AdaptLab: Adaptive Signal Processing Environment for Education

R.W. Stewart, P.Aaserud

Signal Processing Division
Department of Electronic and Electrical Engineering
University of Strathclyde
Glasgow G1 1XW, UK
E-mail bob@spd.eee.strathclyde.ac.uk

Abstract

In this paper we present a software package to be used for the teaching adaptive signal processing. The aim of the *Adapt-Lab* software is to allow students to understand the fundamentals of real world adaptive signal processing by presenting adaptive algorithms, architectures and applications in a visual form. AdaptLab allows the single channel architectures of system identification, inverse system identification, noise cancellation, prediction and filtered-X implementations to be set up. The user can select adaptive filter type and length, algorithm parameters (such as step size, forgetting factor, leakage, update rate, etc), and an appropriate input signal from a signal generator or the on-disk signal library of typical input signals. The simulation can be started and the user can open signal windows (digital oscilloscope or spectrum analysis) at various points in the algorithm architecture. The simulation can be frozen or stopped at any point and algorithm parameters updated, or the simulation signal results and parameters stored.

1. Introduction

In this paper we present an overview of an in-house adaptive signal processing package, *AdaptLab*, to be used for the teaching of a 24 hour lecture, and 12 hour recitation class on *Adaptive Signal Processing*. AdaptLab has been developed over the last 3 years in response to aspects of the class that students clearly find difficult to understand or visualize, and is designed to be an easy to use self learning package for students to use for learning reinforcement and for research and development at a later stage.

Since 1990 the University of Strathclyde, Department of Electronic and Electrical Engineering have run a Master of Science (MSc) degree course in *Communications, Control and Digital Signal Processing* (CCDSP) run over three semesters in each calendar year. The prerequisite for the course is a Bachelor degree in Electronic Engineering. In the first semester the students are given two fundamental courses in each of the core subject headings of commmunications, control and DSP. In the second semester students undertake one additional class in the core areas, and choose the remaining three options from a list of six options. The compulsory DSP class in the second semester is *Adaptive Signal Processing*.

Although students on the CCDSP master's course are formally taught C programming, and use Matlab and Hypersignal software, these packages have not proven to be particularly useful for adaptive signal processing simulation and visualisation. For example, although Matlab is a very powerful tool and can easily be programmed to perform all required algorithms in adaptive signal processing, the presentation of the subtle effects of adaptive signal processing parameters is not straightforward. Similarly Hypersignal Block Diagram provides an excellent simulation interface, however the programming effort to provide the vast wealth of adaptive algorithms, architectures and parameters is not trivial. Therefore since 1991, we have been developing first an adaptive algorithm library, and more recently an easy to use Windows 3.1 user interface to ensure that students can perform useful adaptive DSP within a few minutes of using the software.

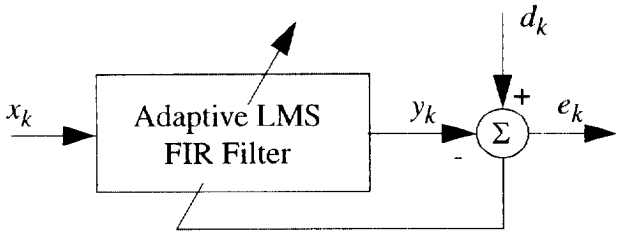
2. Adaptive Signal Processing Course

The aim of the Adaptive Signal Processing course is to present algorithms and architectures for adaptive signal processing for application to real world signal processing problems. As prerequisites it is assumed that students have studied discrete and continuous signals and systems, and introductory linear algebra. The syllabus of the course can be summarised as:

1. **Introduction to adaptive filtering:** a historical perspective; a state of the art perspective.
2. **Statistical signal processing (Revision):** Correlation; Erodicity; Means, variances; Stationarity. Wide sense stationarity; Periodogramme; Frequency response vs. Power Spectrum.
3. **Matrix algebra (Revision):** Addition, multiplication and matrix inverses; Properties of the correlation/ covariance matrix; eigenvalues and eigenvectors. QR algorithm

