ON EFFICIENT MULTIPLIER-FREE IMPLEMENTATION OF CHANNEL ESTIMATION AND EQUALIZATION

M. Rupp and H. Lou
Bell Labs, Lucent Technologies
Murray Hill, New Jersey, U.S.A.
e-mail: {rupp,lou}@lucent.com

Abstract – In state-of-the-art digital communication systems, channel estimation and/or equalization is an indispensable part of the system. Due to limited resources, only relatively simple algorithms with low complexity can be applied for channel estimation and equalization in high data-rate systems. In fact, the complexity of these algorithms can become prohibitive for very high bit-rate applications, for example, several 100 Msymbols-per-second.

In this paper, we explore the signal constellation structure of practical digital communication systems and describe an efficient computation technique that not only eliminates multiplications but also minimizes the number of addition operations required to implement channel estimation algorithms such as the Least-Mean-Squares (LMS) estimation and equalization algorithms such as the Maximum-Likelihood (ML) sequence detection using the Viterbi algorithm. This new technique preserves the numerical precision of the algorithms while it reduces their complexity dramatically.

I. Introduction

In this paper a new low-complexity technique is proposed that can be applied to many algorithms in digital transmitters and receivers. Our method requires only selection operations, i.e., no adders or multipliers, to implement the multiplication of a complex-valued filter tap coefficient with that of a Quadrature Phase Shift Keying (QPSK) constellation point and only add/sub structures for higher modulation schemes like 16- or 64-Quadrature Amplitude Modulation (QAM).

Channel estimation based on the LMS method and equalization using the Viterbi algorithm is by now a standard technique. The effort to transmit higher data rates, however, is hampered by the technology that can be employed for the implementation of such algorithms. In their original form, they require many complex multiplications: $2M$ for the LMS and PM for a full search Viterbi, where M is the order of the channel estimator, and P is the size of the symbol alphabet.

Early digital implementations[1]-[3] of LMS tried to reduce multiplications by using either signed approximations of the regression vector and/or the error signal. These nonlinear methods, however, alter the training behavior. In general, the training speed is considerably reduced. Thus, only in situations where long training sequences are available (like broadcasting) these methods can be applied.

The idea proposed in this paper is of a different nature. The algorithms will remain numerically exact in their implementation. Thus, their learning behavior remains unchanged. The essential improvement is as a result of exploiting the structures of most commonly used modulation schemes such as the QPSK and QAM constellations. It will be shown that for these typical modulation schemes all operations on the symbols of this alphabets can be implemented numerically exact with a few add and shift operations only. The proposed designs in this paper are extensions of the method designed for real values given in [4].

II. Basic Concept

The basic idea of our design will be explained as an example using a QPSK modulation. Rather than using the conventional constellation, shown in Figure 1, with the constellation points, $u_i \in \{1/\sqrt{2}(\pm 1, \pm j)\}$, a rotated constellation with points $u_i \in \{1, j, -1, -j\}$ is used instead. Rotating the constellation points does not change the behavior of the digital modulation scheme nor its transmission since the hardware dependent implementation as well as the transmission channel will add arbitrary rotations anyway.

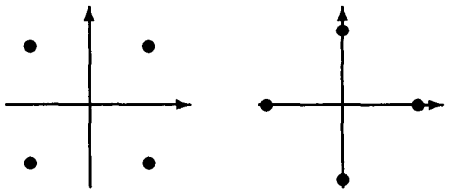

Figure 1: *Conventional (left) and alternative (right) constellations for QPSK modulation.*

In [5] it is already mentioned that using $u_i \in (\pm 1 \pm j)$ rather than $u_i \in 1\sqrt{2}(\pm 1 \pm j)$ saves complexity since multiplications can be performed as add/sub operations. However, the rotated constellation as proposed here simplifies this operation even further. To illustrate this, let us implement an inner vector product, commonly used in LMS and Viterbi algorithms. The elements of the channel estimate, for example, are combined in a row vector $\mathbf{w} = [w_1, w_2, ..., w_M]$, while the transmitted modulated symbols are represented by a row vector $\mathbf{u} = [u_1, u_2, ..., u_M]$. The inner vector product to compute is thus given by

$$\mathbf{u}\mathbf{w}^T = \sum_{i=1}^{M} u_i w_i \ ,$$

0-7803-6451-1/00/$10.00 © 2000 IEEE

with T denoting the transpose operation. In other words a complex multiplication is required to multiply each transmitted symbol u_i, where $i = 1..M$, with one of the channel estimate weights w_i, where $i = 1..M$. A complex multiplication usually requires four real multiplications and two add/sub operations. That is,

$$\begin{aligned} u_i w_i &= \text{Real}(u_i)\text{Real}(w_i) - \text{Imag}(u_i)\text{Imag}(w_i) \\ &+ j\{\text{Real}(u_i)\text{Imag}(w_i) + \text{Imag}(u_i)\text{Real}(w_i)\} \end{aligned}$$

