TABLE I
COMPARISON OF $S_n(k)$ WITH UPPER AND LOWER BOUNDS

(n,k,ϵ)	lower bd.(4)	lower bd.(5)	exact	upper bd.(6)	upper bd.(4)
100,2,.001	4.48769E-3	4.63757E-3	4.63806E-3	4.63934E-3	4.63940E-3
100,5,.001	6.84612E-8	6.95605E-8	6.95609E-8	6.95635E-8	6.95637E-8
100,10,.001	1.58197E-17	1.59502E-17	1.59502E-17	1.59503E-17	1.59504E-17
100,2,.01	1.84865E-1	2.52917E-1	2.64238E-1	2.75239E-1	2.75904E-1
100,5,.01	2.89779E-3	3.42731E-3	3.43232E-3	3.44817E-3	3.44947E-3
100,10,.01	7.00604E-8	7.63088E-8	7.63158E-8	7.63652E-8	7.63721E-8
100,20,.01	2.39865E-20	2.49440E-20	2.49441E-20	2.49459E-20	2.49464E-20
100,15,.1	3.26824E-2	6.74719E-2	7.25729E-2	7.80727E-2	7.97673E-2
100,25,.1	8.97293E-6	1.30532E-5	1.30728E-5	1.31676E-5	1.32054E-5
100,50,.1	5.19971E-24	5.83192E-24	5.83203E-24	5.83366E-24	5.83537E-24
1000,2,.001	1.84032E-1	2.52917E-1	2.64241E-1	2.75840E-1	2.75910E-1
1000,10,.001	9.78284E-8	1.07417E-7	1.07428E-7	1.07513E-7	1.07514E-7
1000,25,.001	1.79614E-26	1.86607E-26	1.86608E-26	1.86618E-26	1.86619E-26
1000,15,.01	3.45417E-2	7.56543E-2	8.24123E-2	9.10951E-2	9.13424E-2
1000,25,.01	2.64426E-5	4.19411E-5	4.20292E-5	4.25498E-5	4.25661E-5
1000,50,.01	6.75067E-20	8.30600E-20	8.30637E-20	8.31476E-20	8.31524E-20
1000,120,.1	4.73891E-3	2.05769E-2	2.19961E-2	2.42322E-2	2.46922E-2
1000,150,.1	1.71500E-7	4.47936E-7	4.48944E-7	4.56417E-7	4.57895E-7
1000,200,.1	1.63938E-21	2.92767E-21	2.92805E-21	2.93688E-21	2.93918E-21

accuracy, especially for moderate to low probabilities. The two upper bounds are close across all parameter values tried. From this numerical evidence and an examination of the expressions, it appears that the degree of improvement of (6) over the upper bound of (4) may not be sufficient to warrant its use for most cases. The lower bound (5) gives good accuracy across a range of parameters and, in particular, gives significantly better results than the lower bound of (4) when ϵ is relatively large and k small. This improvement of the lower bound is achieved at a modest increase in complexity and seems worthwhile for many applications. In general, the upper and lower bounds can be used with confidence in place of exact values when their ratio is close to unity. This will occur, for example, when r_1 is small, which will be true when $k\eta$ is much larger than $n\epsilon$.

In the foregoing computations exact values of $b_n(k)$ were used. However, if efficiency of computation is of concern, then in most cases bounds on $b_n(k)$ could be used along with the bounds of (5) and (6) (or (4)). This is a very well investigated problem and very tight bounds are available. In particular, the following version of Stirling's approximation to the factorial might be used:

$$(2\pi)^{1/2} n^{n+1/2} \exp\left(-n + \frac{1}{12n+1}\right) < n! < (2\pi)^{1/2} n^{n+1/2} \exp\left(-n + \frac{1}{12n}\right).$$