1058-6393/95 $4.00 © 1995 IEEE

4. **Wiener filter theory:** normal equations; error performance surfaces; orthogonality; minimum mean square errors.
5. **The least mean squares (LMS) algorithm:** formulation; convergence; stability criteria. Algorithm variations: normalized, sign error, sign data, leaky, filtered-X LMS.
6. **Applications of the LMS:** System identification ; room acoustics, control systems; inverse system modelling; modems, telecommunications adaptive equalisation, echo cancellation; adaptive beamforming (radar, sonar, hearing aids, listening devices); active noise cancellation systems in cars, aeroplanes, medical systems, communication systems.
7. **Recursive LMS-IIR Algorithms:** Output Error Formulation; Equation Error forumulation. Full gradient, simplified gradient, SHARF, Feintuch's algorithm. Applications of recursive LMS algorithms.
8. **Frequency Domain Adaptive LMS:** Architectures, advantages, and disadvantages.
9. **General Least Squares Solution:** Least squares solution of general adaptive system. QR algorithm solution.
10. **Recursive least squares (RLS) algorithm:** RLS formulation; forgetting factors; practical implementations; QR based RLS; numerical stability and integrity issues.
11. **Adaptive Lattice Filters:** Gradient lattice, RLS lattice.
12. **Non Linear Adaptive Filters:** Simple LMS neuron, Adaptive Polynomial filters (Volterra).
13. **Comparitive analysis:** Wiener; LMS-FIR, LMS-IIR; RLS, Lattice; Frequency Domain; and Neural Networks for adaptive signal processing applications
14. **Implementation of adaptive filters:** DSP microprocessor implementation; software; custom hardware.



$$y_k = \sum_{n=0}^{N-1} w_{nk} x_{n-k} = \boldsymbol{w}_k^T \boldsymbol{x}_k$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + f(\boldsymbol{x}_k, \boldsymbol{w}_k, e_k)$$

$$\boldsymbol{w}_k^T = [w_0\ w_1\ \cdots\ w_{N-1}]_k \quad \boldsymbol{x}_k^T = [x_k\ x_{k-1}\ \cdots\ x_{k-N+1}]$$

Figure 1: General adaptive signal processor.

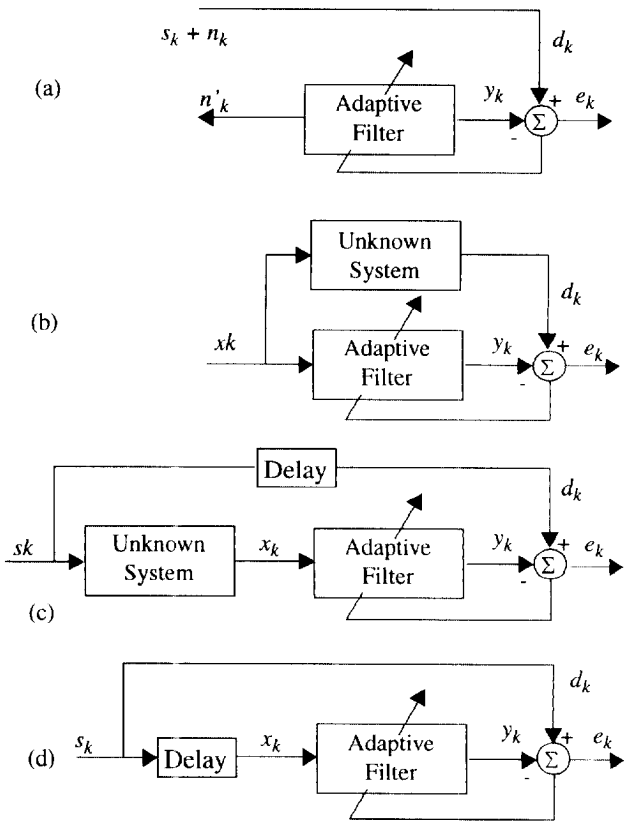


Figure 2: General adaptive applications of (a) Noise Cancellation, (b) System identification, (c) Inverse System Identification (d) Predcition.

No text book is used for the course, however students are refered to the well known texts [2], [3], [4], [1].

