Journal of Marine Science and Technology

Volume 4
Issue 1 *6/15/1996*

Article 10

A Modularized Software Package for Adaptive Beamforming

Shun-Hsyung Chang
Associate Professor, Department of Electrical Engineering. National Taiwan Ocean Universitry, Keelung, Taiwan, R.O.C.

Shao-Wei Leu
Associate Professor, Department of Electrical Engineering. National Taiwan Ocean Universitry, Keelung, Taiwan, R.O.C.

Hsing-Wang Liang
Graduate Student, Department of Electrical Engineering, National Taiwan Ocean Universiry, Keelung, Taiwan, R.O.C.

A Modularized Software Package for Adaptive Beamforming

Acknowledgements

Funds for this research was sponsored by National Science Council, Taiwan, R. O. C. under grant NSC 82-0209-E-019-064.

Journal of Marine Science and Technology. Vol. 4, No. 1, pp. 1-9 (1996)

A MODULARIZED SOFTWARE PACKAGE FOR ADAPTIVE BEAMFORMING

Shun-Hsyung Chang*, Shao-Wei Leu* and Hsing-Wang Liang**

Keywords: Sonar, Signal processing, Adaptive beamforming, Simulation software, User interface.

ABSTRACT

This paper presents a modularized simulation software package for adaptive beamforming. In recent years, the theory and technology of adaptive beamforming have been applied extensively in many different areas such as radar, sonar, communication systems, and geophysical exploration. As many new beamforming architectures and related algorithms are being introduced, researchers find themselves repeating the laborious process of writing similar software routines in order to conduct simulations. This is largely due to the lack of well-defined and modularized supporting software. Therefore, we have implemented a collection of representative algorithms available for adaptive beamforming and constructed a simulator with menu-driven user interface. The algorithms include generalized sidelobe canceller (GSC), Frost, derivative constraints, high order main beam derivative constraints (HOMBDC), least mean square (LMS), leaky LMS, recursive least square (RLS), and variable step size LMS.

Our goal is to provide a powerful and convenient simulation platform for researchers who want to quickly establish and evaluate some beamforming model of their choice. For students learning the theory of adaptive beamforming, this package will not only help them understand the principles, but also encourage them to apply the theory to the solution of practical problems. For the highest portability possible, the software package is written entirely in C language. The user interface is completely independent of the simulation engine. This, of course, will ease the porting of the package to other graphical environment.

INTRODUCTION

In recent years, the theory and technology of adaptive beamforming have been applied extensively in many different areas such as radar, sonar, communication systems, and geophysical exploration [1,2,3,4]. When processing uncertain signal components, adaptive beamforming, which is largely based on recursive algorithms, is much more effective than the traditional beamforming methods. Adaptive beamforming has a long history in its theoretical development. It spans from the early days when Gauss deviced least mean squares method in the nineteenth century to the mid-century advancement of the relevant algorithm by Wiener [5]. The development continued with the least mean square algorithm proposed by Widrow in early 1960s [3,6]. He obtained the optimal weight by keeping the output energy to a minimum and, thus, maximized the SNR for output. In 1972, Frost developed a linearly constrained algorithm which maintained a certain fixed gain for the signals from the target direction, while cancelling jammers from interfering sources [7]. As a result, the target signal goes through array without attenuation.

The next important advancement was realized by Griffiths when he developed the generalized sidelobe canceller (GSC) in 1982 [8]. He improved on Frost's softly constrained algorithm by using hard constraint. This greatly reduced the computational complexity associated with Frost's method. Then, in the following year, Er developed the derivative constraints algorithm [9]. This method was capable of maintaining the output of array in good quality, even in the presence of steering error in the array system.

To explore the capability of the beamforming array, Jablon has given a detailed analysis on the static response and interference cancelling ability of the GSC in his 1986 article [10,11]. Although many different array systems have been developed, most of them belong to two major categories, namely, the array proposed by Widrow [6] and those proposed by Applebaum and Frost [7]. Due to the inherent computational complexity, all the algorithms for these arrays involve a large amount of mathematical operations. Therefore, it normally takes a long period of time to obtain the output. To speed up computation, Huang and Chang have proposed a systolic beamforming array which incorporates parallel pro-

Paper Received April, 1996. Revised June, 1996. Accepted June, 1996.
Author for Correspondence: Shun-Hsyung Chang.
**Associate Professor, Department of Electrical Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C.*
***Graduate Student, Department of Electrical Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C.*

cessing capability into an array system[12].