If, however, taking the structure of the QPSK alphabet, $u_i \in \{1, j, -1, -j\}$, into account, the multiplications completely disappear, and only the following four cases remain:

$$u_i = 1: \quad u_i w_i = \text{Real}(w_i) + j\text{Imag}(w_i) \qquad (1)$$

$$u_i = j: \quad u_i w_i = -\text{Imag}(w_i) + j\text{Real}(w_i) \qquad (2)$$

$$u_i = -1: \quad u_i w_i = -\text{Real}(w_i) - j\text{Imag}(w_i) \qquad (3)$$

$$u_i = -j: \quad u_i w_i = \text{Imag}(w_i) - j\text{Real}(w_i)\,. \qquad (4)$$

In other words, the multiplication becomes a selection operation. Even the add/sub operations is not necessary. We denote the two bits that define a QPSK constellation point, bit b_1 and bit b_0. A possible mapping scheme to map these two bits into a QPSK constellation point is shown in Table 1. Arbitrary mappings, such as using Gray coding, can be used by applying a conversion mapping first.

b_1	b_0	u
0	0	1
0	1	j
1	0	-1
1	1	-j

Table 1: *Mapping two bits into a QPSK constellation point.*

To compute the inner vector product of length M, however, addition operations cannot be eliminated. Assume that the partial sum is computed unto position $(k-1)$:

$$s_{k-1} = \sum_{i=1}^{k-1} u_i w_i\,. \qquad (5)$$

Then the next step to compute s_k is

$$s_k = s_{k-1} + u_k w_k\,, \quad k = 2..M. \qquad (6)$$

Finally, Figure 2 displays a fully recursive structure that can perform the complete FIR operation in (5) without requiring additional hardware.

A. QAM Constellations

How does the proposed technique work for larger signal constellations? As an example 16-QAM is shown to illustrate the basic principle. Similar to the QPSK case, the set of 16 constellation points, $u_i \in \{\pm 1 \pm j, \pm 1 \pm 3j, \pm 3 \pm j, \pm 3 \pm 3j\}$ is rotated by 45^o as shown in Figure 3. The figure also shows how the constellation can be separated into four subsections. Each subsection is a translated QPSK constellation. Note that the center point of each subsection is a QPSK constellation. Thus, the selection of the center point of the subsection is the same as selecting a point in the QPSK constellation as described in the previous section. In the second step, the actual signal is selected. This is another selection process, similar to the one before. Note, however, that in the first step the value corresponding to the center point of each subsection is twice as large as the correction signal that is to be added. Thus, before the second step a shift operation is required. In other words, in order to implement a multiplication with a symbol from a 16-QAM constellation, the corresponding channel weight is now required to be selected as described in (1)-(4), followed by a shift operation and finally added by another selected value. Note that the definition of one unit in actual implementation is arbitrary.

B. Minimal Operations

The methods as explained so far allow not only considerable reductions in complexity but are also well suited for pipelined implementations. In some situations, it is of importance to compute the whole set of possible outcomes when multiplying a symbol from a limited alphabet size with a complex value w_l. In this case the complexity can be reduced further since many intermediate results can be re-used.
Take the first quadrant for a 16-QAM constellation as an example (Figure 3), the four possible coefficients are

$$A + jB = w_R + jw_I, \qquad (7)$$

$$C + jD = 3w_R + 3jw_I, \qquad (8)$$

$$E + jF = 2w_R - w_I + j(2w_I + w_R), \qquad (9)$$

$$G + jH = 2w_R + w_I + j(2w_I - w_R)\,. \qquad (10)$$

The operation in the first line is free, the second line costs two adds and so does the third and fourth. Since all other values can be derived from multiplying several times by j, the values are obtained by flipping the real and imaginary values and inverting them. Since only eight different values (A-H) are involved, additional eight inverters are required. Thus, the complete cost is 6 adders and 8 inverters, or equivalently, 14 add/sub operations. Similarly, if this method is applied to 64-QAM, 36 add/subs plus 32 inverters is required.

III. Implementation Examples

This section applies the new techniques and designs introduced in the previous section to implementing the FIR filters, LMS algorithms and the MLSE using the Viterbi algorithm.

