Int. J. Engng Ed. Vol. 21, No. 4, pp. 572–579, 2005
Printed in Great Britain.

0949-149X/91 $3.00+0.00

A Simulink Laboratory Package for Teaching Adaptive Filtering Concepts*

LILI JIANG, KARL WIKLUND and SIMON HAYKIN
McMaster University, Adaptive Systems Laboratory, CRL-102, 1280 Main Street West Hamilton, Ontario, L8S 4K1, Canada. E-mail: jiang@soma.mcmaster.ca; haykin@mcmaster.ca

This paper describes a Simulink laboratory package for teaching adaptive filtering concepts. Each lab is designed to convey certain important features of a particular adaptive filter, and to provide comparisons with similar adaptive filtering algorithms. The filters covered include the LMS, nLMS, RLS, and GAL, as well as three members of the Kalman filter family and the particle filter. In addition to learning adaptive filtering concepts, these labs also help to ensure a greater familiarity with Simulink on the part of the student.

INTRODUCTION

IN RECENT YEARS, there has been a tremendous growth in the utilization of MatLab for the teaching of courses on digital signal processing, control systems and communications. The emphasis of MatLab on vector mathematics makes it especially well suited for this purpose, so its popularity is of no surprise. However, the utilization of Simulink for teaching DSP concepts has been much lower, despite its appealing visual nature and the ease of system construction that it offers.

At the same time, adaptive filtering, while already a subject of great importance, is becoming even more prominent both as a result of the development of cheaper computing power as well as more powerful algorithms. Accordingly, a package of visually oriented lab exercises has been created to illustrate specific concepts and algorithms relating to adaptive signal processing.

The lab programs themselves are presented as Simulink applications, in order to allow easy access to the system parameters by students. They can then manipulate these parameters in a variety of ways and observe the results. The accompanying exercises are meant to emphasize specific features of the algorithm under study. The package is of most value to the student when complicated algorithms are being examined, such as the particle filter. In these cases, the behaviour of the algorithm can be examined without concern for debugging problems or programming details. A good, intuitive understanding of the material can provide a foundation both for practical work, and ultimately for a more abstract understanding as well.

A further useful feature of the approach outlined here is its modularity. The blocks provided in this package are reusable in other contexts beyond the specific examples provided by us. In other words, the Simulink models may be used in other assignments set by the teacher or for projects of particular interest to the student.

To increase the interactivity of the assignments, we also take advantage of Simulink's capability to add user-defined system functions to perform specific tasks. Students can build their own functional blocks in order to solve some of the problems, thus also developing some practical programming skills. Assignments of this nature emphasize the use of MatLab system functions (S-functions), and may be written in either MatLab or C-MEX files.

PACKAGE CONTENTS

The Simulink examples that are presented with the package run through a variety of adaptive filtering algorithms. These include not only the well-known LMS and RLS algorithms, but also order-recursive filters like gradient adaptive lattice (GAL) filter. In addition, we include progressively more advanced algorithms such as the unscented Kalman filter and the particle filter (PF), both of which are of relatively recent origin. Each filter demonstration comes with a number of exercises designed to illustrate key aspects of the filter's behaviour. By varying the model parameters, students will be able to gauge system performance under a variety of conditions.

An additional benefit to using Simulink is being able to scale up simulations to the system-level design. It allows the user to not only study the filter by itself, but also as a component in a larger system. For example, the role of adaptive filters in communication systems is well known. Thus, a straightforward application of the principles discussed so far would be to integrate these algorithms into such a system subject to both stationary

* Accepted 9 February 2005.

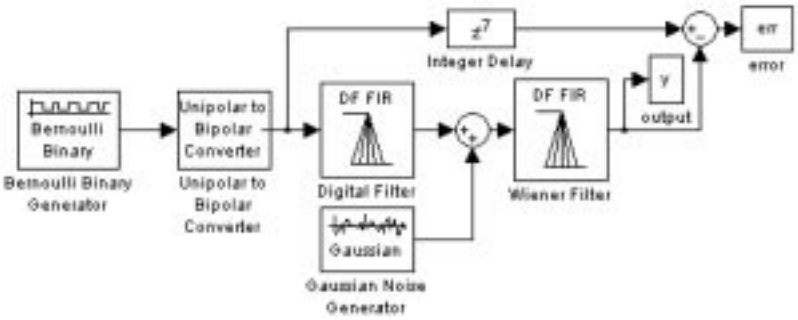

Fig. 1. Wiener filter for channel equalizer.

and non-stationary channels. Further exercises/applications will draw from additional problems in communications, signal processing, and control systems.

