Implementation of Median-LMS Adaptive Filters Using Power-of-Two Multiplication in the Adaptation Arithmetic

Dr Malcolm D Macleod
Department of Engineering, University of Cambridge, Trumpington Street, CAMBRIDGE, UK, CB2 1PZ
email mdm@eng.cam.ac.uk

ABSTRACT
There are many existing methods for reducing the VLSI implementation cost of the adaptation arithmetic in stochastic gradient adaptive filters. Some of them cause significant performance impairments (such as slower convergence or increased misadjustment noise). However the replacement of multiplications by Power-of-Two Quantisers (PTQ), together with shifters, has been shown to reduce cost with very little performance impairment. This paper reviews an older method, Power-of-Two Multiplication (PTM), which is equivalent to exponent-only floating-point multiplication. We show that although it has little or no advantage over the PTQ approach for the LMS, NLMS, and GAL algorithms, the PTM method has significant advantages for implementing the median-LMS adaptive filter algorithm.

1. INTRODUCTION

Consider the well known LMS algorithm [1], with input $x(k)$, output $y(k)$, and desired (reference) signal $d(k)$:

$$y(k) = \mathbf{w}^T(k)\mathbf{x}(k) \tag{1}$$

$$e(k) = d(k) - y(k) \tag{2}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k) \tag{3}$$

where $\mathbf{x}(k) = [\ x(k)\ x(k\text{-}1)\ \ldots\ x(k\text{-}M\text{+}1)\ ]^T$; M is the filter length; $\mathbf{w}(k) = [\ w(0)\ w(1)\ \ldots\ w(M\text{+}1)\ ]^T$ is the filter coefficient vector; and μ is the step size. In the NLMS algorithm [1,6], (3) is replaced by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\tilde{\mu}}{\|\mathbf{x}(k)\|^2 + a} e(k)\mathbf{x}(k) \tag{4}$$

in which the step size is normalised by the denominator term, a stochastic estimate of the input signal power (the addition of a prevents division by zero when the input signal is zero). There are also several Variable Stepsize (VS-)LMS algorithms, which may be viewed as approximations of NLMS.

In all these algorithms, the term $e(k)\mathbf{x}(k)$ is a stochastic approximation of the gradient vector, and its computation requires M multiplications. It is well known that adaptation can still take place if this calculation is approximated [1], and there are many ways in which it can be simplified. In the Signed Error (SE) algorithm, $e(k)$ is replaced by sign($e(k)$); in the Signed Regressor (SR) algorithm, $x(k)$ is replaced by sign($x(k)$); and in the Sign-Sign (SS) algorithm, both $e(k)$ and $x(k)$ are replaced by their signs. The SS algorithm gives great savings, but seriously impairs convergence; the SE algorithm (always as good as or better than the SR) has a smaller, but significant, effect on convergence.

Another approach is to code $e(k)$ using a Power-of-Two Quantiser (PTQ) [2-5]. The B-bit PTQ described in [2,5] is defined by

$$q(u) = \begin{cases} \text{sign}(u) & |u| \geq 1 \quad (5) \\ 2^{\lfloor \log 2(|u|) \rfloor}\,\text{sign}(u), & 2^{-B+1} \leq |u| < 1 \quad (6) \\ 0, & |u| < 2^{-B+1} \quad (7\text{a}) \end{cases}$$

where $\lfloor x \rfloor$ is the integer $\leq x$. $q(u)$ is either 0 or it has a single "1" in one of B-1 positions. The deadband created by equation (7a) may lead to "stalling" (cessation of adaptation for small error values). In an alternative PTQ described in [8] the third line is therefore changed to

$$\{\ 2^{-B+1}\,\text{sign}(u), \quad |u| < 2^{-B+1} \tag{7b}$$

which avoids the deadband created by (7a), at the expense of an increase in misadjustment noise.

