I. INTRODUCTION

Let $F^n$ be the $n$-dimensional vector space over $GF(2)$. The Hamming weight $wt(v)$ of a vector $v \in F^n$ is the number of its nonzero coordinates. The Hamming distance between two vectors $v, u \in F^n$ is $d(v, u) = wt(v + u)$.

A binary linear code $C$ of length $n$ is a linear subspace of $F^n$. The elements of $C$ are called codewords. We will denote by $d$ the minimum distance between any two distinct codewords. We call $C$ an $e$-error-correcting code if $e \leq (d - 1)/2$. A code with only one codeword is said to be trivial. A code with only all-one codeword and all-zero codeword is called a repetition code. In this correspondence we will assume that all codes are binary, linear, and nontrivial.

Given any vector $v \in F^n$, its distance to the code $C$ is

$$d(v, C) = \min_{c \in C} \{d(v, c)\}$$

and the covering radius of the code $C$ is

$$\rho = \max_{v \in F^n} \{d(v, C)\}.$$ 

Given two sets $X, Y \subseteq F^n$, we also define the sum $X + Y$ as the set of all vectors that can be expressed as the sum of a vector in $X$ and a vector in $Y$. We write $X + x$ instead of $X + \{x\}$.

A binary linear code $C$ of length $n$ is called completely regular if for all $v \in F^n$ and $v \neq 0$, the number of codewords at distance $p$ depends only on $p$ and $d(v, C)$.

An automorphism of $C$ is a coordinate permutation fixing $C$. The set of all automorphisms of $C$ is the full automorphism group of $C$ and is denoted by $Aut(C)$. $Aut(C)$ acts in the following way on the quotient set $F^n/C$: for any $\alpha \in Aut(C)$ and $x \in C$, $\alpha(x + C) = \alpha(x) + C$.

We call $C$ a completely transitive code if $Aut(C)$ acting on $F^n/C$ has exactly $\rho + 1$ orbits. Since two cosets in the same orbit have identical weight distribution, we have that a completely transitive code is always completely regular. For a more detailed proof see [10].

Let $C$ be a binary linear $e$-error-correcting code. It has been conjectured for a long time that if $C$ is a completely regular code and $|C| > 2$, then $e \leq 3$. In fact, this conjecture has been stated for nonbinary and nonlinear codes. Moreover, in [8] it is conjectured that the only completely regular code $C$ with $|C| > 2$ and $d \geq 8$ is the well-known extended binary Golay code. In this correspondence we do not attempt to prove this conjecture but we study the subclass of completely transitive codes, which is strictly contained in the class of completely regular codes (see [10]). Our main result is Theorem 5 which states the nonexistence of completely transitive codes with more than two codewords and error-correcting capability greater than four.

II. PRELIMINARIES ON PERMUTATION GROUPS

Let $G$ be a finite permutation group acting on an $n$-set $X$. We say that $G$ has degree $n$. $G$ is called $t$-transitive ($0 < t \leq n$) if for any pair of ordered $t$-tuples of distinct elements of $X \{x_1, \ldots, x_t\}$ and $\{y_1, \ldots, y_t\}$ there exists $g \in G$ such that $g(x_i) = y_i$ ($1 \leq i \leq t$). $G$ is transitive if it is 1-transitive. $G$ is called $t$-homogeneous ($0 < t \leq n$) if for any pair of unordered $t$-sets of distinct elements of $X \{x_1, \ldots, x_t\}$ and $\{y_1, \ldots, y_t\}$ there exists $g \in G$ such that

$$g(\{x_1, \ldots, x_t\}) = \{y_1, \ldots, y_t\}.$$ 

Of course, if $G$ is $t$-transitive, it is also $(t-1)$-transitive and $t$-homogeneous. We also remark that if $G$ is $t$-homogeneous it is $(n-t)$-homogeneous.

The relationship between transitivity and homogeneity is very strong as we can see by means of the following result of Livingstone and Wagner [7].

