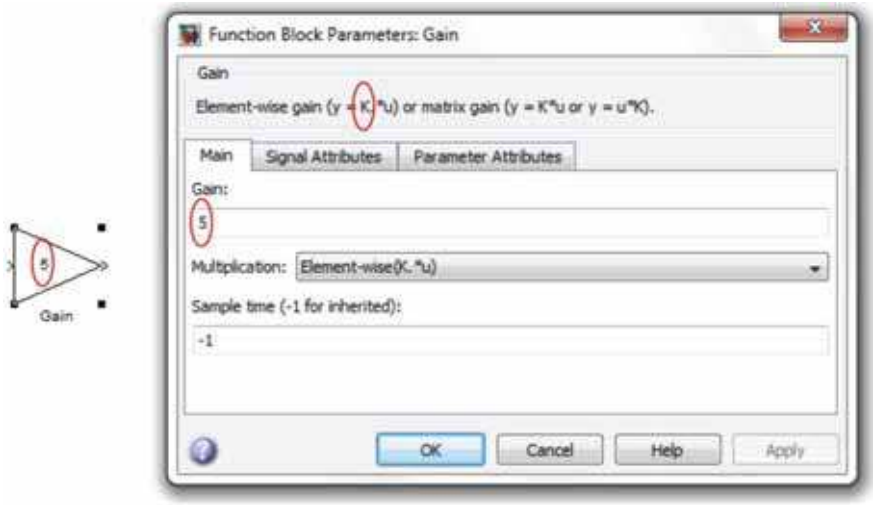


Fig. 10. SIMULINK block parameter viewing/changing

Finally, the facility of viewing the simulation results in various ways, adds more flexibility in performing simulations using SIMULINK. Actually, there are several devices that you might use for this purpose, namely *scope*, *x-y plotter*, *display*, *spectrum analyzer*, and *to work space*. The illustrative example in the next subsection shows the use of a *scope* device to display the simulation results. The user has the choice of inserting the scope device anywhere in the model to view the desired signal in the time domain.

3.2 An illustrative example of running a SIMULINK model: A tapped delay line model

This example shows a very common process in the DSP applications, which is the creation of a tapped delay line, which is a simple practice to learn how to create/run SIMULINK models. So, the idea is to create a lines that carry delayed versions of an arbitrary signal, say x(n). Suppose that there is a need to create delayed versions of x(n) of up to x(n-4). The model will be created on the last saved model, i.e. the one shown in Fig. 6 which holds the model name 'example_1'.

The model creation and simulation may be summarized in the following steps.

Step 1. Insert four unit delays from the discrete blocks category.
Step 2. Insert an arbitrary input signal from the source blocks category, and let it be a step block.
Step 3. Insert a scope block from the sinks category of blocks. Change the number of scope inputs to five, by double-clicking on the scope block and then changing the number of the axes parameter as shown in Fig. 11.
Step 4. Re arrange all of the blocks inside the model and connect them. Follow the procedure given in Fig. 9 to connect a line with a block.
Step 5. Click on the run button, as indicated in Fig. 12, to run the model for a default simulation time of 10 sec.