INTRODUCTORY LABS

These labs are meant to highlight some of the basic characteristics of adaptive filtering algorithms. Specifically, they will examine Wiener filtering, LMS and normalized LMS (nLMS) as well as the RLS and GAL (Gradient Adaptive Lattice) filters [1]. To ensure that the differences between these algorithms are well exemplified, two different real-world applications are used throughout these labs. These examples involve digital equalization as well differential pulse code modulation with backward adaptive prediction (DPCM-APB), which is used in some speech coding applications [2]. The reason for these different algorithms is two fold: both to highlight the differences in adaptation performance, and to demonstrate the multiple layers of signal processing that can go into a real application. In the case of these two examples, it is easy to see that the raw speech may be encoded using one algorithm, while the transmitter-receiver pair is based on an entirely different algorithm. A worthwhile student exercise may then be to build the entire system (speech coding and equalization) from the blocks provided, and to compare the outputs on a qualitative level.

Digital equalization

The digital equalization problem is presented for several of the algorithms considered in this package [1]. For each case the underlying problem is the same, which allows for easy comparisons of the different filters. Real, binary data is passed through a channel with an impulse response of:

$$h(k) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(\frac{2\pi}{W}(k-2)\right)\right), & k = 1,2,3 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

in which the parameter W controls the amount of distortion produced by the channel. The distortion increases with W, as does the condition number of the autocorrelation matrix of the received signal. Thus this parameter exercises some control over the quality of the channel output.

The optimal solution of this channel equalizer is the well-known Wiener solution, which is:

$$\boldsymbol{w}_o = \boldsymbol{R}_{xx}^{-1}\boldsymbol{p}_{dx} \tag{2}$$

where $\boldsymbol{R}_{xx}$ is autocorrelation matrix of the received signal, and $\boldsymbol{p}_{dx}$ is cross correlation vector of the desired output and the received signal. If the desired signal has a unit power, then the minimum square error (MSE) is:

$$J_{\min} = 1 - \boldsymbol{p}_{dx}^{H}\boldsymbol{w}_o \tag{3}$$

The Wiener filter solution for the equalizer gives the students the theoretical performance, which can be readily compared with the MSE performances of each of the filters in a variety of noise and channel conditions.

Through Simulink's graphical user interface (GUI), the system can be built by click-and-drag mouse operations similar to what we usually draw with pencil and paper as shown in Figs 1 and 2.

As mentioned before, the first LMS filter exercise also demonstrates the binary communications

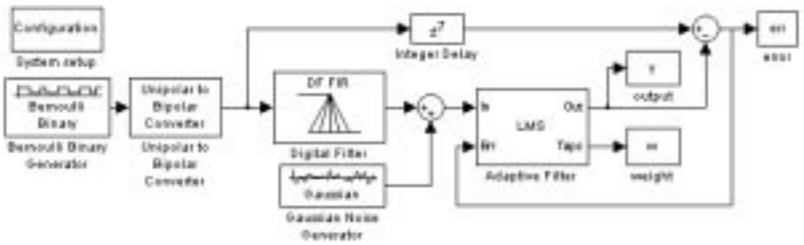

Fig. 2. LMS adaptive filter for channel equalizer.