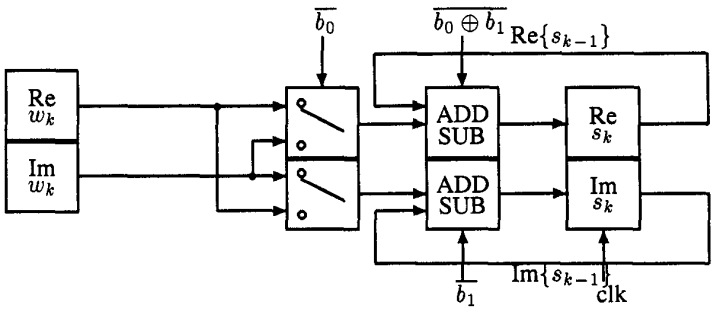

Figure 2: *Recursive add/sub operation for complete vector product computation.*

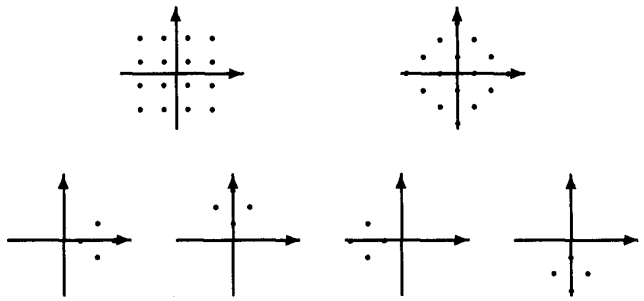

Figure 3: *Rotating a 16-QAM constellation and splitting it into four subsets.*

A. FIR Filter

An example to implement the multiplications in an FIR filter has already been described in the previous section. However, a comparison on complexity has not been given. This will be delivered here. A conventional FIR filter with M complex valued coefficients requires $4M$ real multiplications and $(4M-2)$ real additions. Table 2 shows the number of real add/sub operations required when applying the new technique, given in the previous section. Even for large constellations such as 1024-QAM, the new technique can offer considerably lower complexity. An alternative (as proposed in [4]) is to combine all first stages of the multiplication operations first, then the second and finally the third stages. This approach can save mantissa length and thus chip area. Typical values of M are around three to 64 while 1024-QAM or smaller constellations are typically used.

Note however that the concept of implementing FIR filters of high order requires a long chain of adders. Many applications can apply pipelining to this structure in order to increase throughput at the expense of increased latency and increased chip area due to additional registers. Also note that the new selector technique requires only a few bit of information to be stored for each symbol (for example two for QPSK) so that a pipelining technique with its additional registers does not require much more area.

	Proposed	Conventional
Modulation	No. Add/Subs	No. Add & Mult.
QPSK	$2M-2$	(4M-2) & 4M
16-QAM	$3M-2$	(4M-2) & 4M
64-QAM	$4M-2$	(4M-2) & 4M
256-QAM	$5M-2$	(4M-2) & 4M
1024-QAM	$6M-2$	(4M-2) & 4M

Table 2: *Number of add/sub operations required to implement an FIR filter with M complex-valued taps.*

B. LMS Algorithm

The LMS algorithm is known to be of $2M$ complexity, and $8M$ real multiplications for a complex input. Its complexity is defined by two steps:

$$e(i) = d(i) - \mathbf{u}_i \mathbf{w}_i^T \,, \tag{11}$$

the error equation with

$$\mathbf{u}_i = [u(i), u(i-1), ..., u(i-M+1)], \tag{12}$$

and the coefficient update equation:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \mu e(i) \mathbf{u}_i^* \,, \tag{13}$$

with

$$\mathbf{w}_i = [w_i(0), w_i(1), ..., w_i(M-1)], \qquad (14)$$

denoting $w_i(l)$ the tap weight at time instant i with index l ranging from 0 to $M-1$. If the method as explained in the previous section is applied, the multiplications for the error $e(i)$ computation can be substituted by select and add/sub operations. To compute the coefficient update (13), if the step-size μ is a power of two ($\mu = 2^{-t}$), the multiplication $\mu e(i)$ can be replaced by a simple shift operator. The remaining multiplication with the symbols u_i can also be achieved with the new technique introduced. All that remains is to update the coefficient which is a complex addition. A QPSK constellation requires $2M$ add/sub operations to evaluate the error equation as well as updating the coefficient. For 16-QAM, the error computation requires $4M$ add/sub operations, the multiplication $e(i)\mathbf{u}_i^*$ requires $2M$ add/sub operations and finally, the coefficient update requires $2M$ add/sub operations. That is, $8M$ add/sub operations are required altogether. In summary, $4M$, $8M$ and $12M$ add/sub operations are required for QPSK, 16-QAM and 64-QAM constellations respectively. Thus when the constellation size is increased by four times, another $4M$ additional operations are required (see Table 3).