**Theorem 1:** If $G$ is $t$-homogeneous, where $2 \leq t \leq n/2$, then $G$ is $(t-1)$-transitive, and for $t \geq 5$ even $t$-transitive.

**Proof:** See [7] or [1, Theorem 2.19 p. 251].

The following result shows the nonexistence of nontrivial highly transitive groups.

**Theorem 2:** Let $G$ be a finite $t$-transitive group of degree $n$.

i) If $t > 5$, then $G$ is the symmetric or the alternating group of degree $n$.

ii) If $t = 5$ and $G$ is not the symmetric or the alternating group, then $G$ is one of the Mathieu groups $M_{12}$ or $M_{24}$ (of degree 12 or 24, respectively).

**Proof:** The reader may see [2], [3, p. 591] or [6, pp. 623–625].

III. THE EXISTENCE OF COMPLETELY TRANSITIVE CODES

The following two results may be found in [10, Propositions 7.2 and 7.3].

**Proposition 3:** Let $C$ be a binary linear code of length $n$ and covering radius $r \leq n/2$. If $Aut(C)$ is $\rho$-homogeneous, then $C$ is completely transitive.

**Proposition 4:** If an $e$-error-correcting code $C$ is a completely transitive code, then $Aut(C)$ is $e$-homogeneous on the coordinate positions.

The last proposition enables us to show that there is no completely transitive code with high error-correcting capability.

**Theorem 5:** If $C \subseteq F^n$ is an $e$-error-correcting completely transitive nontrivial code, then $e \leq 4$ or $C$ is a repetition code.

**Proof:** Assume that $C$ is not trivial. $Aut(C)$ must be $e$-homogeneous by Proposition 4. We have $|C| > 1$, then $d \leq n$ and $e \leq n/2$. Thus if $e \geq 5$, $C$ must also be $e$-transitive by Theorem 1.

i) If $e > 5$, then by Theorem 2, $Aut(C)$ is the symmetric or the alternating group of degree $n$. Suppose that $C$ has a codeword $x = (x_1, \ldots, x_n)$ which is neither the all-zero vector nor the all-one vector. Then, let $i, j$ and $k$ be distinct indices such that $x_i \neq x_k (i, j, k \in \{1, \ldots, n\})$. The permutation cycle $\pi = (i, j, k)$ is in the alternating and symmetric group, so...
Suppose that \( e = 5 \) and \( \text{Aut}(C) \) is not the symmetric or the alternating group of degree \( n \). Then by Theorem 2, \( \text{Aut}(C) \) is either \( M_{12} \) or \( M_{23} \). If \( \text{Aut}(C) = M_{12} \), then \( C \) should be a nontrivial linear code of length 12 and minimum distance \( d \geq 11 \). Clearly, the only possibility is the repetition code. Assume that \( \text{Aut}(C) = M_{23} \). Since \( C \) is a nontrivial completely regular code, then the minimum-weight codewords form an \( e\)-\((n, d, \lambda)\)-design on the set of coordinates (see [12]), or \( C \) is a repetition code. In any \( e\)-\((n, d, \lambda)\)-design with \( e \geq 4 \) and \( n \geq d + 2 \), the number of blocks is \( b \geq n(n - 1)/2 \) (this bound is given in [9]). Hence, we have at least \( 2^{4} \cdot 23^2 = 276 \) codewords and \( C \) has dimension \( k > 8 \). But the sphere packing bound says that

\[
2^k \leq \sum_{i=0}^{n/24} \binom{24}{i} \Rightarrow 2^k \leq 302
\]

hence \( k < 9 \), which contradicts the previous result. Thus \( C \) must be a repetition code. \( \square \)

Applying Proposition 3, it is easy to see that perfect and extended perfect Hamming codes are completely transitive with \( e = 1 \). Also, the perfect and the extended perfect binary Golay code are completely transitive with \( e = 3 \). These examples, and some other ones, are also mentioned in [10]. For the case \( e = 2 \), we can consider the truncated binary Golay code \( C \) (deleting any fixed coordinate form each codeword of the binary Golay code). Clearly, \( C \) has length \( 22 \), \( e = 2 \) and is invariant under the Mathieu group \( M_{22} \), which is 3-transitive; therefore, \( C \) is a completely transitive code by Proposition 3. This last example comes from [4].