3. Course Presentation and Chronology

Students are first introduced to the general adaptive model shown in Figure 1. The aim of *all* adaptive signal processing algorithms and techniques is to make the error, e_k, as small as possible. This must be done by *adapting* the signal x_k, such that the filter output y_k is *very similar* to some desired signal d_k. It is straightforward to show students that the only mathematically tractable way forward is to minimize the squared error, or the mean squared error. From this model the four main (single channel) adaptive applications of Figure 2(a)-(d) can be presented and described for the applications presented in syllabus Section 2.6.

3.1 Minimum Mean Squared Error

To a student with an understanding of basic stochastic signal processing and digital filtering, the derivation of the Wiener-Hopf solution is straighforward. The mean square error (MSE) of Figure 1 is given by:

$$MSE = E[e_k^2] = E[d_k^2] + \boldsymbol{w}_k^T \boldsymbol{R} \boldsymbol{w}_k - 2\boldsymbol{p}^T \boldsymbol{w} \qquad (1)$$

where $\boldsymbol{R} = E[\boldsymbol{x}\boldsymbol{x}^T]$ is the autocorrelation matrix, and

$p = E[d_k x_k]$ is the cross correlation vector. To interpret the important concept of minimum MSE (MMSE) the example of a one weight filter with a parabolic error surface (Figure 1(a)), followed by a two weight filter with a 2-D paraboloid error surface (Figure 1(b)) are presented and

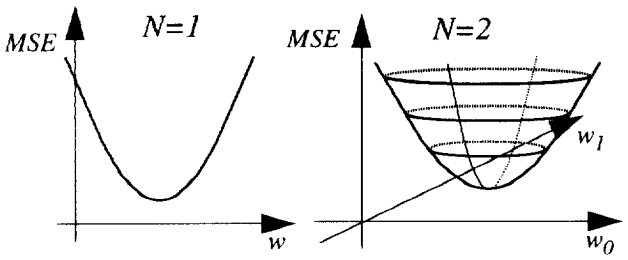


Figure 3: Quadratic error surface.

students then make the geomertrical *leap of faith* by considering 3-D, and higher order hyperparaboloids, all with only one minimum mean squared error point. Finding the zero gradient point, it can be shown that:

$$w_{opt} = R^{-1}p \tag{2}$$

3.2 Gradient Algorithms and the LMS

To introduce gradient techniques the steepest descent equation:

$$w_{k+1} = w_k + \mu(-\nabla_k) \tag{3}$$

is intuitively described with respect to jumping down the inside of the quadratic error surface of Figure 3. To progress to the more practical implementation of the least mean squares (LMS) algorithm, requires that the students accept the available justifications [3] and therefore plausibility of approximating the true gradient vector:

$$\nabla_k = \frac{dE[e_k^2]}{dw} \tag{4}$$

by an estimate thats uses the instantaneous squared error, instead of the mean error, $E[e_k^2]$:

$$\hat{\nabla}_k = \frac{de_k^2}{dw} \tag{5}$$

Thus the well known and omnipotent LMS algorithm is derived:

$$w_{k+1} = w_k + 2\mu e_k x_k \tag{6}$$

Although this approximation of Widrow's [2] may seem obvious, it is of course well documented in signal processing literature that there is no rigorous way to justify it. (Suffice to say, the last 30 years of successful LMS exploitation is perhaps proof enough for most!)

From this point on students can be introduced to the LMS step size (μ) criteria, misadjustment expressions, speed of convergence/time constant issues, and the many suggested variants such as the normalised LMS, Newton techniques, sign error, and so on.

3.3 Recursive LMS IIR Filters

The derivations of the full gradient *output error* LMS filter is mathematically straightforward. However the LMS-IIR is a computationally intensive algoritm, and simplifications can be made to the full gradient algorithm to realise the simplfied gradient algorithm, and further simplification to realise Feintuch's algorithm. There are also variants called SHARF to enhance algorithm performance. The LMS-IIR can also be derived in a completely different form called the *equation error* algorithm which produces a minimum mean square error performance surface that it is quadratic (and therefore has only one minimum), but can introduce misadjustment problems.