While this approximation yields excellent results, it can easily be improved further by taking more terms in the argument (exponent) of the exponential [5]. This result can be used to upper and lower bound $b_n(k)$ as follows:

$$\frac{1}{(2\pi n\lambda\gamma)^{1/2}} 2^{n(H(\lambda,\gamma)+E(\lambda,\epsilon))} \exp\left(\frac{1}{12n} - \frac{1}{12\lambda n+1} - \frac{1}{12\gamma n+1}\right) \le b_n(k) \le \frac{1}{(2\pi n\lambda\gamma)^{1/2}} 2^{n(H(\lambda,\gamma)+E(\lambda,\epsilon))} \exp\left(\frac{1}{12n+1} - \frac{1}{12\lambda n} - \frac{1}{12\gamma n}\right), \quad \gamma = 1-\lambda,\ \lambda = k/n \quad (7)$$

where $H(\lambda,\gamma) = -\lambda \log_2 \lambda - \gamma \log_2 \gamma$ is the binary entropy function and $E(\lambda,\epsilon) = \lambda \log_2(\epsilon) + \gamma \log_2(\eta)$.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their careful reading of the manuscript and helpful comments.

REFERENCES

[1] T. Kasami and S. Lin, "On the probability of undetected error for maximum distance separable codes," *IEEE Trans. Commun.*, vol. COM-32, pp. 998–1006, 1984.
[2] E. R. Berlekamp, "The technology of error correcting codes," *Proc. IEEE*, vol. 68, pp. 564–593, 1980.
[3] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1, 3rd ed. New York: Wiley, 1968.
[4] W. W. Peterson, *Error Correcting Codes*. Cambridge, MA: MIT Press, 1961.
[5] P. A. P. Moran, *An Introduction to Probability Theory*. London: Oxford Univ. Press, 1968.

On the Equivalence Between Berlekamp's and Euclid's Algorithms

JEAN LOUIS DORNSTETTER

***Abstract*—It is shown that Berlekamp's iterative algorithm can be derived from a normalized version of Euclid's extended algorithm. Simple proofs of the results given recently by Cheng are also presented.**

I. INTRODUCTION

The similarity between Berlekamp's iterative algorithm [1] and the extended Euclidean algorithm [7] has been previously noticed by several authors [3], [4]. Recently, Cheng [6] gave a more explicit description of the relations between the partial results produced by both algorithms. The original version of the iterative algorithm has been simplified by Massey [2]. We shall show in Section II that all partial results generated by this simplified version are in agreement (to a reciprocation and a normalization factor) with those given by Euclid's algorithm. The equivalence of the two algorithms is made more explicit in Section III by demonstrating that the iterative algorithm can be derived from a normalized version of Euclid's algorithm.

II. FORMAL EQUIVALENCE OF THE TWO ALGORITHMS

All polynomials to be considered in the following have their coefficients in a given commutative field K. By convention, the degree of the zero polynomial is $-\infty$. We will first restate some of the basic properties of the extended Euclidean algorithm for computing the greatest common divisor of R_0 and R_{-1} when $R_{-1} = X^n$ and R_0 is the polynomial $S(X) = S_0 + S_1 X + \cdots + S_{n-1} X^{n-1}$ (the syndrome polynomial in the context of decoding BCH codes).

Manuscript received April 19, 1985; revised July 15, 1986.
The author is with the Laboratoire Central de Telecommunications, 18-20 rue Grange Dame Rose, 78141 Velizy-Villacoublay Cedex, France.
IEEE Log Number 8611419.

0018-9448/87/0500-0428$01.00 ©1987 IEEE

The extended Euclidean algorithm generates a sequence of polynomials U_i, V_i, R_i, $i \geq 0$ defined by the rules:

$$U_{-1} = V_0 = 0 \quad U_0 = V_{-1} = 1 \quad R_{-1} = X^n \quad R_0 = S(X)$$

$$R_{i-1} = Q_i R_i + R_{i+1} \qquad \deg R_{i+1} < \deg R_i$$

(which defines Q_i and R_{i+1})

$$U_{i+1} = Q_i U_i + U_{i-1} \qquad V_{i+1} = Q_i V_i + V_{i-1}.$$

The process eventually terminates with, say, $R_m = 0$.

The following properties are easily shown by induction on i:

P1) $\deg R_i < n - \deg U_i$,
P2) $U_i V_{i-1} - U_{i-1} V_i = (-1)^i$,
P3) $U_i S - V_i X^n = (-1)^i R_i$.

Combining P1) and P3), we obtain

P4) $\deg(U_i S - V_i X^n) < n - \deg U_i$.

P4) is strong in the sense that it admits a converse, as follows.