Hence, for \( e = 1, 2, \text{ or } 3 \), there exist completely transitive codes. As we have mentioned above, for \( e > 3 \) it has been conjectured that there is no completely regular code containing more than two codewords, and hence there is no completely transitive code, with the exception of the trivial or the repetition codes. This has been proved for the case \( e = \rho \) (perfect codes) by Tietäväinen in 1973 (see [11]), and also for the case \( e + 1 = \rho \) (uniformly packed codes) by Van Tilborg in 1976 (see [5]). For \( \rho > e + 1 \) there is no proof of the conjecture.

We have shown the nonexistence of completely transitive codes for \( e > 4 \) with more than two codewords. However, the existence of such codes for \( e = 4 \) remains an open question.

Acknowledgment

The authors wish to thank P. Solé for his comments that were the starting point for the proof of the case \( e = 5 \) of Theorem 5. The authors also wish to thank M. Giudici and C. Praeger for helpful comments during the preparation of this correspondence and the anonymous referees who enlightened us about the example of the truncated Golay code.

References

the dual codes, and to find the weight distribution and associated 1-designs for the subclass $C_\ell(1, n)$.  

II. ABELIAN GROUP CHARACTER CODES  

Let $A$ be an additive Abelian group of exponent $m$ and order $N$, with 0 as the identity element. Let $K$ be a finite field with characteristic not dividing $N$ and which contains the $m$th roots of unity. Let $K^*$ denote the multiplicative group of nonzero elements of $K$ and let $M$ denote the multiplicative group of characters from $A$ to $K^*$. The group $M$ is isomorphic noncanonically to $A$ [3, Ch. VI]. In particular, we have $|M| = |A| = N$.  

The following lemma is a well-known result, known as the orthogonality relations in character theory [3, Ch. VI, Proposition 4].

**Lemma 1. Orthogonality Relations:** Let $A$ be a finite additive Abelian group of order $N$ and let $M$ be the group of characters of $A$. For characters $f, g$ in $M$ and elements $x, y$ in $A$, we have

$$
1) \sum_{x \in A} f(x)g(y) = \begin{cases} 
N, & \text{if } f = g^{-1} \\
0, & \text{if } f \neq g^{-1}
\end{cases}
$$

$$
2) \sum_{f \in M} f(x)f(y) = \begin{cases} 
N, & \text{if } x = -y \\
0, & \text{if } x \neq -y.
\end{cases}
$$

Let $M = \{f_0, f_1, \cdots, f_{N-1}\}$, where $f_0$ is the trivial character. For any subset $X$ of $A$, we define a linear code $C_X$ over $K$ as

$$C_X = \{\alpha_0, \alpha_1, \cdots, \alpha_{N-1}\} \in K^N:
\sum_{i=0}^{N-1} \alpha_i f_i(x) = 0 \text{ for all } x \in X\}.$$  

Let $X = \{x_0, x_1, \cdots, x_{N-1}\}$ be a subset of $A$ and let $X^c$ be the complement of $X$ in $A$, indexed such that $A = \{x_0, x_1, \cdots, x_{N-1}\}$.  

**Proposition 2:** Let $A$ and $X$ be as above. For $0 \leq i \leq N - 1$, let $v_i$ denote the vector $(f_0(x_0), f_1(x_1), \cdots, f_{N-1}(x_i))$. Then the set $\{v_0, v_1, \cdots, v_{N-1}\}$ is linearly independent. In particular $2 = \{f_{i-1}(x_{i-1})\}_{1 \leq i \leq 1 \leq N}$ has rank $t$ and is a parity-check matrix of $C_X$  

$$G = \{f_{j-1}(x_{j-1})\}_{1 \leq i \leq N-1, 1 \leq j \leq N}$$

has rank $N-t$ and is a generator matrix for $C_X$, so $C_X$ is an $[N, N-t]$ linear code over $K$. Moreover

$$[f_{j-1}(x_{j-1})]_{1 \leq i \leq N-1, 1 \leq j \leq N}$$

is a generator matrix for $C_X^c$ and $C_X \oplus C_X^c = K^N$.  

