ECOOSE: An Echo Cancellation Object Oriented Simulation Environment

Stephen Braithwaite and Ron Addie
University of Southern Queensland
Toowoomba, Australia
braithwa@usq.edu.au, addie@usq.edu.au

ABSTRACT

This paper introduces an object oriented Matlab toolbox which is being developed for the testing and development of echo cancellation algorithms. The obvious objective of an echo cancellation simulation toolbox is to allow *any* simulated audio environment to be used with *any* echo cancellation method, and observed by *any* system for statistical observation and analysis. Such flexibility is hard to achieve. In this paper we explain the advantages of using an object oriented approach, as provided in Matlab 2008, for the design of this toolbox. The design is presented along with some initial results.

Categories and Subject Descriptors

I.6.5 [**SIMULATION AND MODELING**]: Model Development—*Modeling methodologies*; H.5.5 [**INFORMATION INTERFACES AND PRESENTATION (e.g., HCI)**]: Sound and Music Computing—*Signal analysis, synthesis, and processing*; D.2.13 [**SOFTWARE ENGINEERING**]: Reusable Software—*Reusable Libraries*

1. INTRODUCTION

Echo cancellation has been studied for several decades and millions of devices which implement particular instances of echo cancellation algorithms now exist in our communication networks, computers, and telephones. However, as our appetite for interactive video and audio communication over the Internet grows, and becomes more and more a part of the way we work and live, there is a need for ever more sophisticated echo cancellation algorithms. In the study and development of such algorithms, it is essential to read and understand descriptions of algorithms developed by echo cancellation researchers, and to independently evaluate and compare these algorithms, with each other, and with new algorithms which are under development.

A flexible echo cancellation object oriented simulation environment (ECOOSE) has been developed as part of a project to develop echo cancellation for remote auscultation. The software for ECOOSE is written in Matlab 2008 and is available at [1]. This project aims to allow echo cancellation algorithms, recorded sounds, experimental channels (from measurements or purely theoretical), and statistical analysis methods to be specified, interchanged and used together in experiments which investigate (primarily) the ability of the algorithms to cancel echoes. In this way it is hoped that the process of evaluating and comparing echo cancellation algorithms can be promoted and developed as efficiently as possible.

In Section 2 the context of echo cancellation research and development is explained; in Section 3 the reasoning which lead us to choose an object oriented approach is explained; in Section 4 the reason why Matlab 2008 was found to be both necessary and adequate for the development of this toolbox is explained; in Section 5, the design of the current version of the toolbox is described; and in Section 6 some shortcomings of this design are described and a new design is outlined. Finally, conclusions are presented in Section 7.

Simutools March 2008, Rome Italy

2. ECHO CANCELLATION BASICS

Echo management first became necessary in the 1930's when international telephony introduced one way delays of up to 80 milliseconds(e.g. London-Hawaii [3]). Transmission over distances of this length requires amplification, which is only practical in systems which use 4 wires: 2 in each direction. At that time, and still today, telephone systems use just two wires between the local exchange and the residence, so at some point near the residence, usually the local exchange, a conversion between 2 wire and 4-wire communication must take place.

The device which implements this conversion is known as a *hybrid*. Unfortunately, hybrid's inevitably produce an echo, which is very disturbing to audio communication unless the level of the echo signal is reduced to considerably lower than the intended voice signal passing in the same direction or the delay of the echo path is very small (eg 10 milliseconds).

Initially, in the 1940's and 1950's, echo suppression, which inhibits both-way conversation, was used to reduce the impact of echos. The introduction of satellite links caused long delays of up to half a second [3]. This encouraged research into echo management and echo *cancellation* was introduced. Instead of *avoiding* echos by preventing simultaneous two-way communication, echo cancellation estimates the echo and removes it on its way back from the point where it is generated. Bell Laboratories was the first company to patent a practical echo cancelling design [12]. This design or algorithm is commonly known as the Least Mean Squares algorithm.

2.1 Echo Cancellation

Echo cancellation works by modelling the "channel" that acts upon the input giving the delayed echo, e.g. the telephone line, or

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5775
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5775

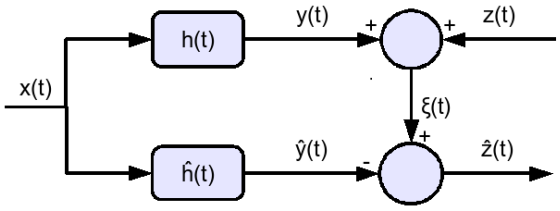

Figure 1: Block diagram for basic echo cancellation.

the speaker/microphone/room combination. The echo canceller automatically develops a model for the channel, and uses this model in combination with the input to predict what the echo will be. It then removes the predicted echo from the output, by subtracting it.