Theorem 1: Let P and Q be two relatively prime polynomials such that

$$\deg(QS - PX^n) < n - \deg Q.$$

Then, there exist a nonzero scalar c and an integer $i \leq m$ such that

$$Q = cU_i \qquad P = cV_i.$$

For a proof, see [4, th. 8.5], or [5].

Let us define, for some fixed $i < m$,

$$S^*(X) \triangleq X^{n-1} S(X^{-1}) \qquad d \triangleq \deg U_i,$$

$$\sigma(X) \triangleq X^d U_i(X^{-1}) \qquad \omega(X) \triangleq X^{d-1} V_i(X^{-1}).$$

Lemma 1: a) $\sigma(0) \neq 0$. b) $d = \max(\deg \sigma, 1 + \deg \omega)$. c) The pair (σ, ω) satisfies:

$$\sigma S^* \equiv \omega \bmod X^{2d}.$$

Proof: a) This follows trivially from the fact that $d = \deg U_i$. b) By P2), U_i and V_i are relatively prime; thus either $U_i(0) = 0$ and then $\deg \sigma < d$ but $V_i(0) \neq 0$ and hence $\deg \omega = d - 1$, or $U_i(0) \neq 0$ and $\deg \sigma = d$, but always $\deg \omega < d$. c) Dropping the index i, we have by P4)

$$U(X) S(X) = V(X) X^n + R(X)$$

and

$$\deg R \leq n - 1 - d.$$

Multiplying by X^{1-d-n} yields

$$X^{-d} U(X) X^{1-n} S(X) = X^{1-d} V(X) + X^{1-d-n} R(X).$$

This may be written as

$$X^d U(X^{-1}) X^{n-1} S(X^{-1}) = X^{d-1} V(X^{-1}) + X^{2d}\left(X^{n-1-d} R(X^{-1})\right)$$

or

$$\sigma(X) S^*(X) = \omega(X) + X^{2d} Q(X),$$

which concludes the proof.

Conversely, every pair of relatively prime polynomials (σ, ω) such that

$$\sigma(0) \neq 0, \ \sigma S^* \equiv \omega \bmod X^{2d}, \ \max(\deg \sigma, 1 + \deg \omega) = d$$

may be converted by reversing the above process into a pair

$$P(X) = X^{d-1} \omega(X^{-1}) \qquad Q(X) = X^d \sigma(X^{-1})$$

such that $\deg Q = d$, $\deg(QS - X^n P) < n - d$.

By Theorem 1, such a pair will be found by Euclid's algorithm. We now remark that the quantity $\max(\deg \sigma, 1 + \deg \omega)$ is just a compact definition of Massey's minimum "shift register length." It is the smallest number of shift register stages that are necessary to generate linearly the power series $\omega(X)/\sigma(X)$. This informal argument can be made precise, as the following result shows.

Theorem 2: The Berlekamp–Massey algorithm solves the following problem: for each j, $0 \leq j \leq n$, find a pair (σ_j, ω_j) such that $\sigma_j(0) = 1$, $\sigma_j S \equiv \omega_j \bmod X^j$ and $d_j = \max(\deg \sigma_j, 1 + \deg \omega_j)$ is minimum.

The proof of Theorem 2, together with a slight variant of Massey's version of the iterative algorithm, is presented in the Appendix.

Defining the degree of an ordered pair of polynomials (σ, ω) by

$$\deg(\sigma, \omega) = \max(\deg \sigma, 1 + \deg \omega)$$

we have the following unicity result.

Lemma 2: If, given j, the problem of Theorem 2 admits a solution (σ_j, ω_j) of degree d, and $2d \leq j$, then this is the only solution.

Proof: Suppose there are two pairs (a, b) and (e, f) that solve the problem and are of degree d. We have

$$aS^* \equiv b \bmod X^j \qquad eS^* \equiv f \bmod X^j.$$

Thus $af - be \equiv 0 \bmod X^j$, and since $2d \leq j$, this congruence is an equality. The conclusion now follows from the fact that a and b are relatively prime since d is assumed to be the minimum possible.