**Proof:** The linear independence of the set $\{v_0, v_1, \cdots, v_{N-1}\}$ follows from Lemma 1. The other conclusions of the proposition then follow from this linear independence.  

III. ELEMENTARY 2-GROUP CHARACTER CODES  

Hereafter we let $A$ be the elementary 2-Abelian group $(\mathbb{Z}/2\mathbb{Z})^n$, for which we prescribe a basis $\{e_1, \cdots, e_n\}$ of generators. Moreover, we denote the neutral element of $A$ by $e_0$. For the field $K$ we take a finite field $\text{GF}(q)$ for $q$ odd. Since $\{\pm 1\}$ is contained in $\text{GF}(q)$, the character group $M$ of $A$ is defined. Relative to the basis for $A$ we can define characters by $f_j(e_i) = (-1)^{i+j-1}$, where

$$j = \sum_{k=0}^{n-1} j_k 2^k, \quad \text{for } 0 \leq j < 2^n.$$  

One easily verifies that this gives an indexing $M = \{f_0, \cdots, f_{2^n-1}\}$ on the group of characters from $A$ to $\text{GF}(q)^*$, where $\text{GF}(q)^*$ is defined to be $\text{GF}(q) \setminus \{0\}$.

**Theorem 3:** For any subset $X$ of $A$, let $X^c = A \setminus X$. Then the dual code $C_X^c$ equals $C_{X^c}$.  

**Proof:** This follows from the orthogonality relations of Lemma 1, the description of a generator matrix for $C_{X^c}$ of Proposition 2, and from the fact that every element of $A$ is its own inverse.  

The classes of codes over $\text{GF}(q)$ described in this section have many subclasses of codes. Different choices of the set $X$ give different subsets of codes over $\text{GF}(q)$. We now describe a subclass of codes over $\text{GF}(q)$ which have the same parameters as the binary Reed–Muller codes.

**Definitions:** The Hamming weight $||e||$ of an element $e = a_0 e_1 + \cdots + a_n e_n$ of $A$ is defined to be the number of nonzero $a_k$. For $-1 \leq r \leq n$, let

$$X(r, n) = \{a \in A: ||a|| > r\}$$

and let $C_q(r, n)$ denote the code $C_{X(r, n)}$ over $\text{GF}(q)$. For a word $e = (a_0, \cdots, a_{2^n-1})$ in $K^{2^n}$, let the support of $e$ be defined as

$$\text{Supp}(e) = \{i: 0 \leq i < 2^n, \text{ and } e_i \neq 0\}$$

and let its weight be defined as $|\text{Supp}(e)|$. By convention we define the minimum distance of the zero code to be $\infty$, which we represent by any integer larger than the block length of the code.  

We use $[\cdot, \cdot]$ to denote concatenation of codewords, and for two sets $U$ and $V$ of codewords define $U[V]$ by the set $\{u \oplus v: u \in U, v \in V\}.

**Lemma 4:** The dimension of the code $C_{q}(r, n)$ is

$$s_n(r) = \sum_{j=0}^{r} \binom{n}{j}.$$  

**Proof:** This follows from Proposition 2 and the definition of $X(r, n)$.

**Lemma 5:** The code $C_q(r + 1, n + 1)$ decomposes as the direct sum

$$C_q(r + 1, n + 1) = (C_q(r, n + 1)\{0\}) \oplus \{0\} \oplus \{0\}$$

where $\{0\}$ is the zero code in $K^{2^n}$.  