3.4 Least Squares and the RLS

Again, from the starting point of Figure 1 the least squares solution can be derived, followed by the more computationally efficient recursive least squares (RLS). The square root RLS, followed by the QR based RLS can then be presented.

3.5 Non-Linear Adaptive Filters

Towards the end of the course, simple neural networks based on LMS neurons are presented. Also the non-linear adaptive Volterra filter is discussed.

4. Course Appreciation

When teaching this course, although many students appear to understand the mathematics of adaptive signal processing, their awareness of how and when to use the various adaptive algorithms were not well developed. When students taking this course were to later undertake an adaptive DSP project, the problems they had highlighted the inadequacy of the lecture material. Typically, they would ask?

- How important is the length of the adaptive filter?
- What value should the step size be?
- When do I use LMS-FIR, LMS-IIR or RLS?
- What is the MMSE value?
- What adapts faster, an LMS, RLS, or IIR LMS
- What happens when a filter goes unstable?
- Does the error really go to zero?

and so on..... While a single sentence and mathematical anwer is always possible, it was clear the learning process would be greatly enhanced a software tool that should allow most questions to be answered by the student, with just a few minutes of reasoned simulation.

5. Adap-Lab Software

The AdaptLab software was written in Borland C++ and runs in Microsoft Windows 3.11. The minimum platform recommended is a 486DX, running at 33MHz,

with 8MB RAM, although it will run on slower systems. Version 1.0 is exclusively for 16 bit platforms, supported for Windows 3.11, and will run equally well on Chicago (Windows 4). Version 2.0 of AdaptLab, however is likely to be exclusively written for Chicago i.e. a 32 bit platform. The software is menu driven, with Figure 4 giving an overview of the available choices.

5.1 Using the Software

From the `Applications` menus (or using the toolbar), one of the four adaptive applications is chosen. Figures 6 to 9 show the AdaptLab plots of the four general adaptive applications. AdaptLab will prompt with a dialog box, which requires that the user choose an algorithm via the radio buttons, and select input signals, unknown filters, and names for output files. On choosing OK, a dialog box for the algorithm chosen will be prompted. As an example, Figure 5(a) shows the system identifcation dialog box with LMS chosen, and Figure 5(b) shows the LMS dialog box that will follow. AdaptLab will then draw the full window diagram of the particular application. From the `Options` menu, the user can choose the arithmetic type, either as floating point or fixed point. To begin the simulation the user chooses from the simulation menu (or again, the toolbar can be used), and the simulation run continously, or in steps of N iterations, with the option to halt at anytime.

5.2 Oscilloscope and Spectrum Analyser

When running a simulation, the aim of AdaptLab is to allow the user to observe the algorithm adapting and save results for later inspection or comparison. Therefore in the main Window of AdaptLab, the signal at each point in the application architecture can be observed, either in the time domain or frequency domain. If the user pushes the triangular push button with the mouse, then the signal flowing through can be displayed (see Figure 9). Drop down menus control time base, volts/division etc or switch to frequency domain. When a simulation is halted, or finished, scroll bars on the signal windows can be used to look back at the signal.

6. Current Development

AdaptLab is currently at the Beta release stage with Version 1.0 being tested. Version 1.1 will have new signal window functionality and will be made available to a graduate DSP class in mid-1995.

File	Application	Simulation	Options	Window	Sig Gen
View ...	System ID...	Start/Stop	Arithmetic...	Cascade	Sine...
Print	Inverse Sys ID...	Run	Config...	Arrange	Square...
Print	Noise Canc...	Run Step...		Icons	Triangular..
Setup...	Prediction ...			Autostretch	Noise...
Export...	Filtered-X...			Stack	Aperiodic...
Open...					Library...
Close					
Save					
Save As..					

Figure 4: AdaptLab file menus (Help menu not shown).

Version 2.0 is currently under development and will include provision for real time implementation and also include filtered-X architectures, A library of signals (in addition to the signal generator) for use with AdaptLab is currently being put together to allow real world relevant simulations to be performed. This library will include standard noise, speech, music, ECGs, vibration signals, impulse response of acoustic environments, telephone lines and so on and is likely to be stored on a CD. The on-line help is also being updated to full hypertext and will provide a full set of course notes, as well as a guide to the software. It is also anticipated that lattice filters and frequency domain adaptive filters will be implemented in Version 2.0. For more information on the software, please contact the authors directly.