The relationship between the two algorithms is now clear: every pair (σ, ω) such that $2d \leq j$ yields a pair (Q, P) such that

$$\deg(QS - X^n P) < n - \deg Q$$

and this pair is necessarily found, within a constant factor, by Euclid's algorithm (Theorem 1).

Conversely, every output (U_i, V_i) generated by Euclid's algorithm may be converted into a pair (σ, ω) such that

$$\deg(\sigma, \omega) = \deg U_i = d \qquad \sigma S^* \equiv \omega \bmod X^{2d}.$$

By Lemma 2, such a pair will be found by the iterative algorithm for $j = 2d$.

If we denote by $l_0, l_1, \cdots$ the successive values assumed by d_j, $(l_0 = 0)$, it is easy to verify by inspecting the behavior of the iterative algorithm that

$$\deg(\sigma_{2l_k}, \omega_{2l_k}) = l_k.$$

Thus we have proved the following theorem.

Theorem 3: For each k, $0 \leq k \leq m$, there is a nonzero scalar C_k such that

$$X^{l_k} U_k(X^{-1}) = C_k \cdot \sigma_{2l_k}(X)$$

$$X^{l_k} V_k(X^{-1}) = C_k \cdot \omega_{2l_k}(X).$$

Of course, not all partial results generated by the iterative algorithm satisfy $2d_j = j$. However, those for which $2d_j < j$ are identical to their predecessors.

For each i we have

$$U_{i+1} = Q_i U_i + U_{i-1}, \ Q_i(X) = q_{e_i} X^{e_i} + q_{e_i - 1} X^{e_i - 1} + \cdots + q_0,$$

$$e_i = l_{i+1} - l_i.$$

If we agree to consider the e_i extra pairs of polynomials defined by

$$U_{i+1}^s = \left(q_{e_i}X^{e_i} + \cdots + q_sX^s\right)U_i + U_{i-1}$$
$$V_{i+1}^s = \left(q_{e_i}X^{e_i} + \cdots + q_sX^s\right)V_i + V_{i-1},$$
$$s = e_i, e_i - 1, \cdots, 1$$

as "outputs" of Euclid's algorithm, then it is possible to check that these additional results (to be inserted between (U_i, V_i) and (U_{i+1}, V_{i+1})) are in one-to-one correspondence, as in Theorem 3, with those of the iterative algorithm for which $2d_j > j$. (This is true provided that the initialization proposed in the Appendix is used. This would become false for the first few iterations, namely, $l_1 \le j < 2l_1$, if Massey's initialization were chosen.)

At this point, we could derive an explicit relation between the successive discrepancies Δ_j in the iterative algorithm and the coefficients of the quotient Q_i, but the various index manipulations are tedious.

Furthermore, such a derivation would not tell us why such a simplification of Euclid's venerable algorithm is possible when R_{-1} has the simple form X^n. (Remember that Euclid's algorithm is roughly twice as complex as the iterative algorithm if we discard ω_j, V_i, whereas both compute the same thing, according to the preceding discussion.)

The aim of Section III is to show how to simplify Euclid's algorithm when $R_{-1} = X^n$, the result of this simplification being nearly identical to the Berlekamp–Massey algorithm.

III. Derivation of the Iterative Algorithm from Euclid's Algorithm

By P3) in Section II we have, for $i \ge 0$,

$$U_{i-1}S = V_{i-1}X^n - (-1)^i R_{i-1}$$
$$U_iS = V_iX^n + (-1)^i R_i.$$

The key observation is that it is not really necessary to track the various R_i explicitly since they appear as the low degree part of U_iS. We really need only the quotients Q_i defined by

$$R_{i-1} = Q_iR_i + R_{i+1}.$$

For each i, we let $t_i \triangleq \deg R_i$ and $\Delta_i \triangleq$ (the leading coefficient of $(-1)^iR_i$). Clearly, the leading coefficient of Q_i is $-\Delta_{i-1}\Delta_i^{-1}$.