In most cases the channel is linear, time invariant and causal, so that the echo canceller models the channel using a finite impulse response, as illustrated in Figure 1. In this diagram, $x(t)$ represents the speech signal from the remote end. $z(t)$ is the speech signal from the local end, and $\hat{z}(t)$ is the desired result, relatively free from any echoes of $x(t)$. The real impulse response of the echo path channel is $h(t)$ and our estimate of h, used in the cancellation of the echo, is denoted by $\widehat{h}$. The equation connecting z, h, x and $\widehat{z}$ is as follows:

$$\begin{aligned} \widehat{z}_j &= z_j + (h*x)_j - (\widehat{h}_j * x)_j, \quad j = 0,1,\ldots, \\ &= z_j + ((h-\widehat{h}_j)*x)_j, \quad j = 0,1,\ldots. \end{aligned} \tag{1}$$

We denote by $y(t)$ the signal given by the convolution $y(t) = h(t)*x(t)$ and we set $\widehat{y}(t) = \widehat{h}(t)*x(t)$.

2.1.1 The Least Mean Square algorithm

If we let $\widehat{\mathbf{h}}_i$ denote the vector $(\widehat{h}_0, \widehat{h}_0, \ldots)'$, sampled at time i, and $\mathbf{z}_i$ denote $(z_i, z_{i-1}, \ldots)'$, the Least Mean Squares algorithm, after initializing $\hat{\mathbf{h}}_0$ to zero, uses the iterative step:

$$\hat{\mathbf{h}}_{i+1} \leftarrow \hat{\mathbf{h}}_i + k\,\hat{z}_i\,\mathbf{x}_i \tag{1}$$

where k is an arbitrary constant between 0 and 1.

White noise is the ideal excitation for the LMS algorithm and in fact the algorithm will not converge to the true value of $\mathbf{h}$ if $\mathbf{x}$ is frequency deficient [19].

Many patents for incremental enhancements were applied for and awarded after the basic LMS algorithm.

One of the most important enhancements to the algorithm is the introduction of a normalisation term, first used by Nagumo and Noka in 1967 [15]. Since y and $\hat{y}$ are both proportional to $\mathbf{x}$, our update is proportional to $|\mathbf{x}|^2$, i.e. the square of the volume. At low volumes, (i.e. for a quiet talker), the basic LMS algorithm will only converge very slowly; if we make k large in order to compensate for this problem, at high volume (loud talker) convergence could become unstable.

This is fixed by adding a normalisation term.

$$\hat{\mathbf{h}}_{i+1} = \hat{\mathbf{h}}_i + \frac{k\,\hat{z}_i\,\mathbf{x}_i}{\mathbf{x}^T\mathbf{x} + \Delta}$$

The Δ in the denominator prevents the algorithm from becoming unstable when $\mathbf{x}$ is very small. The normalisation term itself can be updated rather than recomputed, so it does not represent any substantial computing overhead. This version of the LMS algorithm may be more explicitly referred to as the Normalised Least Mean Squares (NLMS) algorithm.

2.1.2 Recursive Least Squares

Godard by applied the Kalman filter to the problem of adaptive equalization [7]. By doing this, he introduced the Recursive Least Squares (RLS) algorithm. Subsequently Gitlin and Magee derived the same algorithm using Woodbury's inversion lemma [8] [6].

The RLS algorithm features fast convergence and can achieve up to 40 dB of echo cancellation in practice [9]; it also converges well if the input is frequency deficient [11]. It has complexity of order N^2 however "fast" versions of the RLS aglorithm have been developed which can perform a single iteration in 8N operations, as opposed to 2N operations with the NLMS algorithm [11].

In general, the channel is not fixed and information from the recent past is more important than information from the distant past. Weightings are used in practice, so that the effect of information from the past reduces exponentially with time [9].

2.1.3 The Affine Projection Algorithm

The Affine Projection (AP) algorithm is an attractive compromise between the NLMS and the RLS algorithm [16]. An AP algorithim of order L lies in between the two. It uses the L most recent innovations, with L being a small number typically in the range 2 to 5.