In [3] the PTQ quantiser is applied to the LMS algorithm to replace each multiplication $e(k)x(k\text{-}i)$ by sign($x(k\text{-}i)$) $\times$ $q(e(k))$; it is an approximation of the SR algorithm, and requires addition of $\pm\, q(e(k))$ to each coefficient, when computing (3). In [4,5] $e(k)x(k\text{-}i)$ is replaced by $x(k\text{-}i) \times q(e(k))$, requiring addition of a shifted version of the value $\pm\, x(k\text{-}i)$ to coefficient $w(i)$. This second approach has also been applied to the VS-LMS algorithm [5], the Gradient Adaptive Lattice (GAL) algorithm [4], and the NLMS algorithm [7].

In (4) the term $\|\mathbf{x}(k)\|^2$ is the sum of squares of input samples. Cheng and Evans [7] describe circuits for approximate squaring, the simplest of which requires no hardware at all - it simply maps the bits of the input value to different locations in the output word. This approach is compatible with PTQ in adaptive filter implementation [7].

In the present paper, we re-consider an earlier approach to the simplification of the adaptation arithmetic in which *both* inputs to a multiplication are coded using PTQ [6]. The multiplication is then replaced by addition of the PTQ code values, followed by "decoding" of the result. This approach, which we call Power-of-Two Multiplication (PTM), may be regarded as floating-point arithmetic with an exponent but no mantissa, or as a coarse form of logarithmic arithmetic. In [6] it was used in an NLMS echo canceller.

In section 2 we define PTM, and in section 3 we explain how it can be used in an (N)LMS adaptive filter. In section 4 we derive the implementation costs of PTM, and in section 5 we compare it with the cost of PTQ plus shifters. We show that for the (N)LMS and GAL algorithms PTM probably has no advantage over PTQ. However in section 6 we show how PTM may be applied to the median-LMS filter, and demonstrate that it offers significant cost savings in that application.

2. POWER-OF-TWO MULTIPLICATION

In practice, the use of PTQ in [4,5] is implemented by encoding the sign and the magnitude of the error using an encoder with a transfer function such as the following:

$$C(u) = \begin{cases} 0 & |u| \geq 1 \quad (8) \\ \lfloor \log_2(|u|) \rfloor, & 2^{-B+1} \leq |u| < 1 \quad (9) \\ -B \text{ or } -B+1, & |u| < 2^{-B+1} \quad (10) \end{cases}$$

$$S(u) = \operatorname{sign}(u)$$

In [4,5] each $x(k-i)$ is input to a negater controlled by $S(e(k))$, and the result is shifted under the control of $C(e(k))$. In PTM, multiplication of two variables is approximated by adding their code values and decoding the result. Similarly division is replaced by subtraction.

From (9) we see that code value $p = C(u)$ implies

$$2^{p} \leq |u| < 2^{p+1},$$

provided $-B < p < 0$. Thus if we replace multiplication by adding together two code values p and $q = C(v)$, then

$$2^{p+q} \leq |uv| < 2^{p+q+2}. \qquad (11)$$

Hence the most appropriate power of two to output as the decoded value of $p+q$ is 2^{p+q+1}, the harmonic mean of the two bounds in (11). By the same argument, if we replace division by subtraction of code values, the most appropriate power of two to output as the decoded value of $p-q$ is 2^{p-q}. If a combined multiplication and division are replaced by addition of codes $p+q$ and then subtraction of a third code r, then we can show that, on average, 2^{p+q-r} is equal to the correct result multiplied by 0.72. It is therefore best, by a small margin, to decode the code value as $2^{p+q-r+1}$.

The optimum strategy is therefore:

- when adding code values, add 1 to the result;
- decode the magnitude of code value s as $D(s) = 2^{s}$.

As with all finite-precision arithmetic systems, it is necessary to handle overflow. That is considered further in section 4.

The code values defined by (8-10) were chosen for convenience of explanation. Note that the actual code values may be altered if it makes implementation of the coder $C(u)$ and decoder $D(s)$ easier. Our implementation adds an offset of $+B$ to $C(u)$ as defined in (8-10), and a corresponding change is made to the decoder (including a binary shift to give correct overall scaling).