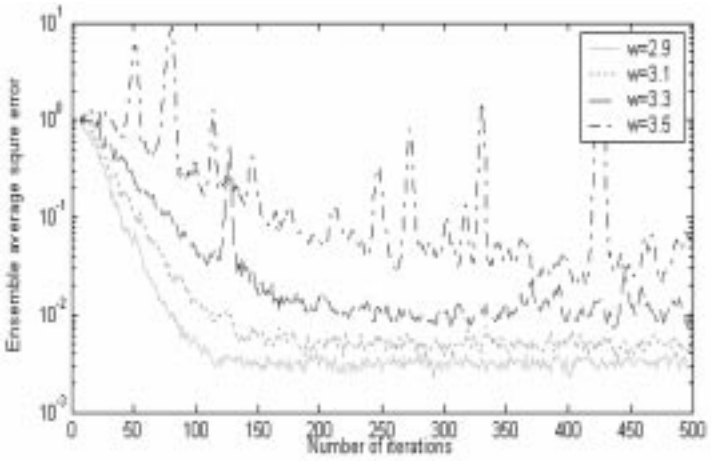

Fig. 3. LMS learning curves for $\mu = 0.075$ averaged over 200 trials. As can be seen here, the convergence depends on the value of W.

channel (Fig. 2). Users can easily access the significant parameters and most importantly make use of comparative studies between a number of other filters using this same problem.

In this and related experiments, students can achieve a good intuitive understanding of the performance of the various algorithms by changing the channel distortion levels and the learning rate. This introduces the students to the differences and potential pitfalls of each approach.

The equalization exercises also allow for the running of multiple, independent trials so as to compare overall system performance through learning curves. Sample results are shown in the following graphs, which were produced by varying the filter learning rate μ and the channel parameter W. In Fig. 3 for example, the LMS filter is used, and the learning rate is held constant at $\mu = 0.075$ while W is changed; the results are averaged over 200 trials. The plot in Fig. 4 shows the opposite scenario, where μ changes and W does not.

As mentioned before, this group of labs also includes other filters as well, which operate on the same problem described here. Figures 5 and 6 show some typical results that students should be able to achieve for the RLS and GAL filters.

Speech coding

In some speech communications applications, where the channel may involve several links, DPCM coding using backward adaptive filtering is utilized [2]. This is so because of the need to reduce buffering times, and to eliminate the need to transmit side information such as the predictor coefficients. It is well known that the LMS filter is what is typically used for this application, however any other prediction filter may be used as well. This results in an attractive lab problem, since the DPCM-APB algorithm is quite simple to implement and also provides a straightforward example of a non-stationary problem.

As in the previous labs, which dealt with digital equalization, students will be expected to compare the performance of several different algorithms. In this case, the quality of the output is ultimately determined by its variance. A more accurate predictor produces smaller errors, which means increased compression. Of course, students

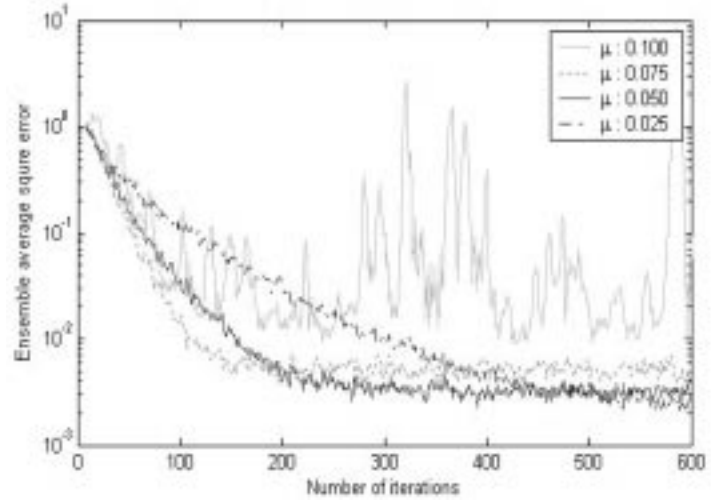

Fig. 4. LMS learning curves for $W = 3.1$ averaged over 200 trials. In this case, it can be seen in addition to determining convergence speed, μ also determines whether convergence occurs at all.