As adaptive beamforming continues to be applied to many different areas, more research effort is being directed into this field. Unfortunately, researchers often find themselves repeating the drudgery writing similar software routines when conducting simulations. This is largely due to the lack of support from a well-defined and modualized software package. To relieve the researchers of the burden creating simulation software, we have developed a package capable of simulating many different beamforming architectures and algorithms. Users of this package can quickly model a beamformer and execute various algorithms of their choice. Moreover, students learning the theory will be able to try out different architecture/algorithm combinations with ease. This provides them an efficient means to verify the theory after being exposed to this field.

The organization of this article follows. Section II discusses the theoretical background of adaptive beamforming. Section III introduces the organization of this software package. Section IV describes the user interface. Section V presents some experimental results. Finally, section VI is the conclusion and future work.

THEORETICAL FOUNDATION

Consider a liner array as shown in Fig. 1, which consists of M omnidirectional sensors each with an adjustable weight w_i, where i = 1,2,..,M. Suppose that the signal is far-field and can be regarded as a planar wave, then the signal and noise received and the associated weight for the array can be expressed, respectively, as

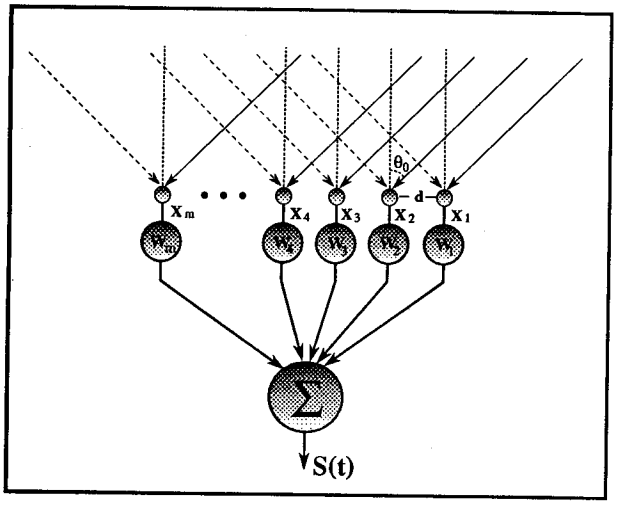

Fig. 1. Simplfied block diagramof a linear array.

$$\mathbf{L}(t)=\begin{bmatrix} l_1(t) \\ l_2(t) \\ l_3(t) \\ \vdots \\ l_M(t) \end{bmatrix},$$

$$\mathbf{N}(t)=\begin{bmatrix} n_1(t) \\ n_2(t) \\ n_3(t) \\ \vdots \\ n_M(t) \end{bmatrix},$$

$$\mathbf{W}(t)=\begin{bmatrix} w_1(t) \\ w_2(t) \\ w_3(t) \\ \vdots \\ w_M(t) \end{bmatrix},$$

where $l_i(t)$ is the signal received by the i^{th} sensor, $n_i(t)$ is the noise received by the i^{th} sensor, and $w_i(t)$ is the weight associated with the i^{th} sensor. Let $\mathbf{X}(t)$ = $\mathbf{L}(t)$ + $\mathbf{N}(t)$, then the output of this array is

$$y(t) = \mathbf{W}^H\mathbf{X}(t), \tag{1}$$

where H denotes Hermitian transpose operation. For the array as configured to be able to suppress jamming and noise, it is necessary to obtain the optimal weight W_{opt}. However, how to obtain the optimal weight is heavily dependent on the underlying algorithm adopted. The computational algorithm also affect the performance of the array system. Using this package a designer can choose among different algorithms according to the system's requirements. We now summarize in the following the theoretical framework and the computational algorithms of some adaptive beamformers adopted in our software package.

Theoretical Framework

Frost Array

The expected output for this type of array is [7]

$$E[y^2(t)] = E[\mathbf{W}^T\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{W}] \tag{2}$$

$$= \mathbf{W}^T\mathbf{R}_{xx}\mathbf{W}. \tag{3}$$

A Frost array produces minimum power output. Also, in the direction of arrival, the signal gain is limited to a fixed level, given by the constraint

$$\mathbf{A}^T(\theta_0)\cdot\mathbf{W} = \mathbf{F}, \tag{4}$$

where $\mathbf{A}^T(\theta_0)$ is the steering vector, T denotes the trans-pose operation, and $\mathbf{F}$ is the vector of summed weight value. If we let $E[\mathbf{X}(n)\mathbf{X}^T(n)] \triangleq \mathbf{R}_{xx}$, then the optimal weight is

$$\mathbf{W}_{opt} = \mathbf{R}_{xx}^{-1}\mathbf{A}(\theta_0)[\mathbf{A}^T(\theta_0)\mathbf{R}_{xx}^{-1}\mathbf{A}(\theta_0)]^{-1}\mathbf{F}. \quad (5)$$

Derivative Constraint Array

The adverse effect of steering errors can be suppressed by the use of derivative constraints. This can be done by assuming a zero value for the derivative at the main beam directions. As a consequence, the acceptance angle is enlarged. Based on high order derivative constraints, the steering vector can be expressed as,

$$\mathbf{A}^T(\theta_0) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & d & \cdots & 0 \\ 1 & 2d & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \vdots & \vdots & (m-p)^p d^p \end{bmatrix}. \quad (6)$$

If we select main beam directions as

$$F = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{(p+1)\times 1}, \quad (7)$$

then the optimal weight $\mathbf{W}_{opt}$ can be found by the method of Lagrange multipliers [7].