3. APPLICATION OF PTM TO (N)LMS

To apply PTM to the LMS algorithm (3), we could:

- encode $e(k)$: $E = C(e(k))$; $S = \operatorname{sign}(e(k))$;
- encode data values: $X_i = C(x(k-i))$; $S_i = \operatorname{sign}(x(k-i))$;
- add code values, decode, and update each coefficient:

$$w_i(k+1) = w_i(k) + S \times S_i \times D(\, E + 1 + m + X_i \,),$$

where $\mu = 2^m$. Figure 1 shows the implementation. Since the hardware cost of a coder having J-bit output is greater than the hardware cost of a J-bit parallel latch (see section 4) it is best to use a single encoder C_X of the input data [$X_0 = C(x(k))$; $S_0 = \operatorname{sign}(x(k))$] and then feed its output into a shift register from which [X_i; S_i] can be read out, as shown in Fig.1.

In Fig. 1, blank rectangles are stores, the input data wordlength is N bits, the code values (including sign) are J bits, and only the adaptation arithmetic for the first tap is shown; the other taps are identical.

To apply PTM to NLMS (4), we would instead:

- compute $g(k) = \|\mathbf{x}(k)\|^2$ using the approximate squarer described by Cheng and Evans [7];
- encode $g(k) + a$: $G = C(g(k) + a)$;
- update each coefficient as follows:

$$w_i(k+1) = w_i(k) + S \times S_i \times D(\, E - G + 1 + m + X_i \,),$$

where $\tilde{\mu} = 2^m$. Subtraction of G replaces division.

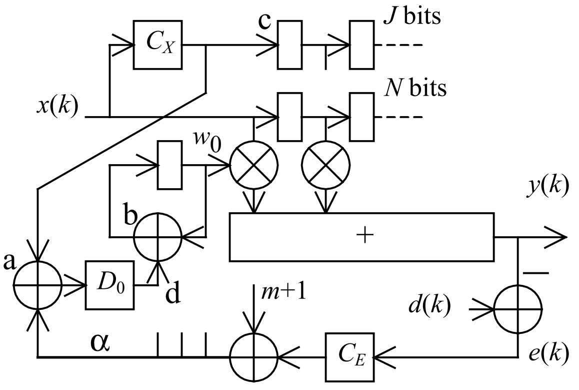

Figure 1 - LMS adaptive filter using PTM

4. LOGIC IMPLEMENTATION OF PTM

The encoder $C(u)$ (8-10) may be implemented by first negating the input u if it is negative, and then generating the index of the highest power of 2 in the resulting binary number. A further small approximation may be used to reduce implementation cost: instead of negation (2's complement) we implement inversion (1's complement). Assuming that the input is an N-bit 2's complement value, the (N-1)-bit output $\bar{u}$ can be obtained using N-1 XOR gates, all with the MSB as one input. Assuming that transmission-gates [9] are used, this requires 2+6(N-1) transistors (including a saving because the MSB is a common input to all the XORs). The remainder of the encoder can be implemented with a NOR-NOT ladder to produce a "thermometer code" (in which all bits from the LSB up to the highest "1" in $\bar{u}$ are "1") which uses 6N-10 transistors, followed by a NAND network, which requires 28 transistors for 8-bit input, 3-bit output, or 46 transistors for 12-bit input, 4-bit output, or 72 transistors for 16-bit input, 4-bit output. Hence the overall cost for an encoder C is 110 transistors for 8-bit input, 176 for 12 bits, or 250 for 16 bits (and more efficient designs may exist).

An ***N*-bit parallel latch** may be implemented using 18N transistors, and an ***N*-bit full-adder** (with no carry into the LSB stage) using 34N-20 transistors.

The decoder D starts with a 3 (or 4)-line to N-line decoder, which can be implemented using 50 transistors for 8-bit output, rising to 116 transistors for 16-bit output. To handle overflow, the decoder can easily be modified to take as an additional input the MSB carry-out of the adder **a**, and if that carry-out (overflow bit) is 1, to output a "1" in its most significant output bit.