Modulation	Complexity	Minimum latency
QPSK	$4M$	$2 + \log_2 M$
16-QAM	$8M$	$4 + \log_2 M$
64-QAM	$12M$	$6 + \log_2 M$

Table 3: *Number of add/sub operations for LMS.*

As mentioned in the FIR section before, faster realizations are sometimes required. For the LMS algorithm, the update cannot be performed until the error signal is available. Thus, a very rapid FIR filter chain is important. This can be implemented with the hierarchical tree structure. The computation of the error signal requires an additional subtraction: $e(i) = d(i) - \mathbf{u}_i\mathbf{w}_i^T$. If the order of the filter is chosen to be $M = 2^L - 1$, $d(i)$ can be treated as one filter tap-weight element and does not require additional delay. As previously mentioned, depending on the sign of the last signal (in this case the error signal), a final two's complement inverter is required. This would cost an additional delay since two's complement inverters cause carry-ripple effects. However, in the LMS algorithm the last inverter can be incorporated into the selection process of the coefficient update. If the negative information is active, $-\mu e(i)$ is applied rather than $\mu e(i)$. The modified selection is faster because it is realized by simple logical gates and does not require an adder-structure. For QPSK, one complete update cycle requires $T_a \log_2 M$ for the FIR error-part, possibly one T_a for the error to multiply by the step-size μ, and finally one add-operation for the updates of the coefficients. Thus, the minimum update time is given by $(\log_2 M + 2)T_a$. Table 3 lists the operations required to implement the LMS for selected modulation schemes. If the step-size μ is a negative power of two, the multiplication by the error is a simple scaling operation and can therefore be realized without any additional time delay. Other values of the step-size can be approximated with a sum of two such values $\mu = 2^{-l_1} \pm 2^{-l_2}$, so that one add/sub operation is sufficient for the multiplication with the error. This operation gives a wide range for possible step-size values.

To give an example, let us assume that the LMS algorithm is applied to train a channel estimator of order $M = 15$, and the technology used implements an add/sub operation in 1ns and a multiplication operation in 6ns. Thus, the whole update for a BPSK/QPSK training sequence can be performed in 6ns with add/sub operations while it takes about 12ns when using multipliers. The required chip area and power consumption, on the other hand, might become ten times higher to implement the multipliers. Real-time processing is thus possible for up to 1/6ns = 166 Msymbols-per-second in this case.

C. Maximum-likelihood sequence detection

In a digital communication system that transmits information over a channel causing Inter-Symbol-Interference (ISI), the optimum detector is a maximum-likelihood symbol sequence detector [6, 7]. An efficient algorithm that implements the maximum-likelihood sequence detection (MLSD) is the Viterbi algorithm, which was originally devised for decoding convolutional codes [6]-[8]. In this case, the ISI channel is modeled as a Finite-State Machine (FSM), called a trellis, with P^{M-1} states. Here P is the information symbol alphabet size and M is the number of the complex-valued channel FIR filter coefficients, w_l. In the trellis, there are P transitions diverging from each state, corresponding to the P different values of the information symbol, $u(k)$. The values associated with the transitions between the states are $w_l u(k)$, the possible received values, given the estimated channel coefficients w_l.

Assume that the received sequence is $r(k)$, the estimated channel coefficients are w_l, and the input information symbols are $u(k)$, the Viterbi algorithm finds the most-likely transmitted symbol $\vec{u}(k)$ by finding the path in the trellis that is closest in Euclidean distance to the received noisy sequence $r(k)$ recursively. That is, it implements the ML detector criterion by recursively minimizing w.r.t. $\vec{u}(k)$,

$$\min_{\vec{u}(k)} \left| r(k) - \sum_{l=1}^{M} w_l u(k-l) \right|^2 \qquad (15)$$

At each recursion step, the Viterbi algorithm searches over the P^M possible transitions. If the channel model coefficient changes from one recursion to the other, for every one of the M estimated channel coefficients w_l, P multiplica-

tions with the P symbols of the symbol alphabet are required. Thus, a total of $PM = O(PM)$ complex multiplications are required to compute the various terms initially. In order to compute the cost metrics of the P^M transitions between the states in the FSM, another $(M-1) \times P^M = O(MP^M)$ complex additions are required. If this is done using the traditional method, a complex multiplication is performed with four to eight add/sub operations depending on the symbols. The value $(3+7j)$, for example, requires one addition for the multiplication by three (3=1+2) and one subtraction for seven (7=8-1). This needs to be performed on the real as well as on the imaginary part of the coefficient, resulting in $O(PM) = 4PM$. Finally, the real and imaginary part needs to be added to $r(k)$ which costs another 2 add operations. For all M coefficients and P^M states thus $O(MP^M) = 2MP^M$ add operations.

