Adaptive Filter Receiver

The term, adaptive filter, means that the characteristics of a filter are changing in an automated fashion to obtain the best possible signal quality. In this example, we show how MatDeck can be used to adaptively receive the noisy signal, u(n) ,through the TCP/IP channel. The TCP/IP channel communication mimics the data acquisition from a specific piece of hardware. The second, TCP/IP, channel is used to transmit the desired signal, d(n) , which is used to adaptively remove the noise direct from the noisy signal.

In an adaptive Wiener filter the error signal is fed back to the filter weights to adjust them using the steepest-descent algorithm. In practical implementation estimating the gradient from the available data using the least-mean-square (LMS) algorithm is necessary. The parameters of the LMS algorithm are: filter length, step size and the receiver buffer size.

MatDeck simulation

A simple MatDeck simulation is constructed by using two MatDeck documents. Document, LMSsender.mdd generates a single sinusoid at a normalized frequency of f_0=0.2 to which the additive white Gaussian noise is added. The relation between the amplitude of the sinusoid and the standard deviation of the noise is defined by the Signal and Noise Ratio (SNR). LMSsender.mdd creates two TCP/IP channels, the first for sending the noisy signal, and the second for sending the pure sinusoid as the desired signal. The simulation is run using 1000 samples per second with SNR equal to 10dB.

In this document, adaptive_receiver.mdd, we connect to two channels created by LMSsender.mdd. Parameters for LMS algorithm are defined as: step size at mu=0.001, filter length is 11, and the buffer size is 1000 to collect exactly one second of the signal. Further, MatDeck's function ,lmsreceive() , is also used. The function, lmsreceive(), has two additional arguments in channel objects which are used to communicate with the sender.

All necessary variables are given in the next canvas. The code of the function: lmsreceive() are written in MatDeck script language and is shown in the segment below.

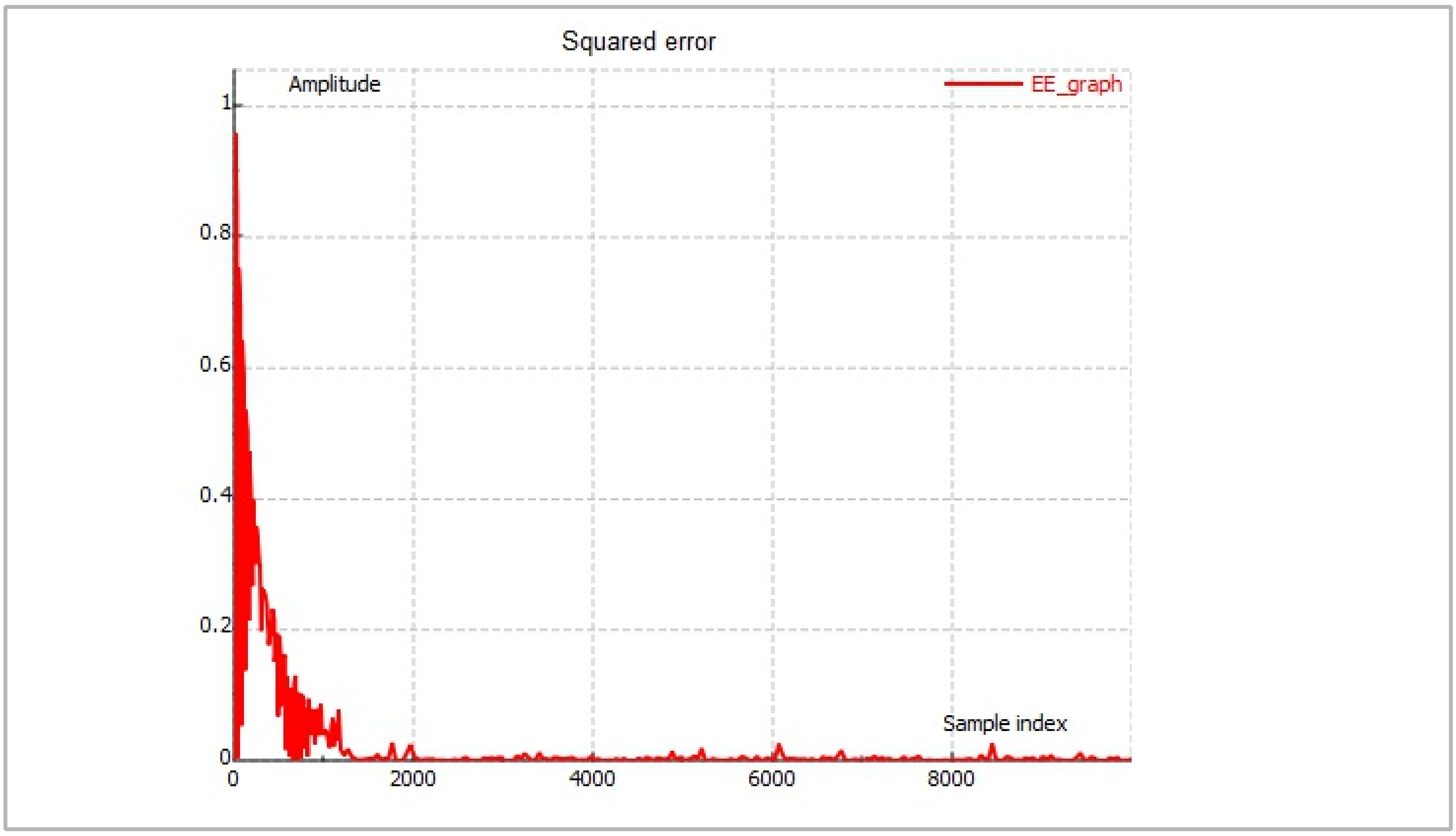
```
channels table( ) = undefined



g1 := channel connect("127.0.0.1" , 1805)    TCP/IP channel used to send noisy signal u(n)
g2 := channel connect("127.0.01" , 1806)     TCP/IP channel used to send desired signal d(n)
mu := 0.001     Step size for LMS algorithm
length := 11    Wiener filter length
  Required variables
```