The performance of the AP algorithm approaches the performance of the RLS algorithm in practice [9]. Fast version of the AP algorithm exist, and their numerical complexity approaches that of the NLMS algorithm.

2.1.4 Non Linear Echo Cancellation

While most echo channels are linear, distortion such as speaker clipping can easily introduce non linearity into the channel, which can nevertheless still be cancelled by a suitably designed echo cancellation algorithm [2]. A general class of non linear echo cancellation algorithms based on the Volterra series is described in [18].

3. JUSTIFICATION OF THE DEVELOPMENT OF AN OBJECT-ORIENTED TOOLBOX

Matlab is an attractive platform for the implementation of the algorithms because it is a high level language with a wide range of medium level operations immediately available.

Mathworks Simulink is a popular platform for performing the simulation of echo cancellation [4] [17]. Simulink allows the modelling of echo cancellation blocks in a natural and obvious way, and reveals the end results.

However, the concept of *channel* with which Simulink deals may not be adequate for certain types of experiments. For example, when there is frequency deficient input, it is possible for algorithms such as LMS to converge on an echo path channel estimate such that the echo is cancelled *without* convergence to the true channel. It may be instructive to see whether the algorithm finds the correct model , as well as correct results. Also, it can be instructive to see how different parts of an estimate converge. An example is shown in Figure 2 where the non linear part of the channel estimate actually gets worse, initially. We chose to use a simulation that allowed us to monitor the convergence of the *model itself*, which made the use of simulink impractical.

Development of a new *object-oriented framework* (in the sense defined in [10]) for echo cancellation experiments has allowed us, in particular, to incorporate facilities which compare the internal state of an estimate with the actual model.

We originally implemented an echo cancellation toolbox using only the procedural features of Matlab in 2007. Interfaces for the

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5775
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5775

various elements of a simulation were defined, so that they could be interchanged as needed. This earlier toolbox had the following elements as well as code that allowed the elements to be selected and run:-

- A directory of sound files in waveform audio format.
- A directory of linear channels in waveform audio format
- A directory of non linear distortions in the form of Matlab functions.
- A directory of echo cancellation algorithms in the form of Matlab functions,
- A directory of experiment specifications or run files in the form of small Matlab functions. Each file performed one experiment or run.

This first implementation was unsatisfactory because it was not flexible enough to meet the primary goal of the project. Any experiment could be carried out, but it was necessary to meddle with the code for algorithms and other elements in order to do so. Examples of the way the first implementation lacked flexibility are as follows:

- It was difficult to separate the algorithm from the channel. Ideally, the algorithm should not already know the channel, but should have to discover the channel for itself. But progress, typically in the form of euclidean distances, needed to be monitored at each iteration, and that required knowledge of the channel. If the algorithm state variables were not to be passed out of the algorithm in order to save the values between calls, it was necessary for the algorithm to be provided with the information about the channel and do those calculations with each iteration.

  Because the algorithm and the channel were not separated, it was difficult to try out random combinations of channel and algorithm, e.g. trying a non linear channel with a linear algorithm, or a linear channel with a non linear algorithm.

- Each new algorithm seemed to impose new demands on the algorithm interface. The algorithm interface had to be expanded at each turn. In order to allow interchangeability, the changes in the interface had to be propagated back to the existing algorithms. The only alternative to this would be to have a huge, all encompassing interface with every need anticipated.

- Different algorithms required different parameters, and in order to cater for this, it was necessary to format arguments into a string and implement parsing. In practice, it was so much easier to hack, changing the parameters in the code itself.

- Algorithms had to be modified in a major way to model a change in the parameters of a channel during a run.

4. OBJECT ORIENTED PROGRAMMING IN MATLAB 2008

We looked for an alternative way to implement the echo cancellation test harness. We considered implementation in a 3rd generation object oriented language such as C++. Using C++ would allow us to solve the above problems, but we would lose the easy access to well developed well tested mathematical primitives offered by Matlab. Instead, we would have to search for them or implement them. Then, of course, we would have to test them ourselves.

We considered using the object oriented capabilities offered by Matlab 2007, but soon discovered that it was too limited. No references are possible to Matlab 2007 objects and Matlab 2007 objects are not mutable. Matlab 2007 permits only objects that need to be recreated each time a value in the object is modified.