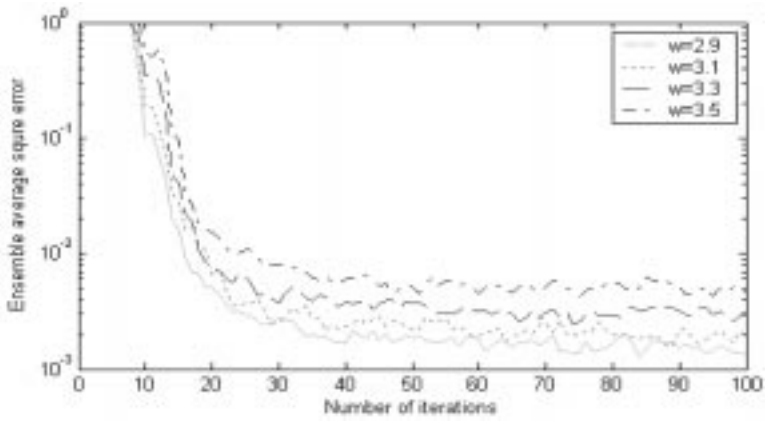

Fig. 5. RLS learning curves given that the initial error covariance estimate is $\sigma^2 = 0.001$. Notice that the learning curves are not as dependent on W here as they are for the LMS filter.

should still compare output against the input according to both qualitative and quantitative criteria.

ADVANCED LABS

Beyond the algorithms that have been discussed so far are others that are less well suited to the applications showcased in the introductory labs. These include the Kalman filter, the Extended Kalman filter, as well as the Particle Filter, all of which are used in a different fashion than either the LMS, RLS or GAL filters. As a result, a set of additional Simulink labs has been provided to demonstrate both the application of these filters, as well as to show some of the significant differences in their behaviour.

The Kalman filter

Kalman filters are a significant topic in themselves, and an in-depth study of related algorithms and applications goes well beyond the scope of an introductory course on adaptive filtering. However, it is nonetheless an essential topic in such a syllabus. To this end, three different members of the Kalman filter family are examined in the labs provided with this package. These are the Kalman filter itself, the extended Kalman filter, and the unscented Kalman filter.

Using the relatively simple mass-spring system, the performance differences between the three filters are readily highlighted. In this case, both a linear and a non-linear system are considered. In the former example, the state and observation equations are given by [3]:

$$\begin{bmatrix} \dfrac{dx_1(t)}{dt} \\ \dfrac{dx_2(t)}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w(t) + \begin{bmatrix} 0 \\ F \end{bmatrix} \tag{4}$$

$$y(t) = [1 \quad 0] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{5}$$

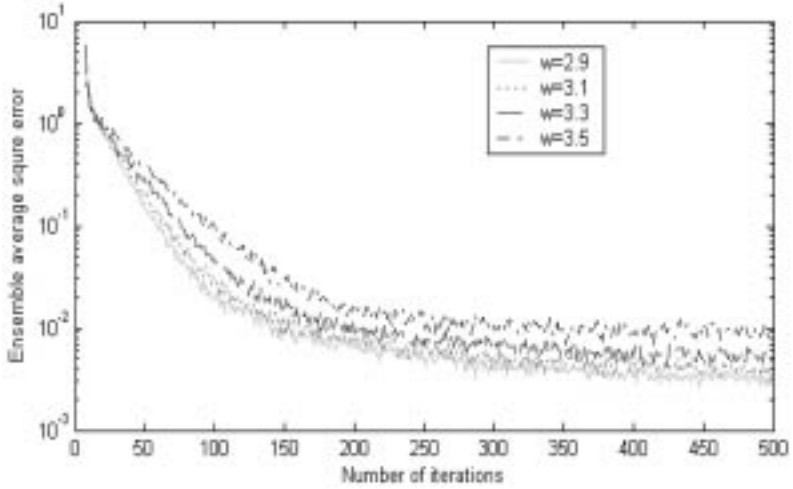

Fig. 6. GAL learning curves for $\mu = 0.5$. It is apparent that the learning curves are not strongly affected by the parameter W.

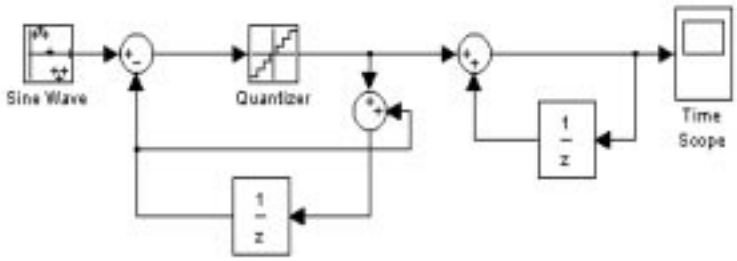

Fig. 7. The non-adaptive DPCM coder and decoder using a sine-wave source. The unit delay in the encoder can be replaced by an adaptive filter in order to test other algorithms.

where $x_1(t)$ measures the displacement, $x_2(t)$ is the velocity, ζ is the damping ratio, and ω is the natural frequency of the system. The term $w(t)$ is used to indicate the process noise, and the final term F indicates the driving force (which in this case is produced by the mass). Students can manipulate these parameters, and the Kalman filter parameters as well in order to determine how the algorithm performs under a variety of conditions, including model mismatches.