Generalized Sidelobe Canceller (GSC)

The GSC method was proposed by Griffiths and im and was an improvement from Frost's algorithm. In Frost's method, computation for optimal weight involves constraining the gain for the signal from the target direction, which is a process demanding a large amount of mathematical operations and, therefore, a long period of time for the output to converge. To reduce the computational burden, Griffiths uses hardware to implement gain constraint for the target direction, which greatly improves the speed of convergence. The key ingredient for the performance improvement over the Frost method is the conversion of a constrained LMS into an unconstrained LMS[8]. Detailed discussion can be found in an article by Buckley [13].

High Order Main Beam Derivative Constraints

In this computational architecture, Griffiths modified the signal blocking matrix, C_n, used in GSC into the main beam derivative constraints. The objective is to prevent the target signal from "leaking into" the sidelobe canceller. Griffiths and Jim have indicated that a first-order signal blocking matrix is capable of correcting the steering error [8].

Algorithms for Adaptive Beamformers

LMS (Least Mean Squares)

The LMS method was proposed by Widrow [6]. It obtains the weight from the following equation:

$$\mathbf{W}(n+1) = \mathbf{W}(n) - 2\mu\epsilon(n)\mathbf{X}(n), \quad (8)$$

where μ is the step size and $\epsilon(n)$ denotes error.

Leaky LMS

This algorithm aims at improving speed of convergence for LMS [2]. Its weight formula is

$$\mathbf{W}(n+1) = \gamma\mathbf{W}(n) + \mu\mathbf{X}(n)\epsilon(n), \quad (9)$$

where γ is the Leaky coefficient.

Variable-Step-Size LMS

This algorithm uses a variable step size. At the beginning stage of computation, large μ is selected in order to speed up convergence. As the computation continues, μ is gradually reduced to lower the possibility of oscillation. The weight formula is modified as [2]

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \frac{1}{K+n}\mathbf{X}(n)\epsilon(n), \quad (10)$$

where K is a positive constant.

RLS (Recursive Least Squares)

This algorithm is developed from the least square method by using recursion to obtain its weight [14]. Detailed discussion can be found in Haykin [1].

A MODULARIZED SOFTWARE SIMULATION PACKAGE

The software package we have developed cov-

ers a collection of representative algorithms and computational architectures, such as GSC, Frost, Derivative Constraints, HOMBDC, LMS, RLS, etc. We started the design process by examining and comparing the various architectures and algorithms. By sorting out as many similar computational tasks as possible from among all the available algorithms, we were able to construct a collection of fundamental software routines. As a result, our simulation package is highly flexible and efficient in simulating various algorithms. Due to a modularized design concept, this package will extend easily as the need to incorporate new architectures and algorithms arises.

As shown in Fig. 2, the core of this package consists of two major parts. One part contains the software routines implementing the available algorithms and architectures; the other is a random number generator. The algorithm/architecture part is the simulator core, which is further divided into four units, as shown in Fig. 3. A brief description of these four units follows.

Signal Generation Unit

This unit provides both the target signals and the jamming signals for the simulation routines. Besides these two types of signal sources, the signal generation unit also includes an array processing unit. As illustrated in Fig. 4, the target signal unit and the jamming signal unit produce the target signal, $exp(jw_s t)$, and the jamming signal, $exp(jw_j t)$, respectively, for the simulator. The array processing unit is responsible for processing the latency which occurs as the signals travel from their sources to the array. This unit also accounts for the steering delay and white noise that are incurred when the signals pass through the sensors.

Algorithms Unit

As the core of this simulation package, the algorithm unit comprises a large amount of software routines implementing the various architectures and algorithms. Since the design is fully modularized, it is fairly easy to add new algorithms or delete any obsolete ones. As mentioned earlier, the modularization is based on careful comparisons between different architectures/algorithms. Based on the characteristic of the computational tasks, our design takes two separate paths. The first path includes software routines for GSC and HOMBDC. The second path handles the Frost method and derivative constraints. The outputs from both paths are then fed into the RLLL subunit, which serves to obtain the optimal weight for a beamforming array. It includes four major algorithms, namely, recursive least squares, leaky least mean squares, least mean squares, and variable step size LMS, as illustrated in Fig. 5.