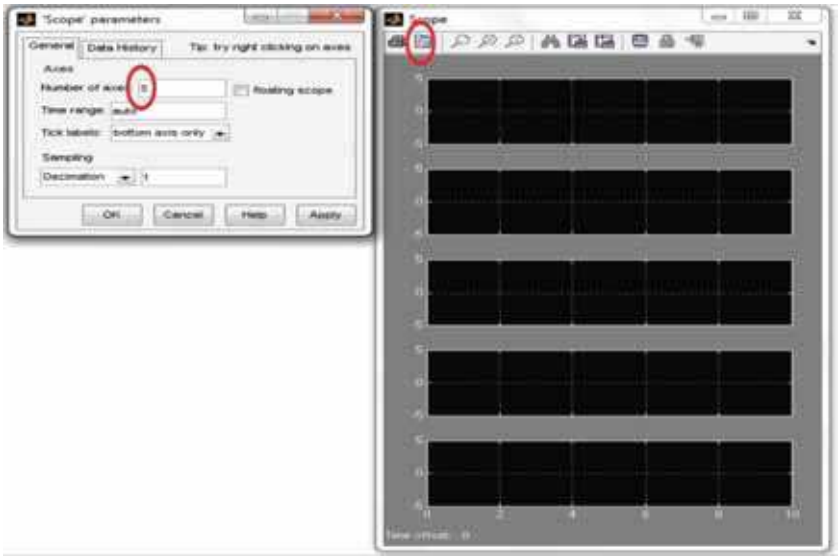


Fig. 11. Changing the number of inputs of the scope block

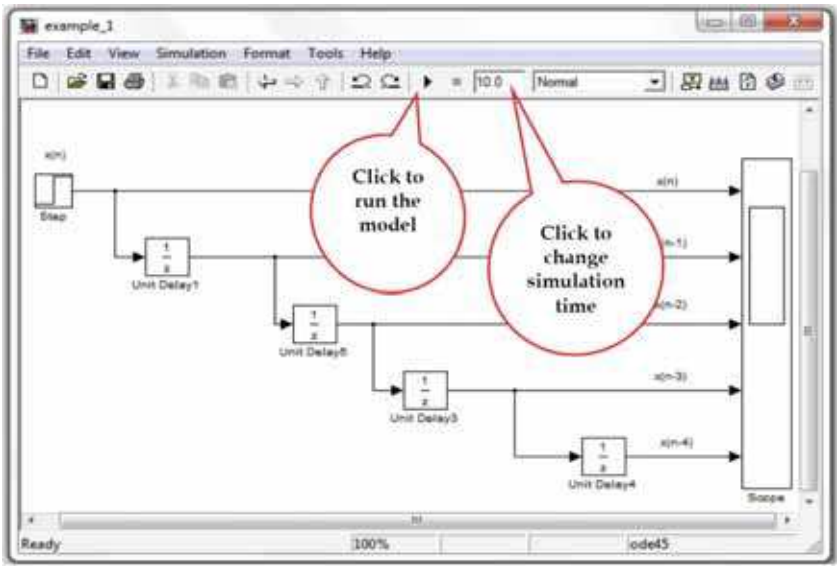


Fig. 12. Tapped delay line model

SIMULINK provides the facility of creating subsystems. This means that you have the ability to create your own blocks by combining a number of blocks together to perform a certain function. It is useful here to explain how to create a subsystem using the following example, as this facility will be adopted in this chapter in the creation of the LMS spectrum analyzer model.

Suppose you need to use a tapped delay line of certain delays many times in your model or perhaps you want to create your own library of blocks within your SIMULINK package. SIMULINK provides the ability to do that by grouping them within one block with a default name of *subsystem 1*. This can be realized by selecting all the corresponding components and then right-clicking on one of them and choosing the create subsystem option form the menu as illustrated in Fig. 13.

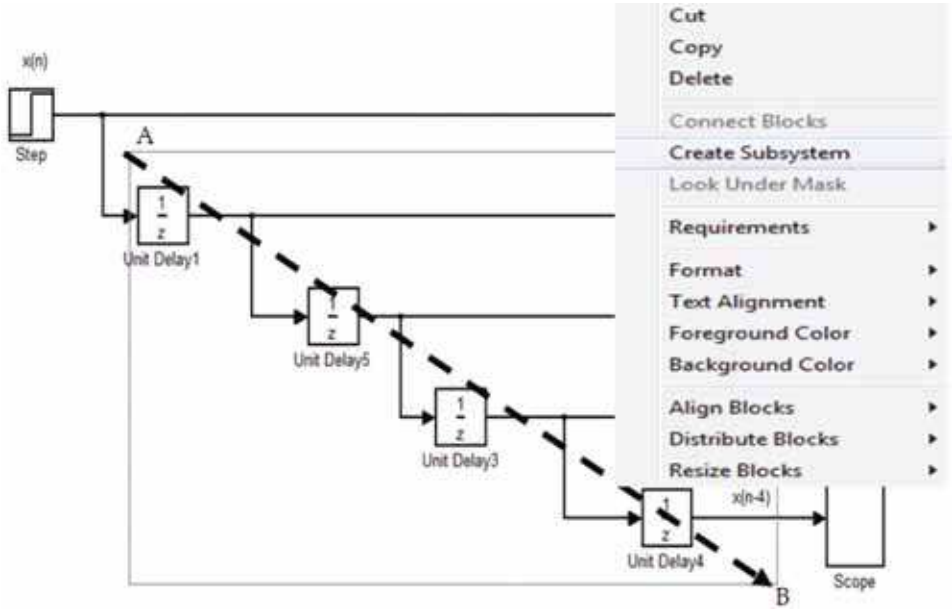


Fig. 13. Subsystem creation; one approach

Now your subsystem has been created, as shown in Fig. 13, and it is ready to use. You can resize your subsystem in order to make it fits its location in the model and also rename the input and/or the output ports in order to make your subsystem more organized and more expressive. Some of the useful operations on a subsystem are shown in Fig. 14.

4. The proposed WASA SIMULINK based design

In this section, a SIMULINK model which corresponds to the theoretical model of WASA is created and later it will be tested for a 4-points DFT case. The complete model is shown in Fig. 15. This model consists of two main subsystems, namely the phasor creation subsystem and the LMS adaptation subsystem. Other blocks are the summer, subtractor, and the mixers bank. The phasor creation subsystem is responsible for the generation of the frequency vector X_j which is defined by equation 3. The procedure that is followed here to realize each phasor is to combine two sine and cosine source signals to form the required phasor according to Euler's formula;