The non-linear system model is similar, but assumes no knowledge of the damping ratio ζ, so it is now included as a state variable. This produces the non-linear state equation shown in Equation (6) and the accompanying observation equation given in (7):

$$\begin{bmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \\ \frac{dx_3(t)}{dt} \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\omega^2 x_1(t) - 2x_2(t)x_3(t)\omega \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} w(t) + \begin{bmatrix} 0 \\ F \\ 0 \end{bmatrix} \tag{6}$$

$$y(t) = [1 \quad 0 \quad 0] \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \tag{7}$$

In this case $x_3(t)$ is the state variable representing the damping ratio ζ. The examples for both the UKF and EKF in this package make use of this model, allowing for valuable comparisons. In particular, students will be able to discover that while the UKF performs better if accurate knowledge about the system is available, it is less robust than the EKF in the presence of noise model mismatches.

It is worth noting that all of these Kalman filter algorithms are implemented as MatLab S-functions. As a result, it is very easy for students to modify the code in order to produce and compare other Kalman-based algorithms (e.g. information filtering, square-root filtering, etc). No model modifications need be made in this eventuality.

The particle Filter

The particle filter is related to the Kalman filter in that both are Bayesian filters. Like the Kalman filter, the particle filter is also suitable for state-space modeling, but is particularly useful for nonlinear or non-Gaussian problems [4]. In contrast to linear Gaussian models there is no explicit optimum solution. Instead the problem is solved through the iterative construction of the state variable probability density functions through Monte-Carlo sampling and re-sampling.

In the example provided in our package, we developed a particle filtering model for a wireless channel tracking problem. Despite the difficulty of the actual problem, the state-space model is relatively simple and is given by:

$$\begin{cases} x_t = \beta x_{t-1} + w_{t-1} \\ y_t = s_t x_t + v_t \end{cases} \tag{8}$$

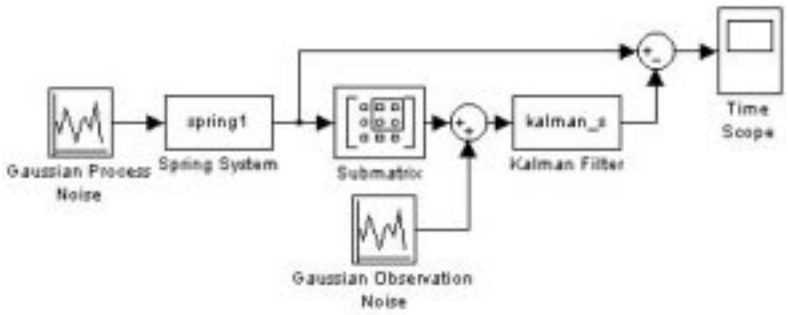

Fig. 8. The Linear Kalman Filter with the mass-spring system. Both the spring system and Kalman filter are implemented as MatLab S-functions.

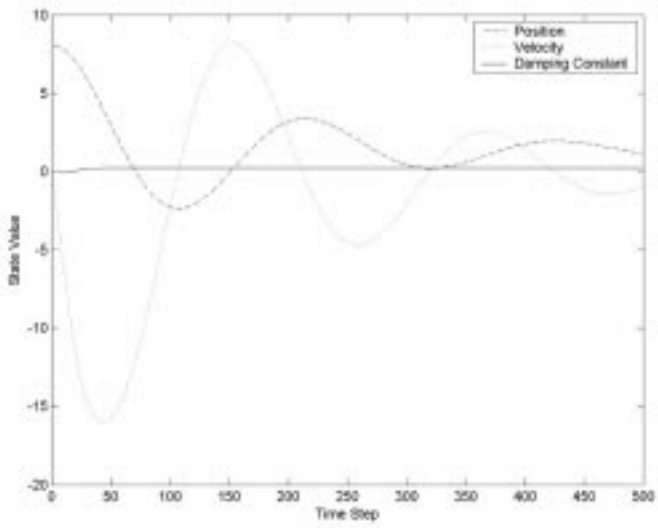

Fig. 9. State values versus time for the UKF filter. For even small mismatches in the model process noise, the damping constant state parameter may not converge.