**Proof:** For vector subspaces $U$ and $V$ of $K^{2^n}$ it is clear from the definitions that $U[V] = U[\{0\} \oplus \{0\}]$. Moreover, by the combinatorial equality

$$s_{n+1}(r + 1) = s_{n+1}(r + 1) + 1$$

Lemma 4 and the linear independence of $C_q(r + 1, n + 1)\{0\}$ and $\{0\}$ imply the result provided that both are contained in $C_q(r + 1, n + 1)$.  

Let $A_n$ be the elementary 2-Abelian group generated by $\{e_1, \cdots, e_n\}$, identified as a subgroup of $A_{n+1} = A_n \oplus \mathbb{Z}/2\mathbb{Z} e_{n+1}$. The character $f_j$ on $A_n$ identifies with the character $f_j$ on $A_{n+1}$ by setting $f_j(e_{n+1}) = 1$ for all $0 \leq j \leq 2^n - 1$. Then by definition $f_{j+2^n} = f_j f_{2^n}$, where

$$2^n \sum_{j=0}^{2^n-1} e_j f_j(a) = 0$$

The words in $C_q(r + 1, n + 1)\{0\}$ are of the form $e\cdot e$ such that $e = (a_0, \cdots, a_{2^n-1})$ and

$$0 \leq e \cdot e < 2^{n+1}\cdot 2^n = 2^{2n+1}.$$
for all $a$ in $X(r + 1, n)$. One readily verifies that $C_\phi(r + 1, n) || \{0\} = C_{X_1}$, where

$$X_1 = X(r + 1, n) \cup (A_n + e_{n+1}).$$

Similarly, $\{0\} || C_\phi(r, n) \subseteq C_{X_2}$, where

$$X_2 = X(r, n) \cup (X(r, n) + e_{n+1}).$$

Since $X_1$ and $X_2$ contain $X(r + 1, n + 1)$, both $C_{X_1}$ and $C_{X_2}$ are contained in $C_{\phi(r + 1, n + 1)}$ and the result holds. \hfill $\Box$

**Theorem 6:** $C_\phi(r, n)$ is a $[2^n, s_\phi(r), 2^{n-r}]$ code over $GF(q)$.

**Proof:** It only remains to prove the correctness of the minimum distance. Since $X(0, n) = A \setminus \{e_0\}$, a generator matrix for $C_\phi(0, n)$ is

$$[f_0(e_0) f_1(e_0) \cdots f_{n-1}(e_0)] = [1 \cdots 1].$$

Therefore, $C_\phi(0, n)$ has minimum weight $2^n$.

At the other extreme, $X(n, n)$ is empty, so $C_\phi(n, n) = K_n^{2^n}$ and $C_\phi(n, n)$ has minimum weight $1$. In particular, it follows that the minimum weight is correct for $n = 1$ and $0 \leq r \leq 1$.

Suppose now that Theorem 6 gives the minimum distance for some $n \geq 1$ and all $0 \leq r \leq n$. By Lemma 5, a word $e$ in $C_{\phi(r + 1, n + 1)}$ is of the form

$$c = u + v$$

where $u$ and $v$ are codewords in $C_{\phi(r + 1, n)}$ and $C_{\phi(r, n)}$, respectively. Then

$$\text{wt}(c) \geq 2\text{wt}(u) + \text{wt}(v) - 2|\text{Supp}(u) \cap \text{Supp}(v)| \geq \text{wt}(v) \geq 2^{n-r}.$$ 

Conversely, $\{0\} || C_{\phi(r, n)}$ is a subcode of $C_{\phi(r + 1, n + 1)}$ with minimum distance $2^{n-r}$, so the minimum distance of $C_{\phi(r + 1, n + 1)}$ is $2^{n-r}$. \hfill $\Box$

**Theorem 7:** The minimum nonzero weight codewords generate $C_{\phi}(r, n)$.

**Proof:** Let $M_{\phi}(r, n)$ denote the set of minimum nonzero weight codewords of $C_{\phi}(r, n)$. The result is clear for $n = 1$ and all $r$, and likewise for $r = -1$ and all $n$. Moreover, $C_{\phi}(r + 1, n + 1)$ is generated by

$$M_{\phi}(r + 1, n) || \{0\} \cup \{0\} || M_{\phi}(r, n)$$

by the direct sum of Lemma 5. This set is contained in $M_{\phi}(r + 1, n + 1)$, from which the result holds by induction. \hfill $\Box$

Let $G$ be defined as the subgroup in the group of linear automorphisms of $K_n^{2^n}$ generated by permutations of coordinates and by multiplications of coordinates by elements of $K^n$. Two codes $C$ and $C'$ are called equivalent if and only if $C' = \phi(C)$ for some $\phi \in G$.