$$e^{\pm j\theta} = cos(\theta) \pm j\, sin(\theta) \tag{10}$$

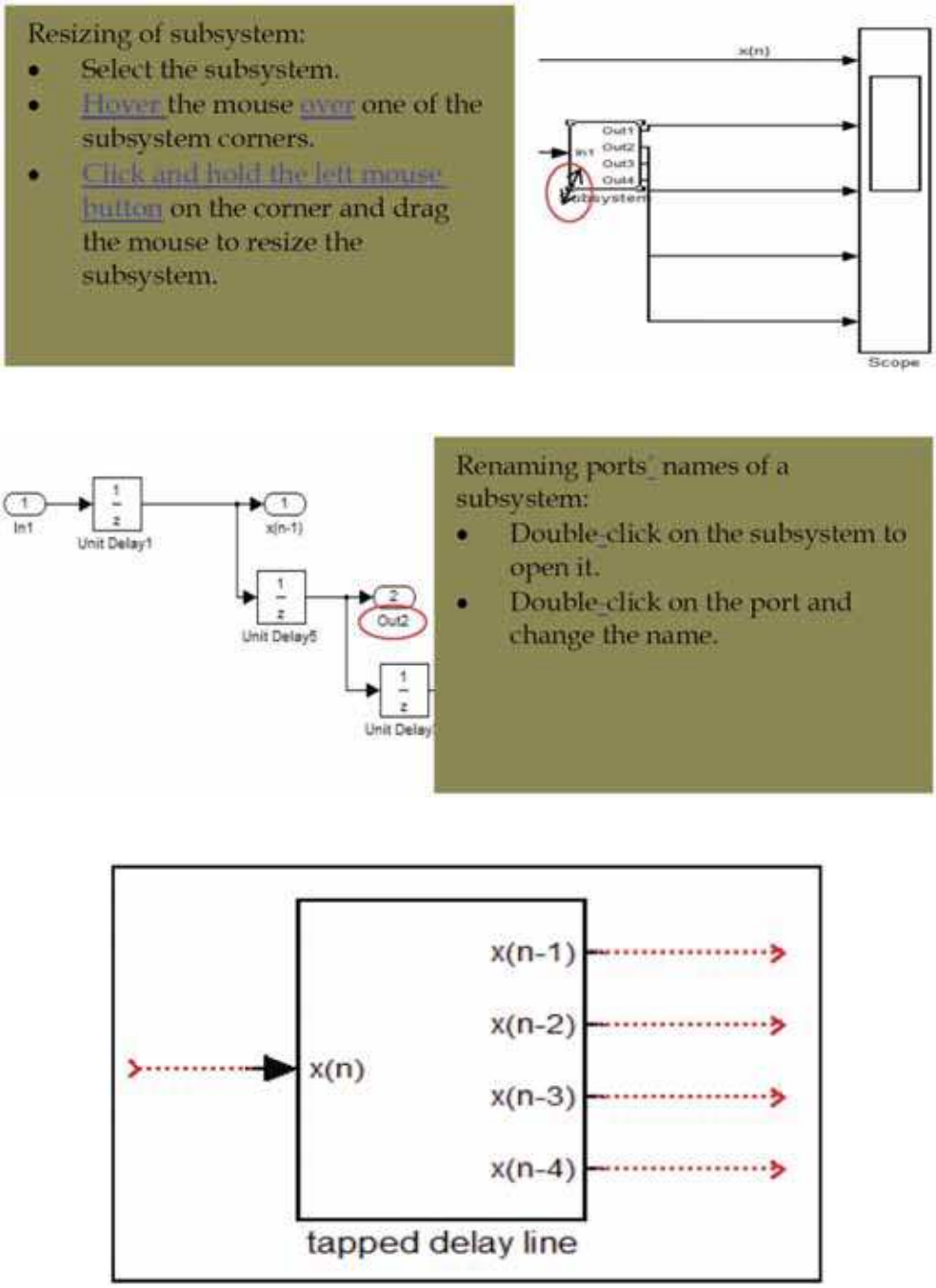


Fig. 14. Useful operations on a subsystem

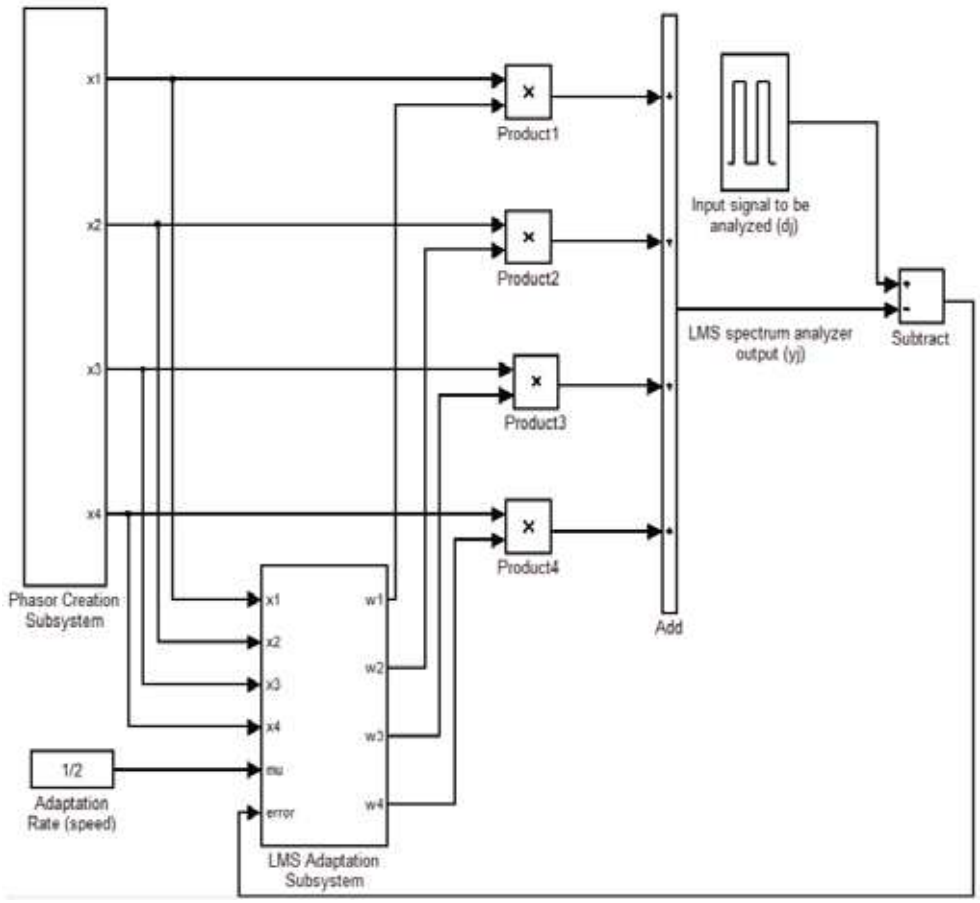


Fig. 15. The proposed WASA SIMULINK model

Fig. 16 shows a proposed 4points DFT phasor creation subsystem. Coming down to the block level, each phasor is composed of sine and cosine signals with the corresponding frequency band that starts from the DC component and ends with the $(\frac{2\pi(N-1)}{N})$ component.

The outputs from this subsystem are then fed to two places: firstly, to the mixer bank for the sake of calculating the spectrum analyzer output, y_j; secondly, to the second main subsystem. This is the LMS adaptation process because these phasors are included in the adaptation process of the weight vector W_j. Fig. 17 shows the contents of the LMS adaptation subsystem. The blocks inside the red rectangle, which are the simulation of equation 5, perform the adaptation role for each weight of the weight vector. Also, this figure shows how the instantaneous error and the adaptation rate constant are fed to all of the adaptation roles of the weights, w_j.