Suppose we start with the knowledge of $U_{i-1}, U_i, t_{i-1}, \Delta_{i-1}$ and that we want to compute

$$Q_i = q_{e_i}X^{e_i} + q_{e_i-1}X^{e_i-1} + \cdots + q_0$$

without considering explicitly R_{i-1} and R_i. First, we want to find $e_i = \deg Q_i$; for that purpose we compute the successive b_k, $k = 1, 2, \cdots$, where

$$b_k \triangleq \left(\text{the coefficient of } X^{t_{i-1}-k} \text{ in } U_iS = V_iX^n + (-1)^iR_i\right)$$

until, say, $b_l \ne 0$. Note that b_k is just the coefficient of $X^{t_{i-1}-k}$ in $(-1)^iR_i$. Thus $t_{i-1} - l = t_i$ and $\Delta_i = b_l$.

Since $t_{i-1} = \deg Q_i + t_i$, we also have $e_i = l$. At this point we also know $q_{e_i} = -\Delta_{i-1}\Delta_i^{-1}$, and we can start the computation of U_{i+1} as follows:

$$U \leftarrow q_{e_i}X^{e_i}U_i + U_{i-1}.$$

To obtain the next coefficient q_{e_i-1}, the usual Euclidean division of R_{i-1} by R_i would lead us to consider b, the coefficient of $X^{t_{i-1}-1}$ in $(R_{i-1} - q_{e_i}X^{e_i}R_i)$, and to set

$$q_{e_i-1} = b\left((-1)^i\Delta_i\right)^{-1} = -b(-1)^{i-1}\Delta_i^{-1}.$$

However, $b(-1)^{i-1}$ is precisely a_{e_i-1}, the coefficient of $X^{t_i+e_i-1}$ in US since

$$US = \left(q_{e_i}X^{e_i}U_i + U_{i-1}\right)S \equiv (-1)^i\left(q_{e_i}X^{e_i}R_i - R_{i-1}\right) \bmod X^n.$$

Thus we can go somewhat further in our computation of U_{i+1}, namely,

$$U \leftarrow U - a_{e_i-1}\Delta_i^{-1}X^{e_i-1}U_i = \left(q_{e_i}X^{e_i} + q_{e_i-1}X^{e_i-1}\right)U_i + U_{i-1}.$$

It is now clear that the whole quotient Q_i may be computed in the way described earlier. After $e_i + 1$ iterations, the polynomial U is equal to U_{i+1}.

Furthermore, since we know t_i and Δ_i, we can start the next computation, that of Q_{i+1}. The natural initialization

$$i = 0 \qquad U_{-1} = 0 \qquad U_0 = 1 \qquad t_{-1} = n \qquad \Delta_{-1} = -1$$

completes the resulting algorithm.

We can restate it in a more compact form, using a temporary storage polynomial T, two polynomials U, U', an integer t, and a scalar Δ. At the beginning of one cycle (computation of Q_i), $U = U_i$, $U' = U_{i-1}$, $t = t_{i-1}$, $\Delta = \Delta_{i-1}$; as soon as the first phase (determination of $\deg Q_i$, steps 2) and 3) below) is completed, U' is set to U_i, Δ to Δ_i, t to t_i, and U is used to build up U_{i+1} as in the description given earlier.

Simplified Euclidean Algorithm When $R_{-1} = X^n$

1) Initialization: $U' = 0$, $U = 1$, $t = n$, $\Delta = -1$, $k = 1$.
2) Compute $b_k \triangleq$ (coefficient of X^{t-k} in US).
3) If $b_k = 0$, set $k \leftarrow k + 1$ and return to 2).
4) Now, the next quotient has degree k. Set

$$T \leftarrow U \qquad U \leftarrow U' - \Delta b_k^{-1}X^kU \qquad \Delta \leftarrow b_k$$
$$U' \leftarrow T \qquad t \leftarrow t - k \qquad k \leftarrow 1 - k.$$

5) Compute $a_{-k} \triangleq$ (coefficient of X^{t-k} in US).
6) Set $U \leftarrow U - a_{-k}\Delta^{-1}X^{-k}U'$, $k \leftarrow k + 1$. If $k \le 0$, then return to 5); otherwise, go to 2). (For simplicity, we omitted the stop test: the algorithm should be stopped at least when $t < k$.)

A further simplification is still possible. We can set in step 4): $U \leftarrow c(U' - \Delta b_k^{-1}X^kU)$ for any nonzero scalar c. This simply multiplies the next a_{-k} computed in step 5) by c.