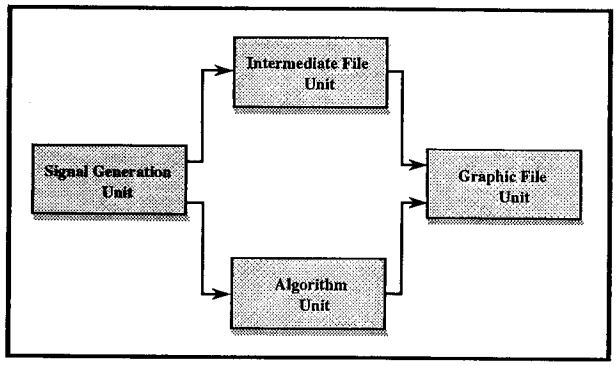

Fig. 3. Structure of the simulator core.

Intermediate File Unit

This unit provides file access for data needed and data generated during simulation.

Graphic File Unit

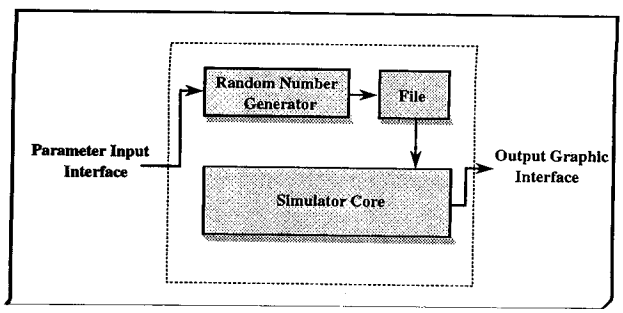

Fig. 2. System organization.

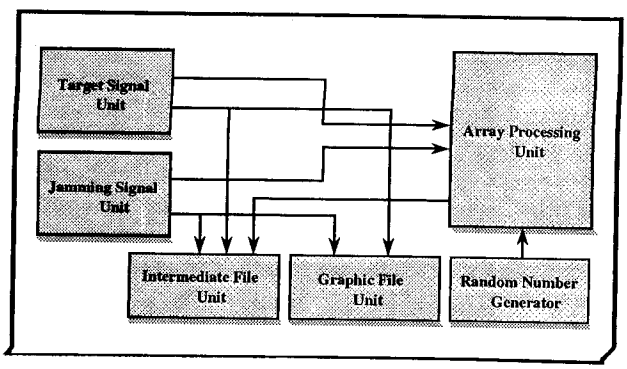

Fig. 4. Signal generationunit.

This unit provides file access for the graphic interface. If ported to a new graphical environment, the data format must change accordingly.

DESIGN OF THE USER INTERFACE

Input Interface

The input interface is needed for setting parameters when running a simulation. Fig. 6 illustrates the functional structure of the interface. The menu-driven input interface includes the following items:

ALGORITHM menu

This menu selects a desired algorithm from among HOMBDC, DERIVATIVE, GSC, Frost, and DELAY SUM, as shown in Fig. 7.

GSC and HOMBDC menu

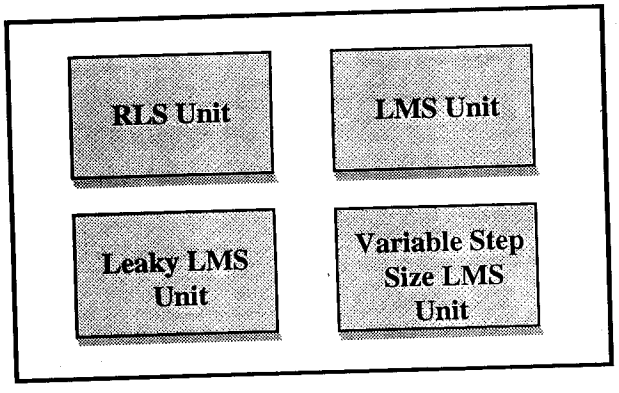

Fig. 5. RLLL operation unit.

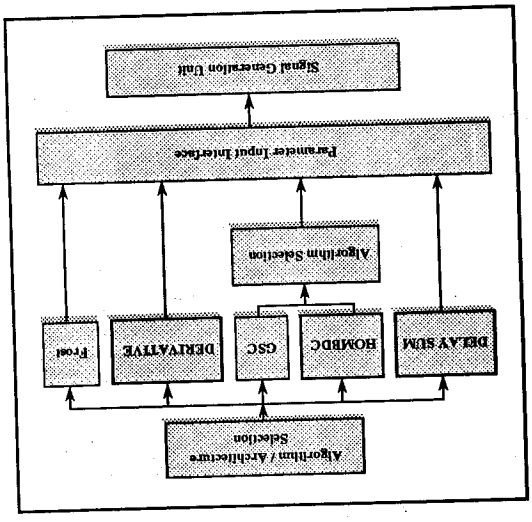

Fig. 6. Structure of the input interface.