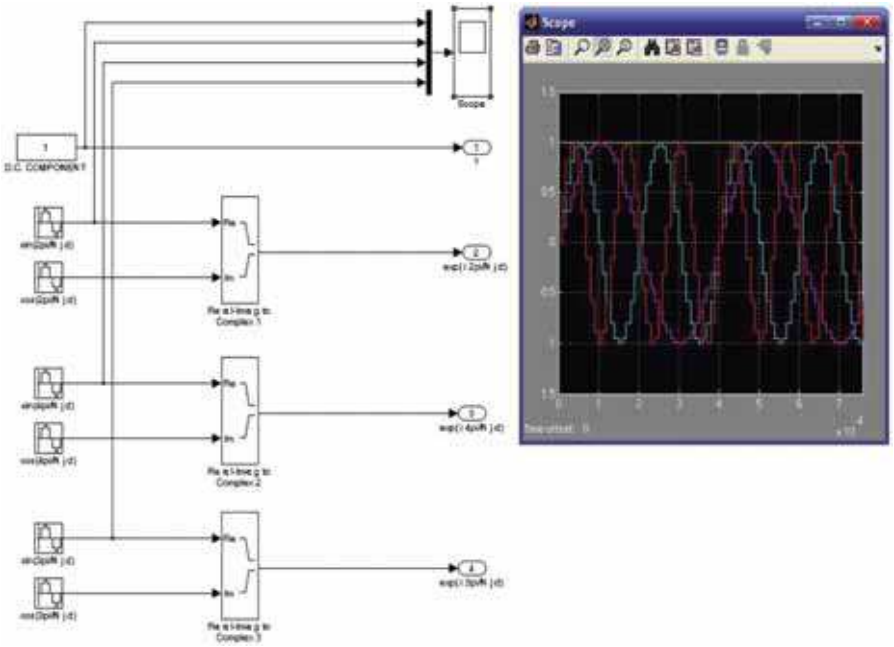


Fig. 16. The block connection details inside the phasor creation subsystem

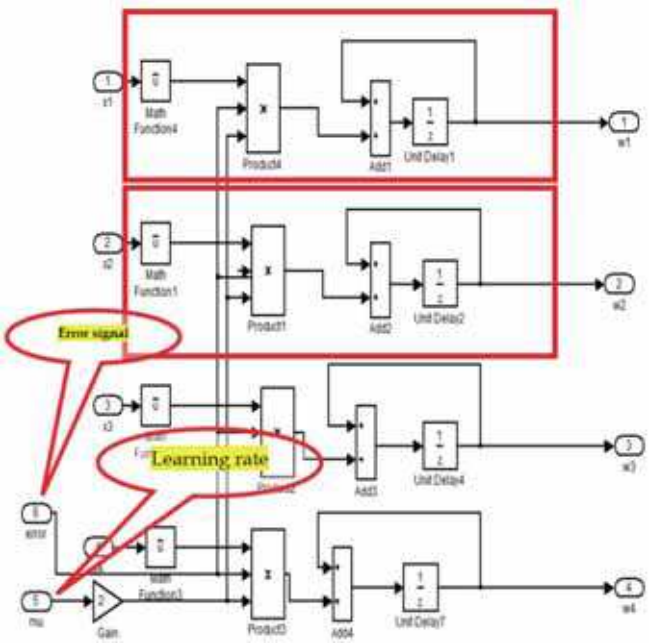


Fig. 17. The detailed blocks inside the LMS adaptation subsystem

The mechanism of operation of this model might be summarized as follows. The phasors are created in the phasor creation subsystem at each time sample index j. These vectors are weighted by the pre-loaded values of the weight vector through the mixer bank, then the weighted phasors are summed together to generate a sample of the LMS spectrum analyzer output y_j. The latter is subtracted from the input signal sample d_j to generate the instantaneous error signal ε_j. This error is fed to the LMS adaptation subsystem to update each weight gain of the weight vector instantaneously with the adaptation speed μ and the scaled phasor vector X_j. Now this model is finalized, and it is ready for setting the simulation parameters and exploring the results.

5. Simulation and results discussions

In this section, the performance of the proposed WASA SIMULINK model is tested using the following two scenarios.

1. Imposing a test signal and the results, i.e. the estimated DFT coefficients of the input test signal (d_j), are compared with those which are obtained from the steady flow DFT operation.
2. Imposing a test input signal and while running the model. New frequency contents are introduced in this test signal to test the adaptability feature of the proposed model and snapshots are taken to show how the power spectrum is updated to include the newly-added frequency contents.

The simulation parameters, results and results discussions for the two scenarios are as demonstrated below:

Scenario (1)

Simulation parameters

Simulation time (run time): 10 sec.

WASA output size: 4-DFT coefficients.

Input signal, d_j: pulse signal with a period time of 4μsec, 50% duty cycle, amplitude of 1volts, and sampling rate of 1 MSps.

Simulation results and discussion

The simulation results for this scenario are compared with those obtained from the conventional steady flow DFT. The steady flow DFT operation is designed from the tapped delay line example which was realized previously in Fig. 12 and a DFT SIMULINK block. The results for the proposed WASA model and the steady flow DFT are shown in Fig. 18.

The *display* block is adopted to display the calculated and the estimated DFT coefficients from the sliding DFT and the proposed WASA models respectively. After performing the running of the model, it is clearly seen that the results are absolutely coincident as they appear in their respective display blocks.

Another interesting result to be shown here is the instantaneous error signal, ε_j. Fig. 19 shows various cases for the sampled version of this signal after running the proposed model three times for three values of the learning speed μ. The results justify the mathematical proof in (Widrow, et al, 1987) which is given by equation (9) and states that if μ is chosen to be 0.5, the output of the adaptive LMS spectrum analyzer will be equal to the DFT of the input sampled signal at the previous time instance. Otherwise, ε_j will never reach zero as the case shows in Fig. 19 (a). Fig. 19 (c) shows that when μ is chosen to be more than 1, ε_j is increasing up to infinity as the LMS algorithm becomes unstable.