Thus after the cycle is completed, we just have cU instead of U; the first nonzero b_k computed in step 2) at the beginning of the next cycle will also be multiplied by c, so that $\Delta^{-1}U'$ never depends on c.

One reasonable choice for c is

$$c = -b_k\Delta^{-1}.$$

The new step 4) is to set

$$T \leftarrow U \qquad U \leftarrow X^kU - b_k\Delta^{-1}U' \qquad U' \leftarrow T$$
$$t \leftarrow t - k \qquad k \leftarrow 1 - k.$$

We get a normalized version of Euclid's algorithm since U is now always monic.

It may be observed that, between two consecutive executions of steps 2) or 5), the value of $j = n - 1 - t + k + \deg U$ is increased by one, the initial value being zero. Thus if we rephrase the above algorithm using:

$$\sigma = X^{\deg U}U(X^{-1}) \qquad \sigma' = X^{\deg U}U'(X^{-1}),$$

we find exactly the iterative algorithm as described in the Appendix.

We have $d_j = \deg U$, $k = j + 1 - 2d_j$, the a_{-k} are the discrepancies for which $2d_j \le j$ and the b_k those for which $2d_j > j$.

In the general case, when $R_{-1} = G(X)$, an arbitrary polynomial of degree n, it is still possible to derive a similar simplification of the Euclidean algorithm, but now we have to compute the V_i and to replace US by $US - VG$ in steps 2) and 5): it becomes necessary to compute the V_i even if they do not represent a useful output of the algorithm.

TABLE I

Stop test	deg $R_i < n/2$		deg $R_i = 0$	
R_{-1}	X^n	arbitrary	X^n	arbitrary
Euclidean algorithm, U_i only	1		2	
Euclidean algorithm, U_i and V_i	5/4		3	
Generalized Berlekamp-Massey algorithm, σ_j only	1/2	1	3/2	3
Generalized Berlekamp-Massey algorithm, σ_j and ω_j	3/4		1	

If we rephrase this more general version of the simplified Euclidean algorithm using the reciprocal polynomial variables as before, we find a generalized Berlekamp–Massey algorithm that is exactly that given in the Appendix except that the computation of the jth discrepancy Δ_j reads as follows:

$$\Delta_j \triangleq \left(\text{the coefficient of } X^j \text{ in } \sigma_j(X)S^*(X) - \omega_j(X)G^*(X)\right)$$

with

$$G^*(X) \triangleq X^n G(X^{-1}).$$

We remark that when $G(X) = X^n$, it is sufficient to look at $\sigma_j(X)S^*(X)$ since $G^*(X) = 1$ and $\deg \omega_j < j$ for all j.

Depending on the application of the Euclidean algorithm, one might be interested in U_i only, or both in U_i and V_i. The stop test also depends on the application: when computing multiplicative inverses, the Euclidean algorithm is stopped as soon as $\deg R_i = 0$, while if the Euclidean algorithm is used as a Diophantine approximation tool (decoding BCH codes, for example), the algorithm is stopped as soon as $\deg R_i < n/2$.

With $n \triangleq \deg R_{-1}$, we can express the complexity of these algorithms as Kn^2, neglecting linear terms. Table I gives the value of K, depending on what has to be computed, the stop test and the value of R_{-1}.

Appendix
The Berlekamp–Massey Algorithm

1) Initialization:

$$j = 0 \qquad \sigma_0 = -\omega_0' = 1 \qquad \sigma_0' = \omega_0 = 0 \qquad d_0 = 0 \qquad \Delta = 1.$$

2) Compute the jth discrepancy Δ_j as the coefficient of X^j in

$$\sigma_j(X)S^*(X) - \omega_j(X).$$

3) If $\Delta_j = 0$,

$$d_{j+1} \leftarrow d_j \qquad \sigma_{j+1} \leftarrow \sigma_j \qquad \omega_{j+1} \leftarrow \omega_j$$
$$\sigma_{j+1}' \leftarrow X\sigma_j' \qquad \omega_{j+1}' \leftarrow X\omega_j'.$$