This menu selects an algorithm for computing the optimal weight from among LMS, Leaky LMS, VAR LMS, and RLS, as shown in Fig. 8.

Array parameter input screen

This screen accepts parameters including number of sensors, azimuth angle, steering angle, sample number, wave speed, average, array distance, step size, phase, and leaky factor. Fig. 9 illustrates the input screen.

Output Interface

The output interface includes menu items such as Time, Frequency, Learn, Pattern, Clear, etc. the graphic data needed by the interface is all from the graphic file unit, as shown in Fig. 10. To clearly illustrate the function of each menu item, Fig. 11 shows the functional flowchart of the simulator core. A complete description of the output interface follows.

COLOR

This item selects the color and line type for each curve displayed, as shown in Fig. 12. (The built-in line types include dotted line, dashed line, and dot-and-dash line. There are 15 available colors to be selected including red, blue, green, brown, etc.)

TIME

This item specifies the time-domain signal and its pull-down menu is shown in Fig. 13. There are four items to be selected and each displays the time-domain plot of the respective signal. In this menu, SIGNAL specifies the target signal; JAMMER selects the jamming signal; INPUT means the array input signal; finally, OUTPUT displays the array signal after weight adjustment.

FREQUENCY

This item selects the frequency-domain response of either the array input or the weighted output of the array. The pull-down menu is shown in Fig. 14.

LEARN

This displays the mean square error between the desired signal and the weighted array output.

PATTERN

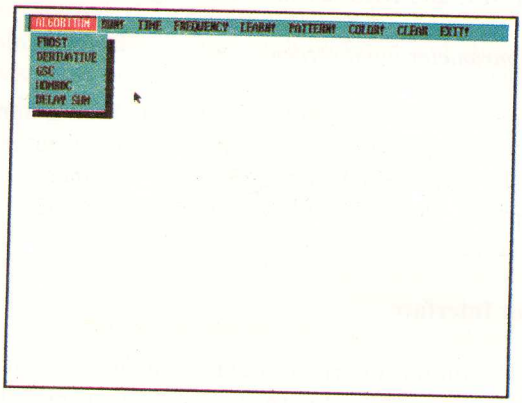

Fig. 7. ALGORITHM sub-menu.

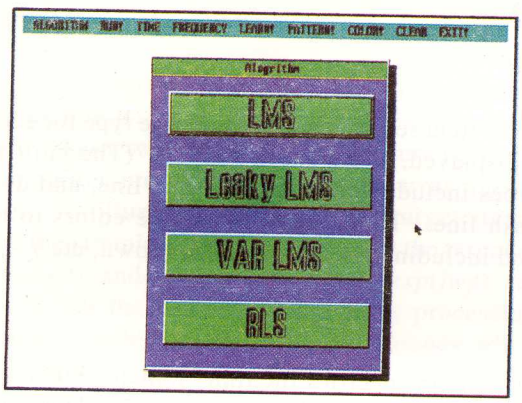

Fig. 8. GSC and HOMBDC sub-menu.

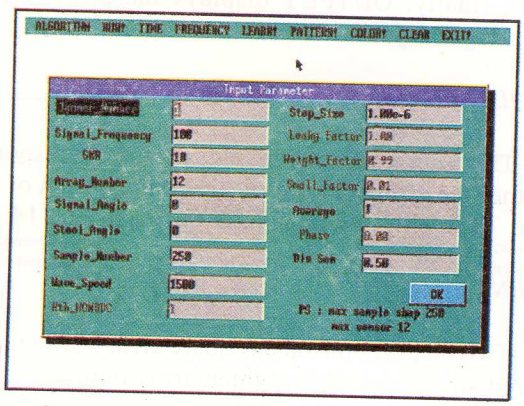

Fig. 9. Array parameter input sub-menu.

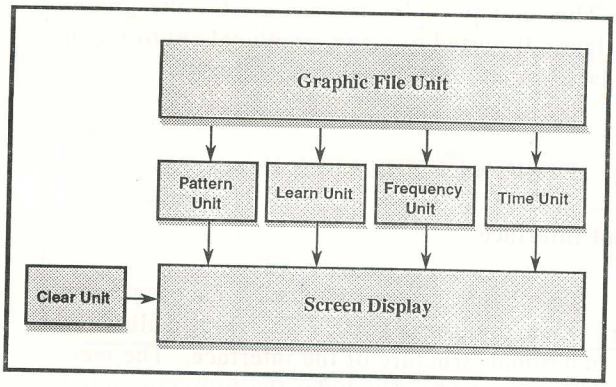

Fig. 10. Graphic file unit.

