11 asptnlms

Purpose

Sample per sample filtering and coefficient update using the Normalized LMS (NLMS) algorithm.

Syntax

```
[w,y,e,p]= asptnlms(x,w,d,mu)
[w,y,e,p]= asptnlms(x,w,d,mu,p,b)
```

Description

`asptnlms()` implements the NLMS adaptive algorithm used to update transversal adaptive filters. Referring to the general adaptive filter shown in Fig. 2.6, `asptnlms()` takes an input samples delay line $x(n)$, a desired sample $d(n)$, the vector of the adaptive filter coefficients from previous iteration $w(n-1)$, the step size mu, and returns the filter output $y(n)$, the error sample $e(n)$ and the updated vector of filter coefficients $w(n)$. The NLMS also estimates the instantaneous power $p(n)$ of the input signal and normalizes the step size mu by this estimate to make the update algorithm independent of the input signal energy. If the input parameters p and b are given, an efficient recursive estimation of $p(n)$ is used, otherwise the inner product of $x(n)$ with itself is used instead. The update equation of `asptnlms()` is given by

$$\underline{\mathbf{w}}(n) = \underline{\mathbf{w}}(n) + (\frac{\mu}{p})e(n)\underline{\mathbf{x}}(n). \qquad (35)$$

The input and output parameters of `asptnlms()` for an FIR adaptive filter of L coefficients are summarized below.

```
Input Parameters [Size]::
   x   : input samples delay line [L x 1]
   w   : filter coefficients vector w(n-1) [L x 1]
   d   : desired output d(n) [1 x 1]
   mu  : adaptation constant
   p   : last estimated power of x(n), p(n-1)
   b   : AR pole for recursive calculation of p(n)

Output parameters::
   w   : updated filter coefficients w(n)
   y   : filter output y(n)
   e   : error signal; e(n) = d(n)-y(n)
   p   : new estimated power of x(n), p(n)
```

Example

```
% NLMS used in a simple system identification application.
% By the end of this script the adaptive filter w should
% have the same coefficients as the unknown filter h.
%
```

```
iter = 5000;                  % Number of samples to process
% Complex unknown impulse response
h    = [.9 + i*.4; 0.7+ i*.2; .5; .3+i*.1; .1];
xn   = 2*(rand(iter,1)-0.5);  % Input signal, zero mean random.
% although xn is real, dn will be complex since h is complex
dn   = osfilter(h,xn);        % Unknown filter output
en   = zeros(iter,1);         % vector to collect the error
% Initialize the NLMS algorithm with a filter of 10 coef.
[w,x,d,y,e,p]=init_nlms(10);

%% Processing Loop
for (m=1:iter)
   x = [xn(m); x(1:end-1)];  % update the input delay line
   d = dn(m) + 1e-3*rand;    % additive noise of var = 1e-6
   % call NLMS to calculate the filter output, estimation error
   % and update the coefficients.
   [w,y,e,p]= asptnlms(x,w,d,0.05,p,0.98);
   % save the last error sample to plot later
   en(m) = e;
end;

% display the results
subplot(2,2,1);stem([real(w) imag(conj(w))]); grid;
subplot(2,2,2);
eb = filter(.1,[1 -.9], en .* conj(en));
plot(10*log10(eb  ));grid
```

Running the above script will produce the graph shown in Fig. 4.12. The left side graph of the figure shows the adaptive filter coefficients after convergence which are almost identical to the unknown filter h. The right side graph shows the mean square error in dB versus time during the adaptation process, which is usually called the learning curve. The lower limit of the error power is governed here by the additive noise at the output (-60 dB).