The object oriented capabilities available in Matlab 2008, on the other hand, are sufficient to justify taking an object-oriented approach in the development of a toolbox [14]. In Matlab 2008 it is possible to have objects that can be referenced from elsewhere in a program and to have methods that alter only some of the properties of an object[13].

Properties and methods can be private, protected or public, as one might expect in an object oriented language. Constructors, destructors, abstract classes, static methods and even multiple inheritance are all part of the language[13].

Objects that are no longer referenced are automatically destroyed, which seems appropriate to a high level application-oriented language such as Matlab (although currently, memory space associated with destroyed objects is not re-used, unless one manually destroys *all* variables [5]).

We opted to upgrade to Matlab 2008 and recreate the echo simulation environment from scratch.

5. DESIGN OF THE OO ECHO CANCELLATION TOOLBOX

5.1 Requirements

The echo cancellation toolbox will be used to develop echo cancellation algorithms and to characterise existing echo channels.

The toolbox should be able to take any local or remote sound in the form a waveform audio input.

The toolbox should be able to handle the following types of channels:-

- Static Linear Channel
- Channel with a timewise discontinuity.
- Slowly changing linear channel.
- Channel preceded by non linear instantaneous distortion.
- General non linear channel.

The toolbox should be able to handle the following types of algorithms:-

- NLMS algorithm.
- RLS algorithm, and fast RLS algorithms.
- AP algorithms, fast AP algorithms.
- Block based time domain algorithms.
- Block based frequency domain algorithms.
- Algorithms that work with a channel preceded by non linear instantaneous distortion.
- The Volterra general non linear algorithm.

The toolbox should be able to generate the following types of output.

- Graphical depiction of algorithm convergence.
- Numerical indicators of obtained convergence.
- File output of obtained channel.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5775
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5775

5.2 Object-Oriented Design of an Echo Cancellation Simulation Toolbox

Figure 3 is a class diagram which shows the relationships between the various classes. There is an abstract class for the channel, from which real channels are derived and also an abstract class for the algorithm from which other algorithm classes are derived.

5.3 The Algorithm Classes

The algorithm class is responsible only for the algorithm, so that the development of an algorithm can be performed with minimal effort.

The algorithm classes are derived from the base algorithm class, algBase. The algorithm classes do not know the channel. It is the job of the algorithm to discover the channel. An important abstract method of the base class is called next(), which is called for each iteration and has as its arguments, just the current value of x and the current value of ξ, just as a real echo canceller would have. The algorithm classes make the results available by means of methods. One of the abstract methods of the base algorithm class is getY(), which returns the current estimated result from the echo channel y.

5.4 The Linear Algorithm Class

The linear algorithm class is a specialization of the base algorithm class. Most of the echo cancellation algorithm classes are linear, and will be derived from this class. Since linear channels are characterised by their impulse response, getH(), which returns the impulse response is one of the methods of this class. setH() which allows the current value of the impulse response to be set artificially, is also a method of this class.

5.5 The Channel Classes

The channel class knows the characteristics of the channel, and knows how to calculate the real value of y from the values of x. The channel class, calculates $\xi = y + z$, and provides the algorithm with x and ξ. The channel will use a method provided by the algorithm to obtain the estimated value of y, and will use it to report the errors to the output classes such as the graphing class.

5.6 Algorithm Delay

The estimate of y is normally produced by convolving x with the estimated channel. This is instantaneous if this is implemented in hardware, but this is not the most efficient method of calculating y in a digital algorithm. The fast Fourier transform actually makes it possible to convert x and $\hat{h}$ to the frequency domain as blocks, perform the calculations in the frequency domain which become trivial, and convert the results back to the time domain with less computational time than it would have taken to perform the calculation directly in the time domain.

These frequency domain algorithms generally require information to be calculated as a block, and such block based echo cancellation algorithms impose a blocking delay. The algorithm knows how long this delay will be and has a method which returns the delay measured in discrete time periods to the caller. At each iteration, a request to the algorithm for the y value will return the estimated value of for y at that many time periods ago. The channel, having obtained the delay from the algorithm performs its comparisons with the real value of y that many time periods ago.

6. FLAWS IN THE FIRST DESIGN, AND THEIR SOLUTION IN THE NEW DESIGN

6.1 Problems in the existing Channel Class

Ideally the channel class would be responsible only for the representing the channel. In the current implementation, however, the channel class is not only responsible for calculating the real channel response to x, it is currently used to calculate and plot the differences between the real channel characteristics and the estimated characteristics over time. This was done because the channel class is well placed to know what can be graphed, and well placed to calculate the difference between the real channel characteristics and the estimated channel characteristics, so it was convenient to have the channel class perform both tasks.