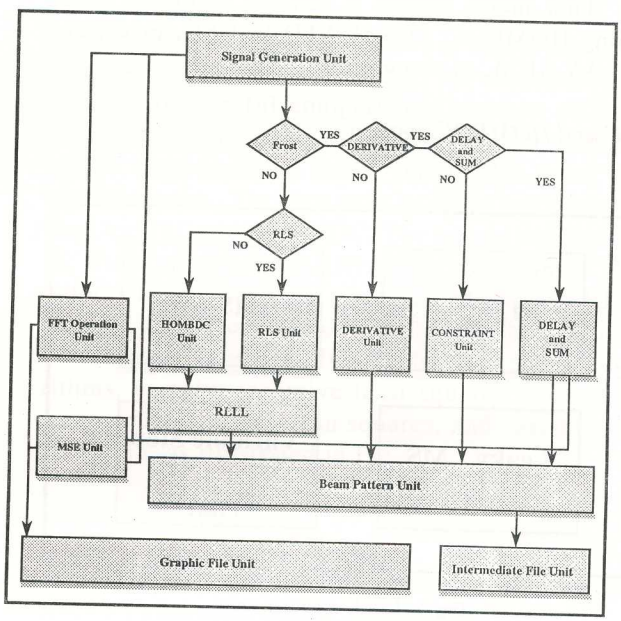

Fig. 11. Functional flowchart of the simulator core.

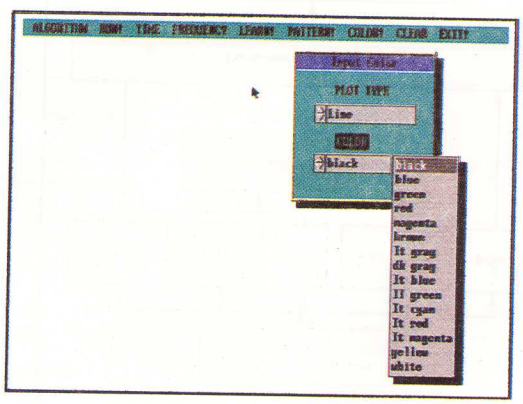

Fig. 12. Color selection for the CLEAR sub-menu.

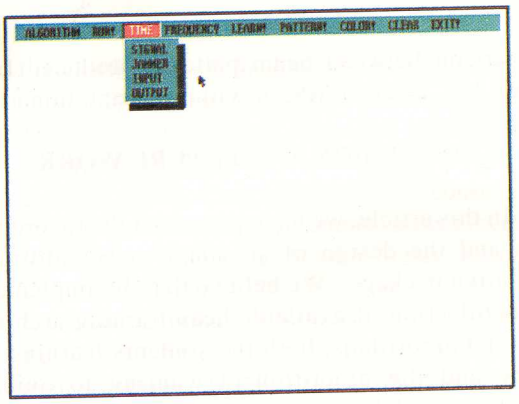

Fig. 13. TIME sub-menu.

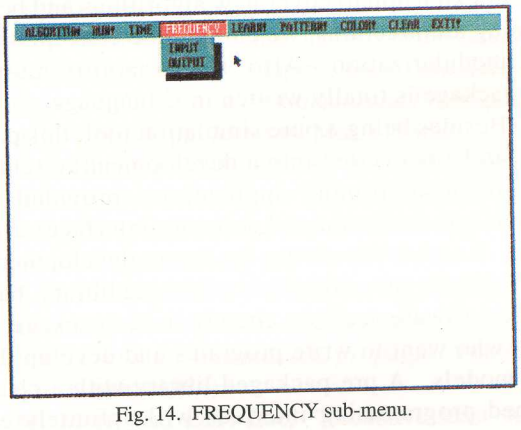

Fig. 14. FREQUENCY sub-menu.

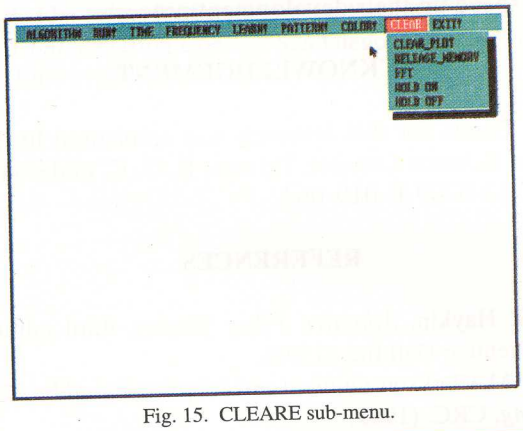

Fig. 15. CLEARE sub-menu.