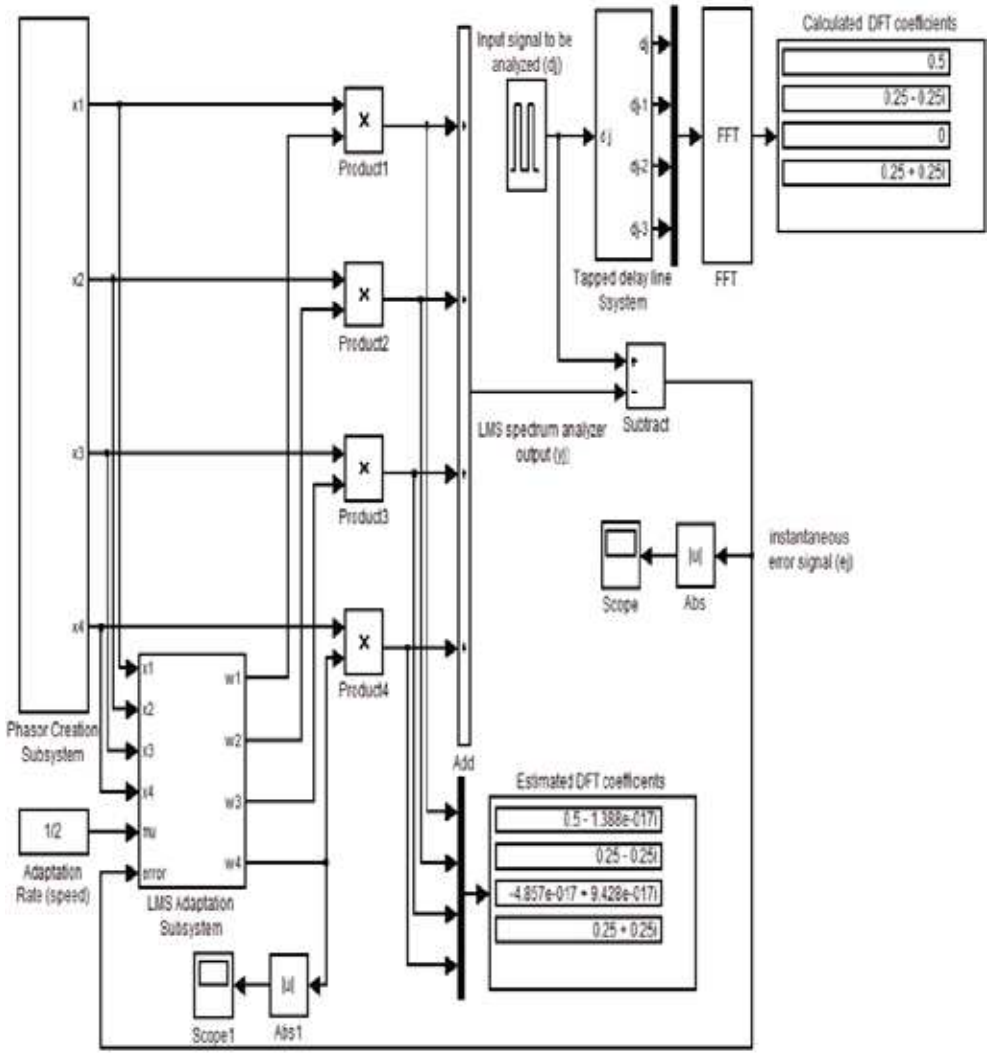
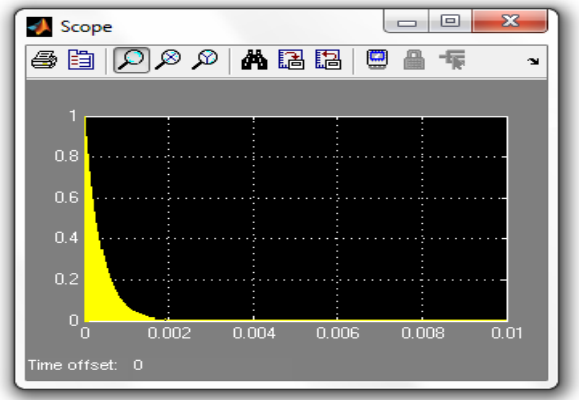
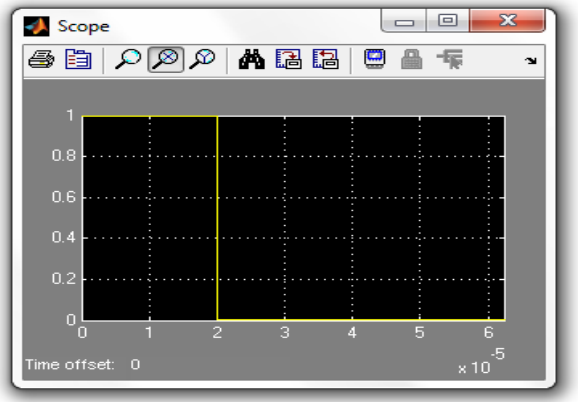


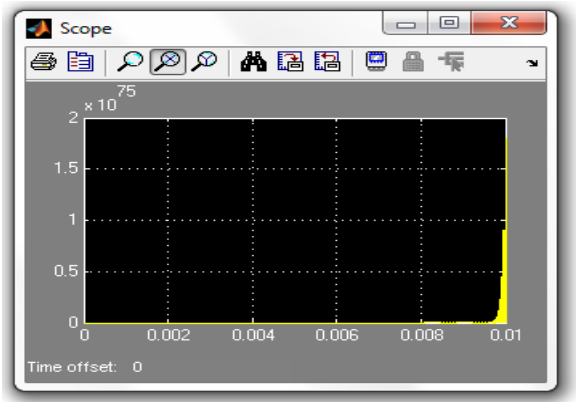
Fig. 18. 4 points DFT WASA model results as compared with sliding DFT



(a) μ = 0.05



(b) μ = 0.5



(c) μ = 1.5

Fig. 19. The effect of the learning speed, μ, on the instantaneous error signal, ε_j

Scenario (2)

Simulation parameters

Simulation time: 0.1 sec.

WASA size: 4-DFT points.

Input test signal: $d(t) = sin(2\pi f_1 t) + sin(2\pi f_2 t).u(t - t_o)$

f_1= 100 KHz.
f_2= 200 KHz.
t_o= 0.01 sec.

Amplitude of 1volts, and sampling rate of 1MHz.

Simulation results

The second test is the adaptability test. This is carried out by imposing a composite signal of two sinusoidal signals to the model to serve as d_j.