This could be followed by an inverter (again as an approximation to a negater) at a cost of 2+6*N* transistors, feeding adder **b** in Fig. 1. However, it is possible to exploit the restricted form of the decoder output to reduce implementation cost. The combined circuit either adds or subtracts a "1" at a single bit position, at a cost of <28*N* transistors. One of the *N* bits of this circuit is shown in Fig. 2.

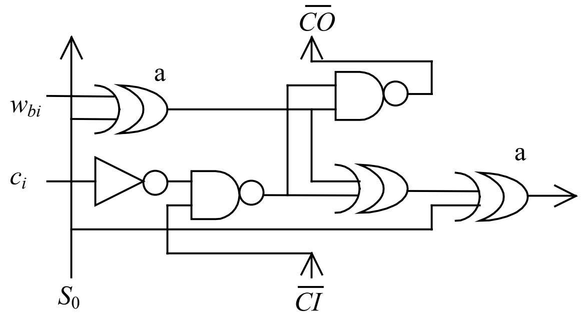

Fig. 2. One bit of the "single-1" adder/subtracter

In Fig. 2, S_0 is the negation control (= $S \times S_i$); two transistors are saved in each of the XORs marked **a** because they share a common input S_0 (common for all i). w_{bi} is the ith bit of the coefficient, and c_i is the ith bit of the coder output (only one at most of these bits is "1"). *CI* and *CO* are the carry-in and carry-out (both negative logic).

5. COMPARISON BETWEEN PTM AND PTQ

In the PTQ approach of [4,5] the coder C_X and latches **c** in Fig. 1 are not required, and instead of the adder **a** and decoder D_0, the data input $x(k)$ is input to a shifter with control input α in Fig. 1, either preceded or followed by a negater. A similar negater and shifter are required at each filter tap. The parallel shifter may be efficiently implemented with a 0/1-bit-shift stage followed by a 0/2-bit-shift stage, a 0/4-bit-shift stage, etc. Again assuming that transmission-gates are used, the basic cost of an *N*-bit *K*-stage shifter is 4*NK* transistors. In the case of a down-shifter, the number of transistors is reduced a little if the output wordlength is truncated. In the case of an up-shifter, extra transistors are required. In comparing PTQ and PTM, we will assume a downshifter. The negater can again be replaced by an inverter (costing 2+6*N* transistors). (If the inverter follows the shifter then true negation may be achieved by inputting a carry into the LSB of adder **b**.)

The resulting transistor counts (per filter tap) are shown in Table 1. For the PTM approach, the count covers the latches **c**, the adders **a**, decoders D_i, and modified adder/subtracters **b**. For PTQ, the count covers the shifter, inverter and adder **b**. Three examples are used:

System	PTQ	PTM
8-bit wordlength, 3-bit+sign coder	381	438
12-bit wordlength, 4-bit+sign coder	620	632
16-bit wordlength, 4-bit+sign coder	844	770

Table 1 - LMS: Transistor cost per filter tap

The cost of the coder C_X for the PTM scheme (one per whole filter) is not included above. (It is 110 - 250 transistors, for input wordlength 8-16 bits.)

From Table 1, we conclude that the implementation costs of PTM and PTQ in the LMS algorithm are very similar. The additional implementation cost of the NLMS algorithm (that is, the computation and coding of $g(k) + a$, and subtraction of G from E) is identical for the two approaches, so the same conclusion applies to NLMS. The adaptation performance of the two methods is similar, but PTM is slightly noisier than PTQ. Therefore it is probable that PTQ is preferable for (N)LMS implementation.

We have carried out a similar comparison for the GAL algorithm, and although the costs are again similar, the advantage of the PTQ approach is slightly greater still.

6. APPLICATION TO MEDIAN-LMS

In [11] the median-LMS algorithm is described, which has greatly improved robustness against impulsive noise. In this algorithm, a length *K* median filter is introduced at the input of *each* coefficient update adder (point **d** in Fig. 1). The median filter outputs the median of its most recent input and the *K*-1 preceding inputs. A hardware implementation [11] can exploit the fact that at each sample time the previous *K* values are already ordered in value. The oldest input must be deleted, and the new input must be placed in its correct place in the ordered list; this requires *K*-1 comparisons. A comparison is equivalent to a subtraction followed by testing the MSB of the result.