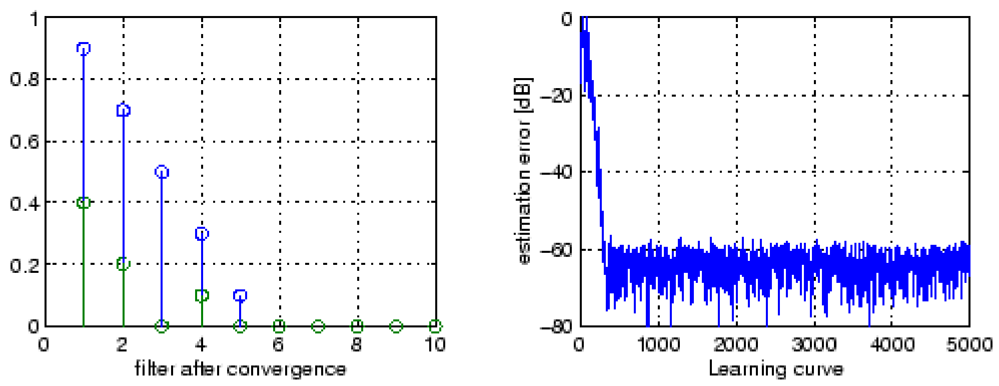

Figure 4.12: The adaptive filter coefficients after convergence and the learning curve for the complex FIR system identification problem using the NLMS algorithm.

Algorithm

The NLMS algorithm is a slightly improved version of the LMS algorithm. The current implementation of `asptnlms()` performs the following operations

- Filter the input signal $x(n)$ through the adaptive filter $w(n-1)$ to produce the filter output $y(n)$.
- Calculates the error sample $e(n) = d(n) - y(n)$.
- Estimates the input signal power p and normalizes the step size mu by this estimate (the improvement upon the LMS).
- Updates the adaptive filter coefficients using the error $e(n)$ and the delay line of input samples $x(n)$ resulting in $w(n)$.

Remarks

Like the LMS, the NLMS is also a stochastic implementation of the steepest-descent algorithm where the *mean* value of the filter coefficients converge towards their optimal solution. Therefore, the filter coefficients will fluctuate about their optimum values given by the Wiener solution. The amplitude of the fluctuations is controlled by the step size. The smaller the step size, the smaller the fluctuations (less final misadjustment) but also the slower the adaptive coefficients converge to their optimal values. Note also the following.

- The NLMS algorithm estimates the energy of the input signal each sample and normalizes (divides) the step size by this estimate, therefore selecting a step size inversely proportion to the instantaneous input signal power. Although this improves the convergence properties in comparison to the LMS, it does not solve the eigenvalue spread problem.
- The NLMS algorithm shows stable convergence behavior only when the step size mu (convergence constant) takes a value between zero and an upper limit defined by the statistics of the filter's input signal. The fastest convergence will be achieved for a white noise input sequence. Such white input signal has all its eigenvalues are equal to the noise variance σ^2 and therefore has a diagonal autocorrelation matrix with diagonal values equal to σ^2.
- The more colored the spectrum of the input signal, the slower the convergence will be. This is due to the large eigenvalue spread for such colored signals. This makes the convergence composed of several modes, each associated with one of the eigenvalues.
- `asptnlms()` supports both real and complex data and filters. The adaptive filter for the complex NLMS algorithm converges to the complex conjugate of the optimum solution.
- `asptnlms()` does not update the input delay line for $x(n)$, this has been chosen to provide more flexibility, so that the same function can be used with transversal as well as linear combiner structures. Delay line update, by inserting the newest sample at the beginning of the buffer and shifting the rest of the samples to the right, has to be done before calling `asptnlms()` as in the example above.

Resources

The resources required to implement the NLMS algorithm for a transversal adaptive FIR filter of L coefficients in real time is given in the table below. The computations given are those required to process one sample and assumes that recursive estimation of the input power is used.

MEMORY	$2L+7$
MULTIPLY	$2L+4$
ADD	$2L+2$
DIVIDE	1

See Also

INIT_ NLMS, ECHO_ NLMS, EQUALIZER_ NLMS, ASPTLMS, ASPTVSSLMS, ASPTLCLMS.

Reference

[11] and [4] for extensive analysis of the NLMS and the steepest-descent search method.

Next Up Previous Contents

Next: 12 asptpbfdaf **Up:** 4 Transversal and Linear **Previous:** 10 asptmvsslms **Contents**