Simulation results and discussion

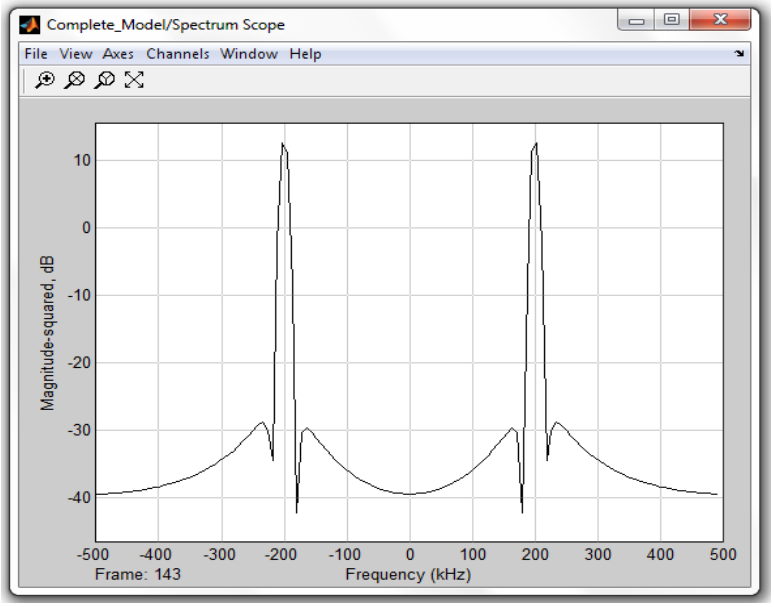
It is clear that this signal contains two fundamental frequencies, f_1 and f_2, but the f_2 component does not appear in the spectrum of the signal until the time instance t_o as the second term of *d(t)* is multiplied by the shifted unit step function. Therefore, the WASA proposed SIMULINK model operates on the term $sin(2\pi f_1 t)$ to estimate the f_1 component until the time instance t_o, then the weight vector will be adapted by the LMS role again to estimate the frequency content of the second term, which is f_2. The results are viewed using the Spectrum scope block which is connected at the output node of the WASA model.

Two snap shots are taken from the spectrum scope block: one before t_o and the other after t_o, as shown in Fig. 20.

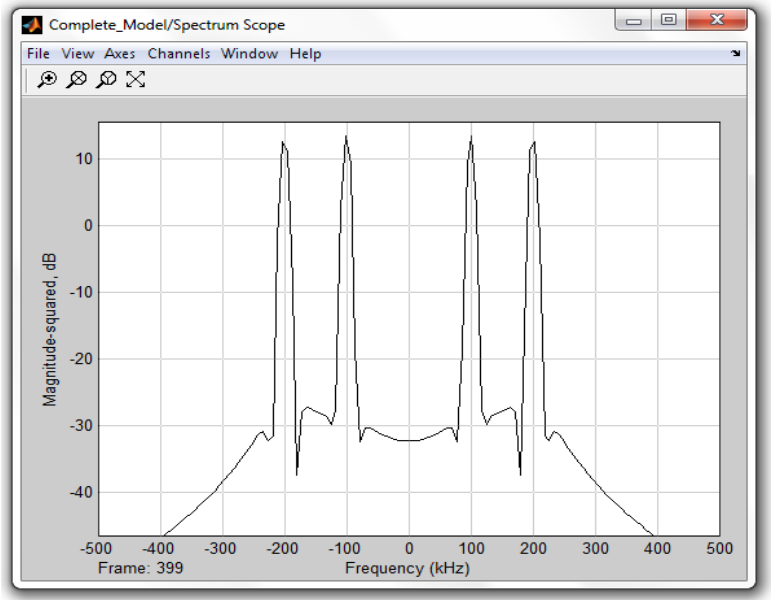
Finally, it is clearly seen that the proposed model simulated the WASA system perfectly; however, there is an issue that seems to be a drawback for the procedure of the proposed WASA model creation. The point is that when the size of the system is small or moderate, i. e. the size of the DFT operation, the system creation is quite simple and doesn't require a lot of effort to finalize it, which is the case for points 2 to 16. But what about the points 32 and up; 128, for example? The issue is that the system will require a bulk of blocks to be embedded in, and the branching of lines across the model will be tedious work. So, there must be another strategy to be adopted in the model creation phase. This will be clarified in the suggestions of future work section.

6. Future work

Actually, there are two tracks to be followed for the use of the proposed WASA SIMULINK-based design and/or developing the proposed adaptive spectrum analyzer. First is to generalize the model by generalizing the two main subsystems of the model, the phasor creation block and the LMS adaptation, to work with any specified number of DFT points. This would be realized by using MATLAB programming to create two script files that represent the mathematical relations of these two subsystems, then turning these scripts into



(a)



(b)

Fig. 20. Results for the second test/adaptability test. (a) Frequency spectrum before adding the 10KHz component (b) Spectrum after adding the 10KHz component

SIMULINK blocks and embedding them later with the main model. This is a facility that SIMULINK provides, which makes the work look more professional and does not require huge and tedious wiring work during the model creation. The input parameter of the phasor creation block, for example, would be the number of the preferred DFT coefficient and the sampling rate.

Secondly is to extend the work of this chapter in the design of a class of digital modulation receivers. This could be the design of frequency domain adaptive equalizers, which adopt the frequency domain of the signals to remove the transmission channel noise from the received contaminated signals. Therefore, one can adopt the proposed WASA model in the design of the two layer linear structure for fast adaptive filtering, which was presented by (Beaufays, Widrow, 1995). Fig. 21 shows WASA location in the system inside the thick box.