In the PTQ approach, the correction terms output from the shifter and input to the median filter are of wordlength equal to the coefficient wordlength *N*.

In the PTM approach, because $C(u)$ is a monotonic function of $| u |$, the median filtering may instead be performed on the composite value $z = S(u) \times C(u)$, before decoding. The pair $(S(u), C(u))$ is a *J*-bit sign+magnitude representation of *z*, and a comparator for numbers in this form comprises a (*J*-1)-bit XOR (to invert one of the inputs, if the signs of the two inputs are the same) followed by a (*J*-1)-bit adder and a small amount of additional logic.

Hence within the median filter, both the stores and the comparators cost less, in a ratio approaching $(J\text{-}1) / N$. Because the output of the median filter is in coded form, the decoder D_0 follows the median filter and the simplified adder/subtracter shown in Fig. 2 can again be used.

Examples of cost per filter tap are given in Table 2, assuming the median filter length $K = 5$. The values in Table 2 are the sum of the cost shown in Table 1 and the cost of *K* stores and *K* comparators; the other median filter circuitry is common to either approach.

System	PTQ	PTM
8-bit wordlength, 3-bit+sign coder	2441	1298
12-bit wordlength, 4-bit+sign coder	3760	1782
16-bit wordlength, 4-bit+sign coder	5064	1920

Table 2 - MLMS: Transistor cost per filter tap

A slight further saving may be achieved by converting the sign+magnitude data at the median filter input to 2s-complement words of the same wordlength, and implementing the median filter using conventional comparators.

6.1 Adaptation Performance

Test data was generated consisting of a binary data signal (±64) input to an FIR filter with impulse response [1.0, -0.1, 0.1, 0.0, 0.05, 0.15, -0.05], with noise added together with "impulsive" "spikes", as shown in Fig. 3(a).

Fig. 3(b) shows the learning curve (averaged over 10 trials) of a standard LMS adaptive filter of length *M* = 7, in system identification mode. For clarity plots 3(b-d) are scaled so that 0dB implies (rms error = 64).

Fig. 3(c) shows the learning curve of a floating-point median-LMS adaptive filter of length *M* = 7, with median filter length *K* = 5. Note the reduced vertical axis range.

Fig. 3(d) shows the learning curve of a median-LMS adaptive filter of length *M* = 7, with median filter length *K* = 5, implemented using the PTQ/PTM method described on the previous page. Note that although the plot is different in detail, both the peak error response to input spikes, and the converged error level, are very similar to those of the full floating-point median-LMS filter.

8. CONCLUSIONS

The PTQ method using shifters [4-5] is the best way to implement reduced-implementation-cost (N)LMS and (N)GAL algorithms with very little adaptation performance impairment. However, the Power-of-Two Multiplication method described in this paper results in a significantly lower-cost implementation of the median-LMS filter.

REFERENCES

1. Clarkson, PM, *Optimal and Adaptive Signal Processing*, CRC Press Inc., 1993.

2. Duttweiller, D, "Adaptive Filter Performance with Non-linearities in the Correlation Multiplier", *IEEE Trans ASSP*, 30, 4, August1982, pp578-586.

3. Xue, P and Liu, B, "Adaptive Equalizer Using Finite-Bit Power-of-Two Quantizer", *IEEE Trans ASSP*, 34, 6, Dec 86, pp1603-1611.

4. Reed, MJ and Liu, B, "Analysis of Simplified Gradient Adaptive Lattice Algorithms Using Power-Of-Two Quantization", *IEEE Int Symp Circs Systs*, May1990, pp792-795.

5.Evans, JB, Xue, P, Liu B, "Analysis and Implementation of variable step size adaptive algorithms", *IEEE Trans SP*, 41, Aug 93, pp2517-2535.