Unfortunately it became apparent that the short term advantages of this approach are a small benefit, but the long term cost of including functions which are not really part of a channel in the channel class is too serious to be tolerated. For example, temporary code modifications were required in the algorithm class itself in order to create the graphs shown in Figure 2 for the purpose of exploring a non linear echo cancellation algorithm. These graphs show the performance of both a linear and a nonlinear algorithm applied to both a linear and a non-linear channel, and demonstrate the need to arbitrarily mix and match channels and algorithms.

In designing classes representing linear and non-linear channels and algorithms we face a challenge in O-O design: naturally, the linear channel understands how to interface to linear algorithms. Likewise, the non linear channel understands how to interface with the non linear algorithms. The linear channel can interface with the non linear algorithm class without difficulty, because although the methods which are exclusive to the nonlinear algorithm class won't be called, this is not a problem. However, the non-linear channel cannot interface with the linear algorithm, without a workaround, because it calls methods of the non-linear algorithm that the linear algorithm does not implement.

This is illustrated by the algorithm being explored in Figure 2. The linear channel and the linear algorithm are fully characterised by an impulse response, while the non linear channel and the non linear algorithm were characterised by both a linear channel and a distortion polynomial.

6.2 Solution

The solution to this, shown in Figure 4, is to take the functionality that the class called channel now has, and split it into two class hierarchies. One of them will be a hierarchy of channel calculator classes, and the purpose of channel calculator classes will be to calculate y, given x. The other will be a hierarchy of algorithm observer classes. An algorithm observer class is responsible for obtaining the estimated echo characteristics from the algorithm class at each time step, comparing them known echo characteristics, and prepare, at a high level, final results such as graphs. The algorithm observer needs to know the details of the channel expressed as nearly as possible in the same model that the algorithm class uses, even if this model would be inadequate for calculating the real result, y.

The base algorithm observer class will be able to observe all algorithm classes, because it only observes the estimated $\hat{y}$, and compares it to the real y as calculated by the channel calculator class.

The linear algorithm observer class will observe the estimated impulse response, $\hat{h}$ as well as estimated result, $\hat{y}$. It will be able to observe all algorithms where the characterisations include the impulse response h. Note that the channel itself may be linear or non linear, now that the channel calculator has been separated from the observation.

6.3 Dynamically Updating the Channel

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5775
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5775