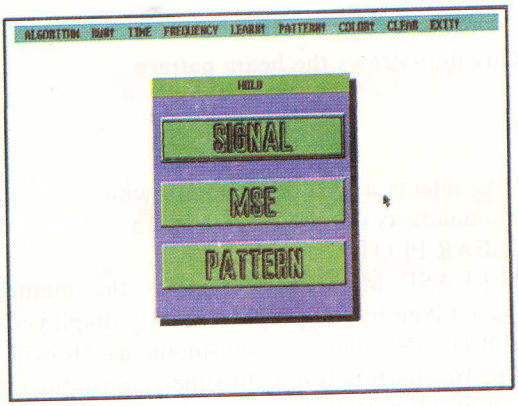

Fig. 16. Superimposition selection under CLEAR.

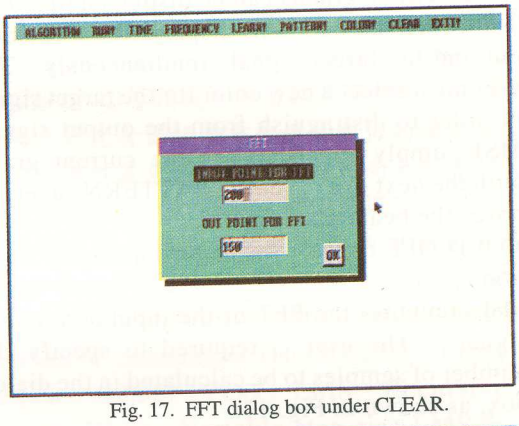

Fig. 17. FFT dialog box under CLEAR.

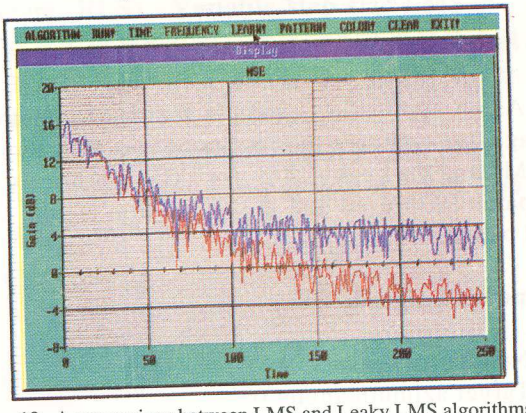

Fig. 18. A comparison between LMS and Leaky LMS algorithms. (- ❑--❑-:Leaky LMS, —: LMS).

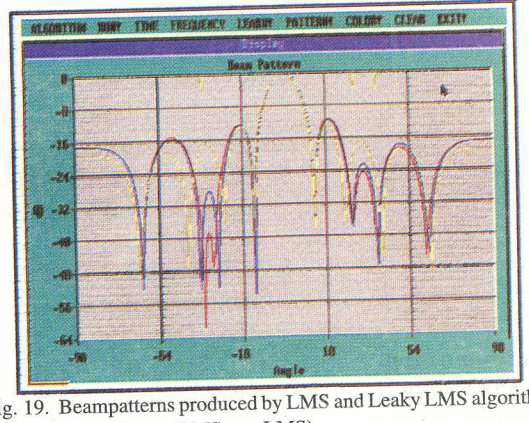

Fig. 19. Beampatterns produced by LMS and Leaky LMS algorithms. (- ❑--❑-:Leaky LMS, —: LMS)

This item draws the beam pattern.

CLEAR

This selects a pull-down menu which contains five commands as illustrated in Fig. 15.

CLEAR PLOT clears the screen.

RELEASE MEMORY releases the memory space taken by the graph currently displayed.

HOLD ON selects a sub-menu as shown in Fig. 16. Its function is to superimpose the current graph with the next graph to be displayed. At most three graphs can be displayed together. The command SIGNAL displays the array output and the target signal simultaneously. The user must select a new color for the target signal in order to distinguish from the output signal. MSE simply superimposes the current graph with the next one. Finally, PATTERN superimposes the beam patterns.

HOLD OFF cancels the superimposition function.

FFT computes the FFT of the input and output signals. The user is required to specify the number of samples to be calculated in the dialog box, as shown in Fig. 17.

SIMULATION EXAMPLE

As an application example for our simulation package, we compare the results of two algorithms, the LMS and the Leaky LMS, for a GSC architecture. The parameters used and the simulation results are listed below.

Parameters assigned:

Number of sensors: 9
Target azimuth angle: 0°
Steering vector angle: 0°
Signal/noise ratio: 10dB
Jammer/directions: -35°, 30°, 40°
Jammer/noise ratio: 40dB for all
Number of samples: 300
Normalized signal frequency: 0.1
Ensemble average: 8
Leaky coefficient: 0.996
Step size: 0.000001

Simulation results:

When adjusting the optimal weight for the array, the added leaky coefficient can speed up the convergence. This can be seen from Fig. 18. Fig. 19 is a comparison between beam patterns produced from using LMS, Leaky LMS, or without adaptation at all.