Where x_t is the channel state, and s_t is the transmitted signal. The term w_t represents the process noise, which is sub-Gaussian in nature. The final term, v_t represents the observation noise, which is generated using a Middleton Class A model. The factor β is a constant, and from real-life problems of this type, has been experimentally measured as being approximately 0.99 [5].

In this lab (see Fig. 10) the modulated data is transmitted across a non-Gaussian, time-varying channel, and the particle filter is applied in order to perform semi-blind tracking of the communications channel. Tracking and performance can be monitored by the students, as can the effects of varying the system parameters (e.g. the number of particles, etc). Unlike the other labs, this assignment is not comparison-based, but is instead a demonstration of the capabilities and behaviour of the algorithm in question.

IMPLEMENTATION

Many of the labs described in this paper make use of algorithms that are already available in MatLab. This is a sensible approach, since in those particular cases there is no need to modify the existing material. It is also hoped that students will become familiar enough with the Simulink blocksets to use them effectively. However, there are number of labs that do not use pre-existing Simulink blocks. These include the GAL experiments, as well as those relating to the Kalman filter and its variations and finally, the particle filter. In these cases, no adequate Simulink block was available for the specific needs of the experiment. As a result, the algorithms in question were implemented as customized S-functions in either MatLab or C.

Since the use of S-functions is itself a topic that students should become familiar with, most of the aforementioned algorithms were implemented in MatLab code rather than C. The MatLab programs are much easier to read and modify, and so are themselves suitable for assignments requiring the implementation of some variation of the original filter. This is of particular interest with regards to the Kalman filter family, since there are several variations that may be of interest to the student or instructor (e.g. square-root filter, information filter, etc.).

The use of MatLab code is not always practical of course, and so some C-code is used to implement certain labs. This is the case for the particle filter, which is implemented entirely in C owing to the relative slowness of MatLab implementations. The only other filter implemented in C is the GAL filter, which again runs much faster in C than in MatLab. However, in the case of the GAL, a MatLab version of this algorithm is also provided for purposes of demonstration. A comparison of the running times of the two approaches is instructive.

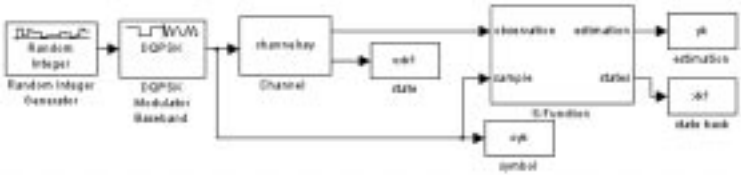

Fig. 10. The particle filter state tracking system for wireless communications.

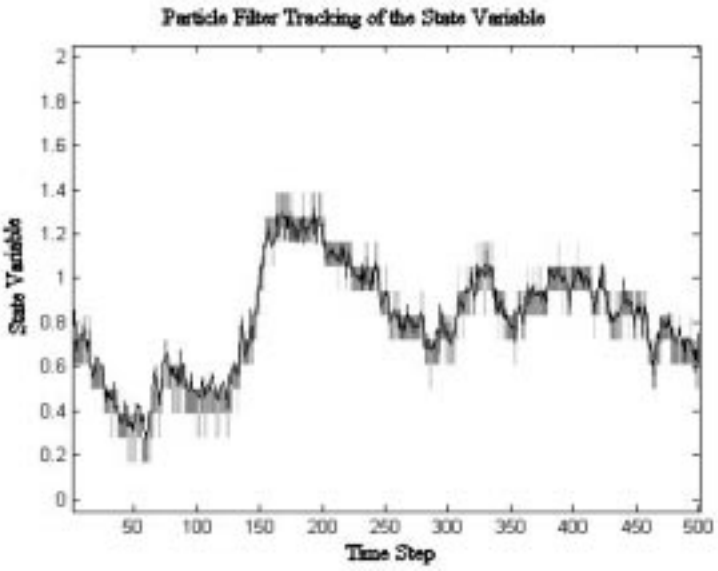

Fig. 11. State tracking using a particle filter. The particles are distributed around the state variable. The solid line shows the actual channel state and the gray squares show the distribution of the particles.