6. Duttweiller, D, "A twelve-channel digital echo canceler", *IEEE Trans Commun*, COM-26, May 78, pp647-653.

7. Cheng, S and Evans, J, "Implementation Of Signal Power Estimation Methods", IEEE Trans Circ Systs-II, 44, 3, Mar 97, pp 240-250.

8. Eweda, E, "Convergence Analysis and Design of an Ada-ptive Filter with Finite-Bit Power-of-Two Quantized Error", *IEEE Trans. Circs Systs-II*, 39, 2, Feb 92, pp113-115.

9. Porat, DI, and Barna, A, "Introduction to Digital Techniques", Wiley, NewYork, 1979.

10. Haweel, T, and Clarkson, PM, "A Class of Order Statistic LMS Algorithms", *IEEE Trans SP*, 40, 1, Jan 92, pp 44-53.

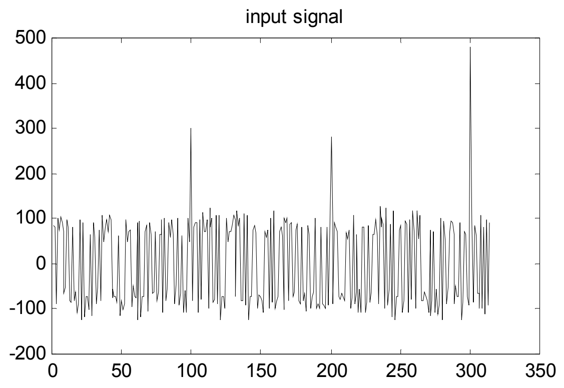

Fig. 3(a) Input data to adaptive filter

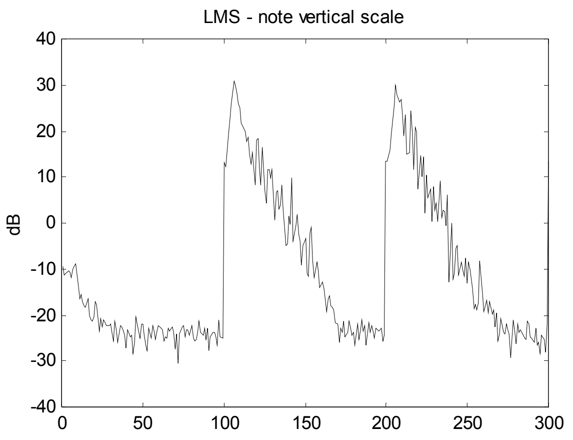

Fig. 3(b) Learning curve of LMS filter

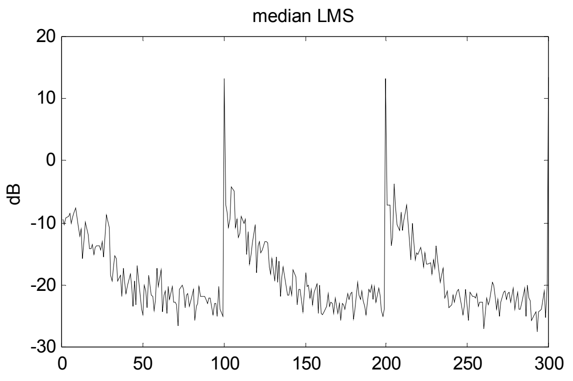

Fig. 3(c) Learning curve of median-LMS filter

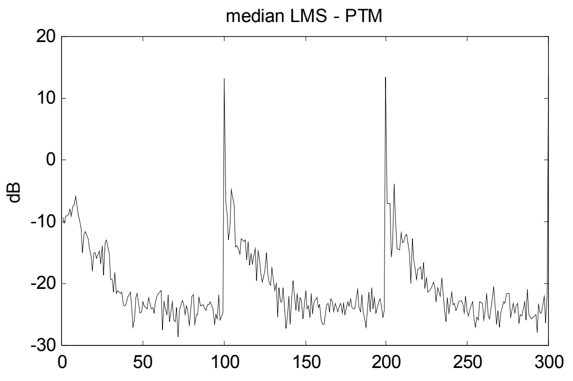

Fig. 3(d) Learning curve of PTM-median-LMS filter