CONCLUSION AND FUTURE WORK

In this article, we have presented the theoretical basis and the design of an adaptive beamforming simulation package. We believe that, by implementing a collection of available beamforming architectures and algorithms, both the students learning the theory and the practitioners wanting to quickly establish a model can benefit from this package. For ease of extension to cover new algorithms and beamforming architectures, we employ the concept of design modularization. Also, for portability reasons, this package is totally written in C language.

Besides being a pure simulation tool, this package can be converted into a development system for realistic beamforming applications, provided that proper A/D hardware and software interfaces are in place. Another possibility for future development is to construct a specialized beamforming library, based on the software routines already in the package, for users who want to write programs and develop their own models. A pre-packaged library with a clearly defined programming interface will definitely ease and speed up their development process.

ACKNOWLEDGEMENT

Funds for this research was sponsored by National Science Council, Taiwan, R. O. C. under grant NSC 82-0209-E-019-064.

REFERENCES

1. S. Haykin, *Adaptive Filter Theory*, third edition, Prentice-Hall Inc. (1996).
2. P.M. Clarkson, *Optimal and Adaptive Signal Processing*, CRC. (1993).
3. B. Widrow, and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall Inc. (1985).
4. K.C. Huang and S.H. Chang "An efficient structure of broadband beamforming," *Proceeding of National Science Council-Part A*, Vol. 13, pp. 397-404 (1989).
5. N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series*, MIT Press, Cambridge, MA (1949).
6. B. Widrow, L.J. Griffiths, and B.B. Goode, "Adaptive antenna systems," *Proc. IEEE*, Vol. 55, pp. 2145-2159, Dec. (1967).
7. O.L. Frost, III, "An algorithm for linearly constrained adaptive array processing," *Proc. IEEE*, Vol. 60, No. 8, pp. 926-935, Aug. (1972).

8. L.J. Griffiths, and C.W. Jim, "An alternative approach to linearly constrained adaptive beamforming," *IEEE Trans. on Antennas and Progagat.*, Vol. AP-30, pp. 27-34, Jan. (1982).
9. M.H. Er, and A. Cantoni, "Derivative constraints for broad-band element space antenna array processors," *IEEE Trans. on Acoust., Speech, Signal, Processing*, Vol. ASSP-31, pp. 1378-1393, Dec. (1983).
10. N.K. Jablon, "Adaptive beamforming with the generalized sidelobe canceller in the presence of array imperfection," *IEEE Trans. on Antennas and Propagat.*, Vol. AP-34, No. 8, pp. 996-1012, Aug. (1986).
11. N.K. Jablon, "Steady state analysis of the generalized sidelobe canceller by adaptive noise cancelling techniques," *IEEE Trans. on Antenna and Propagat.*, Vol. AP-34, No. 3, pp. 330-338, Mar. (1986).
12. K.C. Huang and S.H. Chang "A modified Griffiths-Jim adaptive beamformer based on Givens rotation using the systolic array," *Signal processing V: Theories and Application*, pp. 1591-1594, Sept. (1990).
13. K.M. Buckley, and L.J. Griffiths, "An adaptive generalized sidelobe canceller with derivative constraints," *IEEE Trans. on Antennas and propagat.*, Vol. AP-34, No. 3, Mar. (1986).
14. M. Morf, and D.T. Lee, "Recursive least squares ladder forms for fast parameter tracking," *Proc.* 1978 *Conf. Decision Control*, San Diego, Calif., pp. 1362-1367 (1978).

可適性波束構成器模擬軟體之設計與實現

張順雄　呂紹偉　梁興旺

國立台灣海洋大學電機工程學系

摘　要

本文的主要目的，就是建立一套模組化的可適性波束構成器(Adaptive Beamforming)電腦模擬軟體。現今可適性波束構成器已被廣泛地應用於雷達、聲納、通訊、地球物理探勘(Geophysical Exploration)等領域。但由於不同的陣列架構與演算法則，研究者需經常重覆撰寫類似的模擬程式。而對於可適性波束構成器，國內外尚未有專門的套裝軟體，一般皆針對特定主題個別撰寫模擬程式。故本文收集較具代表性的陣列架構與演算法則，如Generalized Sidelobe Canceller (GSC)、Frost、Derivative constraints、High order main beam derivative constraints (HOMBDC)、Least Mean Sequare (LMS)、Leaky LMS、Recursive Least Square (RLS)、Variable step size LMS等。加上功能表式的人機介面，建立一套可適性波束構成器模擬軟體。如此將大量縮短研究發展時間，亦可幫助初學者印證理論的正確性。

本套模擬軟體為求高度的可攜性，特以C語言撰寫，且將人機介面與軟體核心獨立為二個模組，以便日後移植至其他系統。