$Y\_out := \begin{bmatrix} 0 \end{bmatrix}$ $ee\_out := \begin{bmatrix} 0 \end{bmatrix}$ $time := \begin{bmatrix} 0 \end{bmatrix}$ $Y\_graph := \begin{bmatrix} 0 & 0 \end{bmatrix}$ $EE\_graph := \begin{bmatrix} 0 & 0 \end{bmatrix}$

```
lmsreceive(mu , length , 1000 , g1 , g2)    LMS function with data acquisition
lmsreceive(mu_a , length_a , buffer_size , ch1 , ch2)
{
    buffer:=matrix create(length_a , 1 , 0)
    ww:=matrix create(length_a , 1 , 0)
    y_out:=matrix create(buffer_size , 1 , 0)
    ee:=matrix create(buffer_size , 1 , 0)
    ct:=0
    Temp:=(xnodes([0  buffer_size] , buffer_size-2))^T
    while(true)
    {
        d_in:=channel read(ch1 , true)
        u_in:=channel read(ch2 , false)
        for(i:=0 , i<buffer_size , i+=1 )
        {
            for(j:=length_a-1 , j>0 , j-=1 )
            {
                buffer[j] = buffer[j-1]
            }
            buffer[0] = u_in[i]
            y_out[i] = mat transpose(ww) · buffer
            ee[i] = d_in[i] - y_out[i]
            Temp1:=mu_a · ee[i] · buffer
            ww+=Temp1
        }
        Y_out=join mat rows(Y_out , y_out)
        ee_out=join mat rows(ee_out , ee)
        time=join mat rows(time , ct · buffer_size+Temp )
        ct+=1
        Y_graph=join mat cols(time , Y_out)
        EE_graph=join mat cols(time , ee_out^2)
    }
}
```

Results

MatDeck's function, lmsreceive(), takes a noisy signal, the desired signal from the TCP/IP channels, together with Wiener filter length and step size for LMS algorithm as an input. The function then returns the filtered signal, adaptive filter coefficients, and the error signal. In the next segment, we give the graph of the squared error. From the graph of the squared error, we can see that the adaptive filter minimizes the error in a steepest descent sense as expected.


Squared error
Amplitude
EE_graph
Sample index