Acknowledgements: The authors acknowledge the contribution of Eduardo Aguirre, Viviane Banier, Martin Kristof, and Jaume Juni to the AdaptLab software.

References:

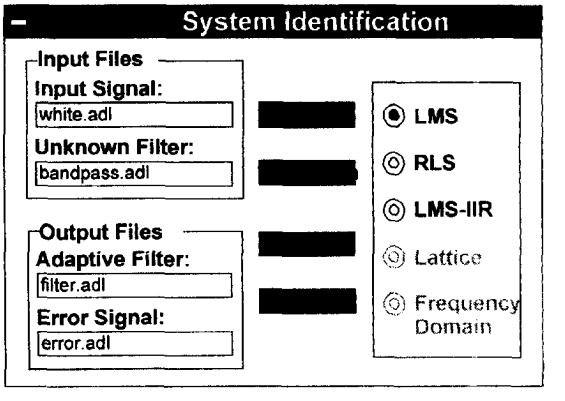
[1] D.J. DeFatta, J.G. Lucas, W.S. Hodgkiss. Digital Signal Processing: A System Design Approach. John Wiley, New York, 1988.

[2] B. Widrow and S.D. Stearns. Adaptive Signal Processing. Prentice Hall, 1985.

[3] S. Haykin. Adaptive Filter Theory. Prentice-Hall, 1991.

[4] C.F.N. Cowan and P.M. Grant (editors). Adaptive Filters.} Prentice Hall, 1985.

(a)



(b)

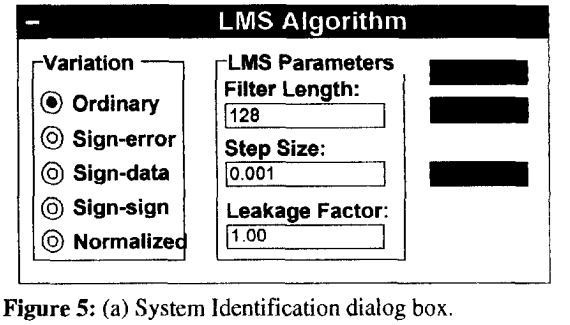


Figure 5: (a) System Identification dialog box.
(b) Chosen LMS algorithm dialog box.