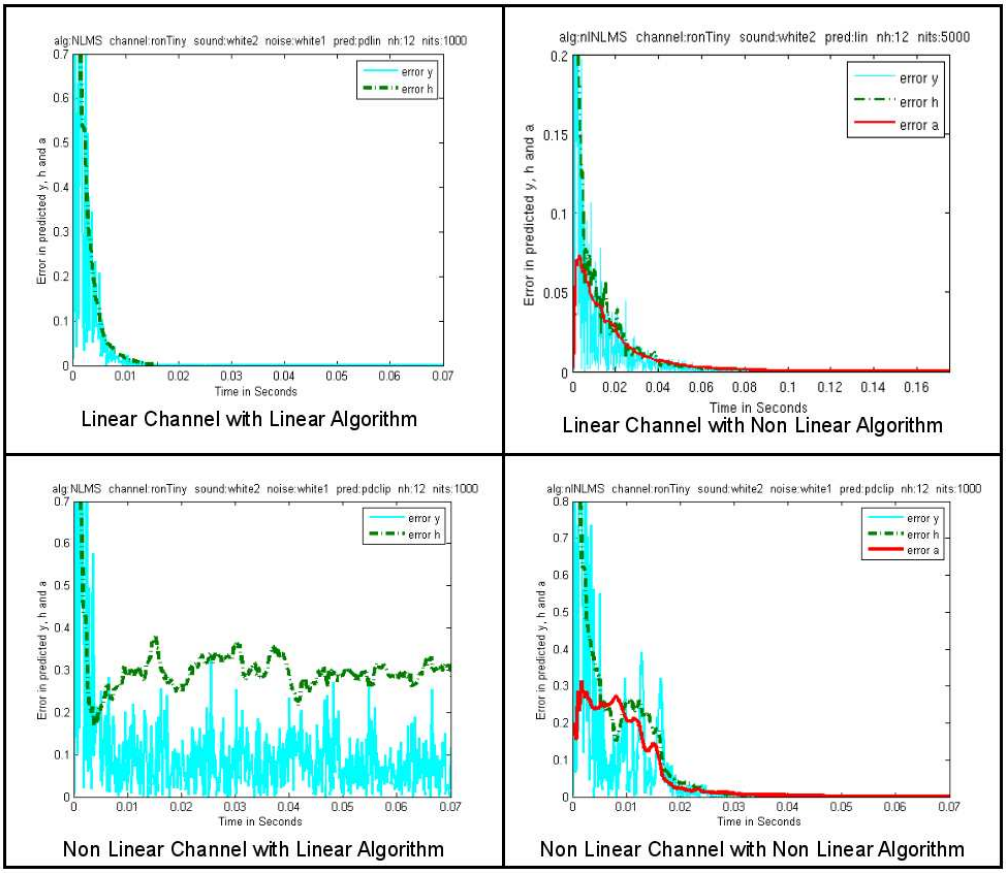

Figure 2: Linear/Nonlinear Channels with Linear/Nonlinear Algorithms

Both the channel class and the algorithm observer class need to know about the channel. From the requirements, ECOOSE must be able to handle a channel that changes slowly, or is subject to sudden change during the course of the simulation. It is therefore highly desirable to have the true source of the channel characteristics in a single class. Failure to do this could easily result in wrong results, because the algorithm observer's understanding of the channel characteristics could easily diverge from those of the channel calculator. Therefore we need a hierarchy of channel source classes.

Having the channel source class call methods in the algorithm observer class and the channel calculator class would not not be a flexible arrangement, because then channel source class would have to make exactly the right calls, i.e. it would have to be matched to both the algorithm observer class and the channel calculator class.

Having the algorithm observer class and the channel calculator class call methods of the channel source class is a much better arrangement. The channel source class must have methods that provide the information for the channel calculator class and the algorithm observer class, i.e. it must have its knowledge in both models. But it need not be an exact match to the algorithm observer class and the channel calculator class, as it can potentially have methods that are not called.

The channel source class knows when the channel has changed, i.e. it knows when the algorithm observer class and the channel calculator class need to call the channel source class to update the channel. When the channel changes, the channel source class will notify these classes via a method, in order to tell them that they need to update.

7. CONCLUSION

The development, analysis, and comparison of echo cancellation algorithms has reached a stage where it is essential that independent and objective tests of algorithms need to be carried out in an agreed manner. The toolbox presented here is publicly available (at [1])and can be readily used by anyone undertaking research in this field. Researchers who wish to propose new algorithms or analyse new ones are welcome to provide Matlab definitions of an algorithm class which can be used with this toolbox, by them or by others, thereby providing objective evidence regarding its performance.

The creation of a flexible toolbox for the simulation of echo cancellation proved difficult to achieve using procedural technology. It proved impractical to separate the algorithm from the channel. Accommodating the extra parameters that each algorithm may have was hard work. A dynamically changing channel would have been a programming challenge. The adoption of object oriented technology will solve these problems, and also give us flexibility without the penalty of having to re-implement all the features of an audio channel or of an algorithm if most of its features are similar to one

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5775
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5775