Using our proposed technique given in Section II.A, complexity can be reduced considerably. For QPSK, multiplication of each coefficient w_l with $u(k)$ becomes a selection process, and $O(PM) = 0$. Only the computation of the transitions remains with $O(MP^M) = 2MP^M$ to compute $\tilde{r}(k) = r(k) - \sum_{l=1}^{M} w_l u(k-l)$ in (Equation 15). For 16-QAM, the first step is a selection process, the next is two add/subs to compute each multiplication per symbol, thus $O(PM) = 2PM = 32M$. For 64-QAM in each subset of four symbols 10 add/subs are required for computing the multiplication, thus $O(PM) = 160M$. The application of this technique has the additional advantage that it can readily be pipelined without adding too much chip area for the additional registers.

At the expense of increasing pipelining complexity, using the minimal operations described in Section II.B can reduce the computation complexity even further. For example, for the 16-QAM modulation, $O(PM) = 14M$ and for 64-QAM, $O(PM) = 68M$.

Initializing the trellis at the beginning of an equalization process requires computing all possible multiplications with all the elements in the symbol alphabet once. This assumes that the channel remains constant over a frame of data. If the channel is rapidly changing from symbol to symbol, this initialization has to be performed at every recursion to update the transition values of the trellis. The so obtained values can be stored in a look-up table for computing the Euclidean norm in the Viterbi algorithm. The complexity for initialization ($O(PM)$) is listed in Table 4. In order to obtain the complete complexity of the Viterbi algorithm $O(MP^M)$ needs to be added if a full search through the trellis is applied. This part can easily exceed the initial complexity of $O(PM)$. Reduced complexity techniques, such as using reduced-state sequence estimation techniques [9], that limit the search through the trellis are often applied. In this case, the initial complexity can become very significant. A tree structure for implementing the add operations can further reduce the complexity as described in Section III.A.

Mod.	Conv.	Sec. II.A	Minimum
QPSK	$2 \times 4 = 8$	0	0
16-QAM	$4 \times 16 = 64$	32	14
64-QAM	$4 \times 64 = 256$	160	68

Table 4: *Number of add/sub operations per coefficient to initialize trellis of the Viterbi algorithm applying different methods.*

IV. Conclusions

A novel efficient computation method has been proposed allowing large reduction of complexity for implementation of standard communication algorithms that requires multiplying a coefficient with a constellation point in a QPSK or QAM constellation. Chip area as well as latency can be reduced due to the substitution of multiplications by simpler functions. Note that applying this technique does not cause any approximation or reduction on precision. The algorithms remain bit-true exact.

1. REFERENCES

[1] T.A.C.M. Claasen, W.F.G.Mecklenbräuker, "Comparison of the convergence of two algorithms for adaptive FIR digital filters," IEEE Trans. on Acoust., Speech, Signal Proc., no. 3, pp. 670–678, June 1981.

[2] D.L.Duttweiler, "A twelve–channel digital echo canceler," IEEE Trans. on Com., vol. COM–26, no. 5, May 1978.

[3] R.D.Gitlin, J.E.Mazo, M.G.Taylor, "On the design of gradient algorithms for digitally implemented adaptive filters," IEEE Trans. on Circuit Theory, vol. 20, no. 2, pp. 125–136, Mar. 1973.

[4] M.Rupp, J.Balakrishnan, "Efficient Chip Design for Pulse Shaping," SPAWC 99, Annapolis, May 1999.

[5] G.M.Durant, S.Ariyavisitakul, "Implementation of a broadband equalizer for high-speed wireless data applications," Proc. IEEE ICUPC 98, Florence, Italy, Oct. 1998.

[6] J. G.Proakis, "Channel equalization," in *The Communications Handbook*, (CRC Press), 1997. Chapter 26.

[7] G. D.Forney, Jr., "Maximum-likelihood sequence detection in the presence of intersymbol interference," *IEEE Trans. on Inf. Th.*, vol. IT-18, pp. 363–378, May 1972.

[8] G. D.Forney, Jr., "The Viterbi algorithm," *IEEE Proceedings*, vol. 61, pp. 268–278, March 1973.

[9] M. Eyuboglu and S. Qureshi, "Reduced-state sequence estimation for coded modulation on intersymbol interference channels," *IEEE Journal on Sel. Areas in Com.*, vol. 7, pp. 989–995, Aug. 1989.