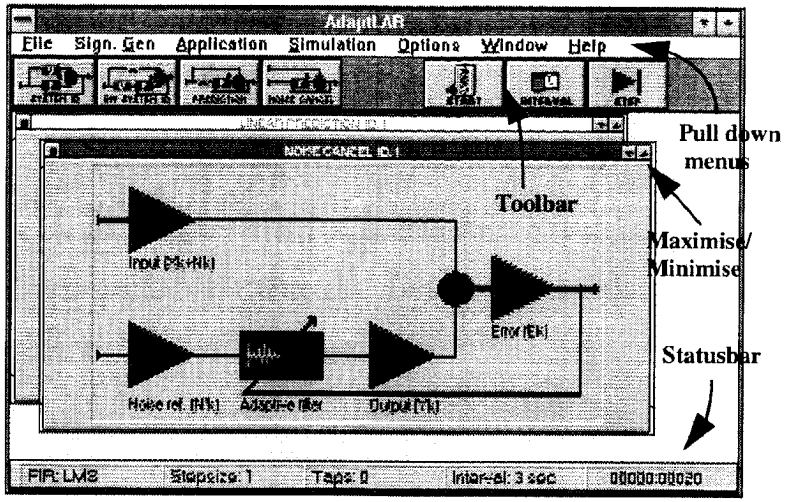


Figure 6: Noise Cancellation

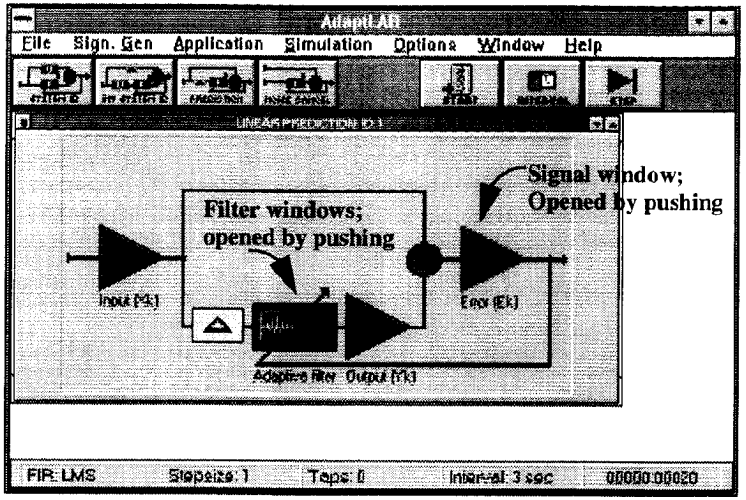


Figure 7: Prediction.

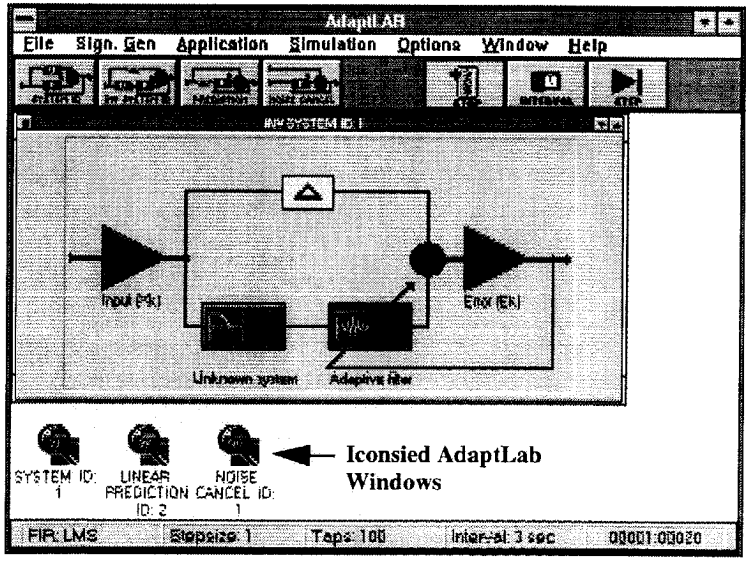


FIgure 8: Inverse System Identification

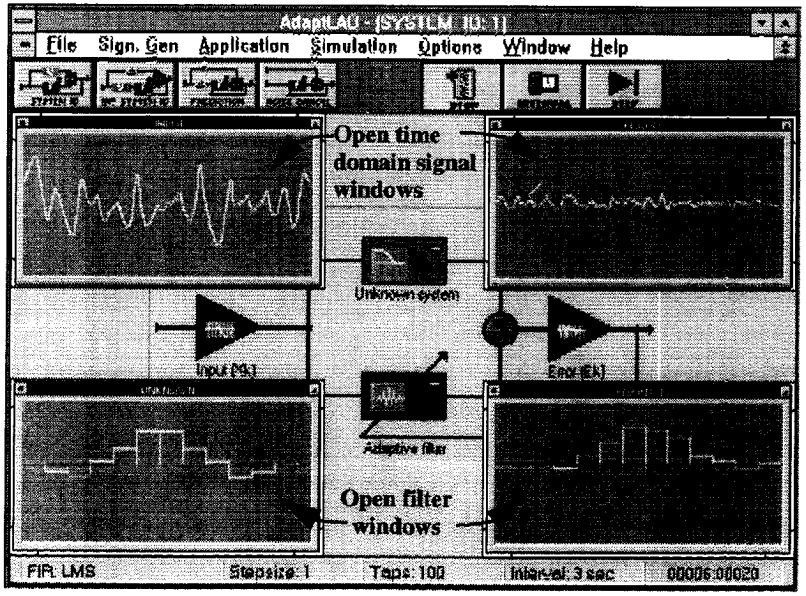


FIgure 9: System identification with signal windows and filter windows opened