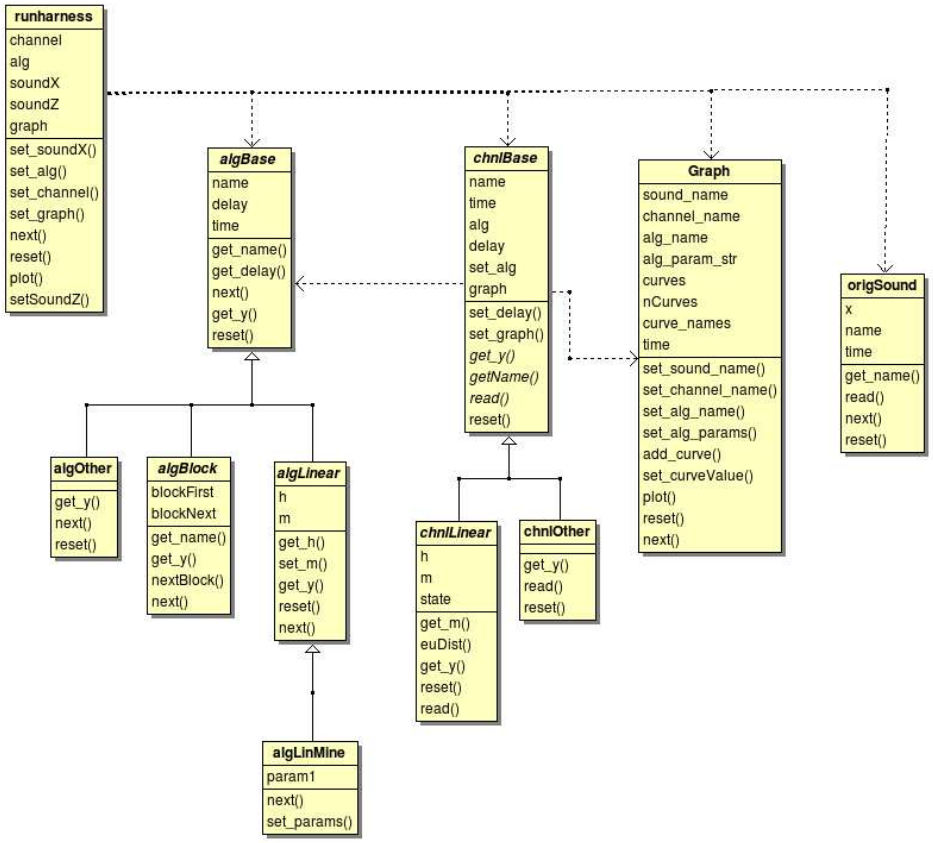

Figure 3: Class Diagram for current design of the Echo Cancellation Object-Oriented Simulation Environment (ECOOSE)

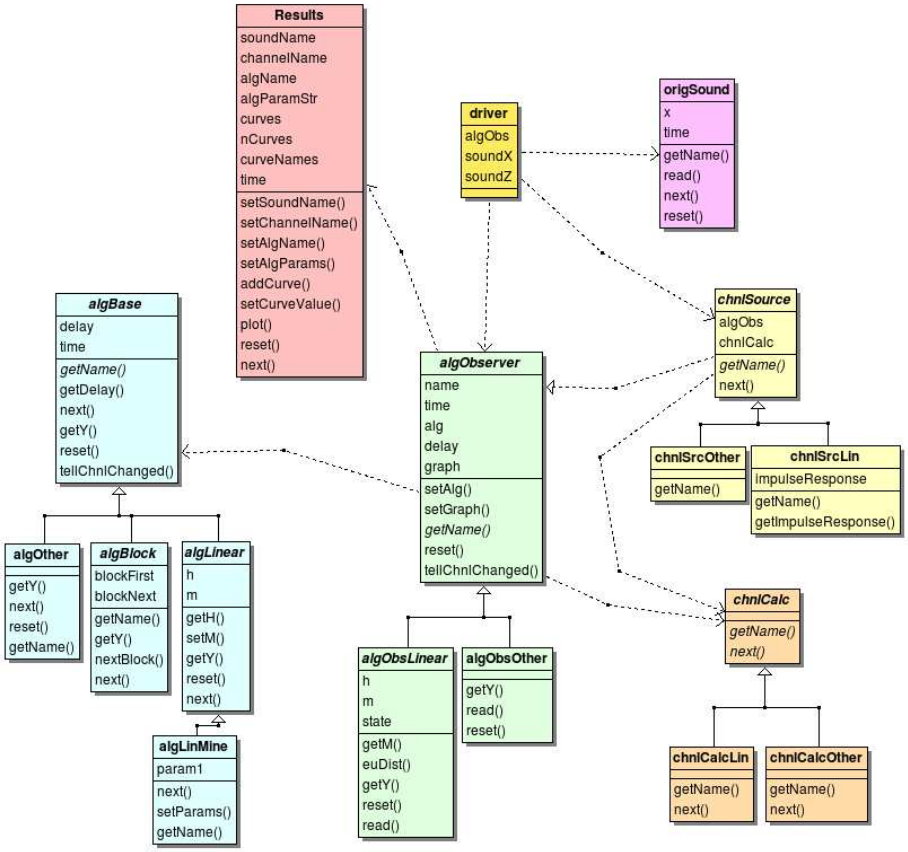

Figure 4: Class Diagram for the improved design for ECOOSE

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5775
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5775

already implemented.