4) If $\Delta_j \neq 0, 2d_j > j$,

$$d_{j+1} = d_j \qquad \sigma_{j+1} \leftarrow \sigma_j - \Delta_j \Delta^{-1}\sigma_j' \qquad \omega_{j+1} \leftarrow \omega_j - \Delta_j \Delta^{-1}\omega_j'$$
$$\sigma_{j+1}' \leftarrow X\sigma_j' \qquad \omega_{j+1}' \leftarrow X\omega_j'.$$

5) If $\Delta_j \neq 0, 2d_j \leq j$,

$$d_{j+1} = j + 1 - d_j \qquad \sigma_{j+1} \leftarrow \sigma_j - \Delta_j \Delta^{-1}\sigma_j'$$
$$\omega_{j+1} \leftarrow \omega_j - \Delta_j \Delta^{-1}\omega_j' \qquad \Delta \leftarrow \Delta_j$$
$$\sigma_{j+1}' \leftarrow X\sigma_j \qquad \omega_{j+1}' \leftarrow X\omega_j.$$

6) $j \leftarrow j + 1$, go to step 2).

This is Massey's version of the iterative algorithm, except for the inclusion of the polynomials ω_j, ω_j', the initialization step 1) and the omission of the stop test. Massey's initialization was $\sigma_0' = X$, but ω_0 and ω_0' are unaffected by this change.

The reason for the slight change in the initialization is that, without it, the first few iterations do not give the same result as Euclid's algorithm. This turns out to be the simplest way to obtain a complete agreement with the Euclidean algorithm.

As a result, we do not have $\Delta_j = 0$ for the first few odd j when decoding binary BCH codes, so that the original initialization is better from the computational point of view (see [1] for the details).

Proof of Theorem 2: The proof of Theorem 2 is similar to that given in [2]. We set

$$E_j \triangleq \left\{(a, b)/a(X)S^*(X) \equiv b(X) \bmod X^j\right\}$$

$$d_j \triangleq \min_{\substack{(a,b)\in E_j \\ a(0)=1}} \deg(a, b)$$

$$E_j^* \triangleq \left\{(a, b) \in E_j / \deg(a, b) = d_j, a(0) = 1\right\}.$$

We may restate [2, Lemma 1] as follows.

Lemma: If $(a, b) \in E_j^*, (a, b) \notin E_{j+1}$, then $d_{j+1} \geq j + 1 - d_j$.

Proof: Since $(a, b) \notin E_{j+1}$ we have $a(X)S^*(X) \equiv b(X) + \Delta X^j \bmod X^{j+1}$ for some nonzero Δ; for any pair (e, f) in E_{j+1}^*, we have

$$af - be = e\Delta X^j + QX^{j+1} \quad \text{for some} \quad Q.$$

Since $e(0) = 1$, $\deg(af - be) > j$, hence the lemma.

It may be shown by straightforward induction on j that

- the coefficient of X^j in $(\sigma_j'S^* - \omega_j')$ is Δ;
- $(\sigma_j, \omega_j) \in E_j, (\sigma_j', \omega_j') \in E_j$;
- $\deg(\sigma_j, \omega_j) + \deg(\sigma_j', \omega_j') = j + 1$.

It remains to be shown that $\deg(\sigma_j, \omega_j)$ is indeed minimum. This is done by induction on j. The induction hypothesis and the lemma proves that this is true whenever step 5) is executed. However, this is also true if steps 3) or 4) are executed since $\deg(\sigma_{j+1}, \omega_{j+1}) = \deg(\sigma_j, \omega_j)$ and d_j is nondecreasing with j.

References

[1] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.

[2] J. L. Massey, "Shift register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 122–127, Jan. 1969.

[3] L. R. Welch and R. A. Scholtz, "Continued fractions and Berlekamp's algorithm," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 18–27, Jan. 1979.

[4] R. J. McEliece, "The theory of information and coding," in *Encyclopedia of Mathematics and Its Applications*. Reading, MA: Addison-Welsey, 1977.

[5] D. M. Mandelbaum, "A method for decoding of generalized Goppa codes," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 137–140, Jan. 1977.

[6] U. Cheng, "On the continued fraction and Berlekamp's algorithm," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 541–544, May 1984.

[7] Y. Sugiyama *et al.*, "A method for solving key equation for decoding Goppa codes," *Inform. Contr.*, vol. 27, pp. 87–99, Jan. 1975.