CONCLUSIONS

The goal of this project has been to demonstrate some of the important concepts that are central to a foundational course on adaptive signal processing. At the same time, we desired that these concepts be discussed in a visually concrete manner that demonstrated both the applications themselves as well as their structure. To this end, Simulink's block diagram structure and modularity were found to be very useful, producing easy to understand yet functional system models. In addition, the labs that have been provided with this package can be readily modified for the purposes of further explorations of the material.

Beyond the teaching of adaptive filtering, we feel that these labs have the added benefit of helping students learn a software package that they are likely to use outside of a university setting. Simulink is often used in industry for a wide variety of applications, including those relating to telecommunications and signal processing, so it is important that students familiarize themselves with it in addition to their other studies. As a result, we have chosen to implement labs that, in addition to teaching the DSP-related material, also highlight some of the useful features of Simulink as well.

REFERENCES

1. S. Haykin, *Adaptive Filter Theory*, 4th Ed., Upper Saddle River, NJ: Prentice-Hall (2002).
2. K. Sayood, *Introduction to Data Compression*, 2nd Ed., San Francisco, CA: Morgan-Kaufman (2000).
3. A. K. M. Azad, M. O. Tokhi, A. Pathania and M. H. Shaheed, A MatLab/Simulink based environment for intelligent modelling and simulation of flexible manipulator systems, *Proc. 2004 American Society for Engineering Education Conf. & Exposition*, 20–23 June, 2004, Utah, USA.
4. S. Arulampalam, S. R. Maskell, N. J.Gordon and T. Clapp, A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking, *IEEE Trans. Signal Processing*, **50**, 2002, pp. 174–188.
5. S. Haykin, K. Huber and Z. Chen, Bayesian sequential state estimation for wireless communication, *IEEE Trans. Wireless Communication*, **140**, 2004, pp. 107–113.

Lili Jiang received her BA and MA degrees in electrical and electronic engineering from Harbin Institute of Technology, Heilongjiang, PR China. She has worked on communication systems in industry for 9 years, and has also been a lecturer on communication and control engineering at Northern Jiaotong University in Beijing for 3 years. Her current research interests include communication systems and cognitive radio.

Karl Wiklund has received the B.Eng.Scty. and M.A.Sc. degrees from McMaster University, and is currently a Ph.D. student at that institution. He has also been employed as a Defence Research Assistant at Defence Research Establishment Atlantic (DREA) in

Dartmouth, Nova Scotia where he worked on problems relating to sonar signal processing. Currently he is working on developing acoustic simulation software for the automatic testing of hearing aid algorithms, although his other interests include signal processing applications relating sonar, geophysics, and neural networks.

Simon Haykin received his B.Sc. (First-Class Honours) in 1953, Ph.D. in 1956, and D.Sc. in 1967, all in Electrical Engineering from the University of Birmingham, England. In 1980, he was elected Fellow of the Royal Society of Canada. He was awarded the McNaughton Gold Medal, IEEE (Region 7), in 1986. He is currently a University Professor in the Department of Electrical and Computer Engineering, McMaster University. He is a fellow of the IEEE, fellow of the Royal Society of Canada, and recipient of the honourary degree of Doctor of Technical Sciences from ETH, Zurich, Switzerland. Dr. Haykin is the author/editor of over 500 technical papers and over 40 books, including the popular undergraduate and graduate textbooks Adaptive Filter Theory (Prentice-Hall, 4th Ed., 2002), Neural Networks: A Comprehensive Foundation (Prentice-Hall, 2nd Ed., 1999), Communication Systems (Wiley, 4th ed., 2001) and Signal and Systems co-authored with Barry VanVeen (Wiley, 2nd Ed., 2003). He is also the first recipient of the Henry Booker Medal, which was awarded by URSI in 2002.