In developing an OO framework, as discussed in [10], we need to think carefully about its design so that a wide range of applications can readily be accommodated. In this paper we have explained some of the difficulties, described our first design, and also an improved OO design for our framework/toolbox which we feel can adapt to the present and future needs of designers of echo cancellation algorithms. Figure 2 shows how the simulation environment has been used, with channels and algorithms having been mixed and matched arbitrarily, allowing us to fully investigate the strengths and weaknesses of the algorithms under study.

8. REFERENCES

[1] Ron Addie and Stephen Braithwaite. Ecoose home page. Web Page, 2008. URL: http://www.sci.usq.edu.au/projects/ecoose.

[2] Ron Addie and Stephen Braithwaite. Non-linear echo cancellation - a bayesian approach. In *Proceedings of the International Conference on Signal Processing and Communication Systems*, Dec 2008. URL: http://icspcs2008.trackchair.com.

[3] J.W. Emling and D. Mitchel. The effects of time delay and echoes on telephone conversations. *Bell Systems Telephone Journal*, 22:2869–2891, Nov 1963. URL: .

[4] G. Eslinger and J. Dixon. Dynamic adjustment of system parameters improves echo and noise cancellation. Technical report, Texas Instruments, 2006. URL: http://focus.ti.com/lit/wp/spry094/spry094.pdf.

[5] D. Foti. Inside matlab objects in r2008a. *Matlab Digest*, Sep 2008. URL: http://www.mathworks.com/company/newsletters/digest/2008/sept/matlab-objects.html.

[6] R.D. Gitlin and F.R. Magee. Self-orthogonalizing adaptive equalization algorithms. *IEEE Transactions on Communications*, 25(7):666–672, Jul 1977. URL: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1093886.

[7] D. Godard. Channel equalization using a kalman filter for fast data transmission. *IBM Journal of Research and Development*, 18(3):267–273, May 1974. URL: http://www.research.ibm.com/journal/rd/183/ibmrd1803I.pdf.

[8] W.W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989. URL: http://link.aip.org/link/?SIR/31/221/1.

[9] E. Hansler and G. Schmidt. *Topics in Acoustic Echo and Noise Control: Selected Methods for the Cancellation of Acoustical Echoes, the Reduction of Background Noise, and Speech Processing*. Springer-Verlag, 2006. URL: http://www.amazon.com/Topics-Acoustic-Echo-Noise-Control/dp/354033212X/ref=pd_bxgy_b_text_b.

[10] Ralph E. Johnson and Brian Foote. Designing reuseable classes. *Journal of Object-Oriented Programming*, 1(2):22–35, June/July 1988.

[11] M. Kahrs, G.W. Elko, S.J. Elliot, S. Makino, J.M. Kates, M. Bosi, and J.O. Smith. The past, present and future of audio signal processing. *IEEE Signal Processing Magazine*, 14(5):30–57, Sep 1997. URL: .

[12] J.L. Kelly and B.F. Logan. Self-adaptive echo canceller. US Patent, Oct 1966. URL: http://www.freepatentsonline.com/3500000.html.

[13] Mathworks. Matlab classes and object-oriented programming. Matlab 2008 built in help, 2008.

[14] S. McGarrity. Introduction to object-oriented programming in matlab. *Matlab Digest*, Mar 2008. URL: http://www.mathworks.com/company/newsletters/digest/2008/mar/matlab_oop.html.

[15] J. Nagumo and A. Noda. A learning method for system identification. *Automatic Control, IEEE Transactions on*, 12(3):282–287, Jun 1967. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=24095&arnumber=1098599.

[16] K. Ozeki and T. Umeda. An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties. *Electronics and Communications in Japan*, 67-A(5):126–132, Feb 1984.

[17] V. Stewart, C.F.N. Cowan, and S. Sezer. Adaptive echo cancellation for packet-based networks, August 2004. URL: http://www.springerlink.com/content/2chlxje3pm3yk3c.

[18] E.J. Thomas. Adaptive echo canceller for non linear systems. US Patent Application, Mar 1970. URL: http://www.freepatentsonline.com/3647992.html.

[19] Bernard Widrow and Samuel D. Stearns. *Adaptive Signal Processing*, chapter ch4. Prentice Hall, 1985. URL: http://www.amazon.com/Adaptive-Signal-Processing-Prentice-Hall/dp/0130040290.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5775
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5775