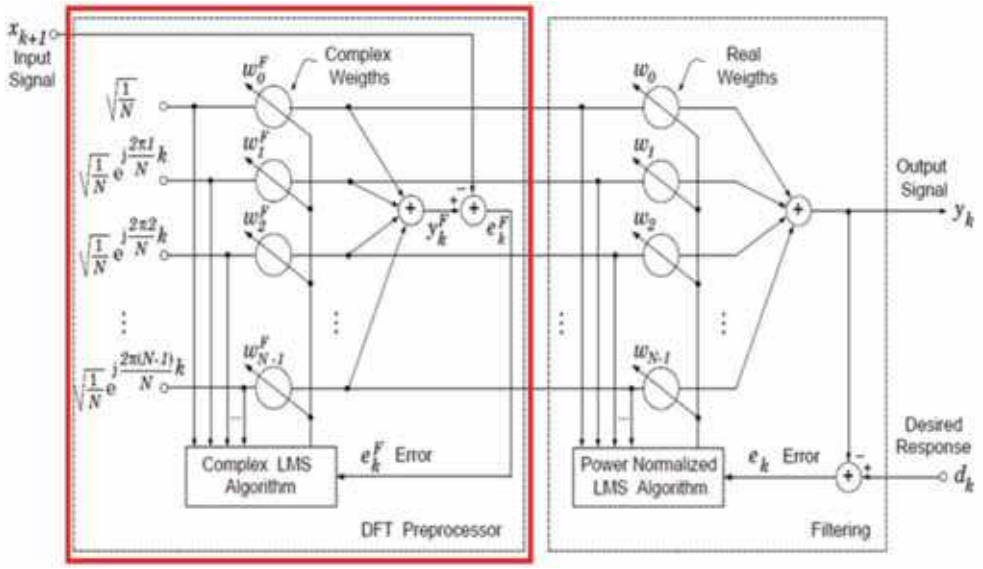


Fig. 21. WASA suggestion for adaptive frequency equalization (Beaufays, Widrow, 1995)

7. Conclusion

A model-based design is proposed to simulate the LMS adaptive spectrum analyzer. A SIMULINK simulation environment is used for its simplicity and high capabilities. The proposed model successfully simulated Widrow's model for 4-DFT points. The results shows a coincidence between the estimated DFT coefficients from the proposed model and the results calculated from a standard steady flow DFT model. Also, it justifies that when the adaptation rate, μ of the model is set to 0.5, then a set of weighted phasors exactly corresponds to the DFT coefficients of the applied input to the model. In addition, new frequency contents are added to the test input signal during the run (online) as a separate test to assess the adaptability property of the spectrum analyser model for the sudden changes in frequency content of the input signal. As a future work, a generalized SIMULINK model is suggested for any specified n-point DFT.

8. References

Alvarez, A. O., and Rivera, L. N., Meana, H. P. (1999). Real time high frequency spectrum analyzer. *Proceedings of 42nd symposium on circuits and systems*, Vol. 2, (August 1999), pp. (753-756)

Beaufays F., and Widrow, B. (1994). Two-Layer Linear Structure for Fast Adaptive Filtering. Proceedings of WCNN, San Diego, USA, Vol. 3, June 1994

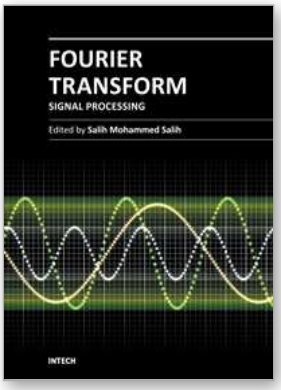
Beaufays, F., Widrow, B. On the advantages of the LMS spectrum analyzer over nonadaptive implementations of the sliding –DFT. (1995). *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, Vol. 42, No. 4, (April 1995), pp. (218-210)

Liu, B., Bruton, L. T. (1993). The two-dimentional complex LMS algorithm applied to the 2-dD DFT. *IEEE Transactions on Circuits and Systems-II: Analog and digital signal processing*, Vol. 40, No. 5, (May 1993), pp. (337-341)

Macgge, W. F. Fundamental relationship between LMS spectrum analyzer and recursive least squares estimation. (1989). IEEE Transactions on Circuits and Systems. Vol. 36, No. 1, (January 1989), pp. (151-153)

Ogunfunmi, T., Au, M. (1994). 2-D discrete orthogonal transforms by means of 2-D LMS adaptive algorithms, Poceedings of signals, systems and computers, 0-8186-6405-3, Pacific Grove , CA, USA, November 1994

Widrow, B., Baudrenghien, P., Vetterli, M., and Titchener, P. (1987). Fundamental Relations Between the LMS Algorithm and the DFT. IEEE Transactions on Circuits and Systems, Vol. 34, No. (7), (July 1987), pp. (814-820)



Fourier Transform - Signal Processing
Edited by Dr Salih Salih

ISBN 978-953-51-0453-7
Hard cover, 354 pages
Publisher InTech
Published online 11, April, 2012
Published in print edition April, 2012

The field of signal processing has seen explosive growth during the past decades; almost all textbooks on signal processing have a section devoted to the Fourier transform theory. For this reason, this book focuses on the Fourier transform applications in signal processing techniques. The book chapters are related to DFT, FFT, OFDM, estimation techniques and the image processing techqniques. It is hoped that this book will provide the background, references and the incentive to encourage further research and results in this area as well as provide tools for practical applications. It provides an applications-oriented to signal processing written primarily for electrical engineers, communication engineers, signal processing engineers, mathematicians and graduate students will also find it useful as a reference for their research activities.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821