**Theorem 8:** The dual code $C_{\phi}(r, n)^{\perp}$ is equivalent to $C_{\phi}(n - r - 1, n)$.

**Proof:** The theorem clearly holds for $r = n$, so we assume henceforth that $0 \leq r < n$. Let $\mu = e_1 + \cdots + e_n$. Then the equality

$$X(r, n)^{\perp} = \mu + X(n - r - 1, n)$$

follows immediately from the definitions. By Theorem 3 and the definitions of the codes, one easily verifies that the map

$$K_n^{2^n} \rightarrow K_n^{2^n}$$

$$(c_0, \cdots, c_{2^n-1}) \mapsto (f_0(\mu)c_0, \cdots, f_{2^n-1}(\mu)c_{2^n-1})$$

is an automorphism of $K_n^{2^n}$ of order two, which induces an equivalence of $C_{\phi}(r, n)^{\perp}$ and $C_{\phi}(n - r - 1, n)$ and vice versa. \hfill $\Box$

**Corollary 9:** $C_{\phi}(r, n)^{\perp}$ is a $[2^n, s_\phi(n - r - 1), 2^{n-r}]$ code which is generated by its minimum nonzero weight codewords.

**Proof:** The conclusions follow from Theorems 6–8. \hfill $\Box$

**Remark:** The Reed–Muller codes $R(r, n)$ are well-known to have the same parameters $[2^n, s_\phi(r), 2^{n-r}]$ as the codes $C_{\phi}(r, n)$, so the latter can be viewed as analogs over $GF(q)$ of the corresponding binary Reed–Muller codes. Moreover, we have

$$R(n, n) = R(n - r - 1, n)$$

of which Theorem 8 is its analog. While strict equality of $C_{\phi}(r, n)^{\perp}$ and $C_{\phi}(n - r - 1, n)$ does not hold, the proof shows that the equivalence is a simple coordinate twist. Stated formally, this says that $C_{\phi}(n - r - 1, n)$ is the dual of $C_{\phi}(r, n)$ with respect to the twisted inner product

$$\langle u, v \rangle = \sum_{j=0}^{2^n-1} f_j(\mu) u_j v_j$$

on vectors $u = (u_0, \cdots, u_{2^n-1})$ and $v = (v_0, \cdots, v_{2^n-1})$ in $K_n^{2^n}$.

**Example 1:** Consider $n = 3$. Then $C_{\phi}(1, 3)$ is a $[8, 4, 4]$ ternary code with generator matrix

$$\begin{bmatrix}
1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 \\
1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \\
1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\
1 & -1 & -1 & -1 & -1 & 1 & -1 & -1
\end{bmatrix}$$

and weight enumerator polynomial $1 + 24x^2 + 16x^5 + 32x^9 + 8x^{16}$. In Theorem 17, we show that the support of the codewords of minimum nonzero weight form a $1-(8, 4, 6)$ design. \hfill $\Box$

**Example 2:** Consider $n = 4$. Then $C_{\phi}(2, 4)$ is a $[16, 5, 8]$ ternary code with generator matrix as seen in the matrix at the bottom of this page and weight enumerator polynomial

$$1 + 40x^8 + 80x_{10} + 32x_{11} + 80x_{12} + 10x_{16}.$$
Its dual code $C_3(2, 4)^*$ is a $[16, 11, 4]$ code with weight enumerator polynomial
\[
1 + 200x^4 + 352x^5 + 2544x^6 + 5600x^7 + 13740x^8
+ 23840x^9 + 34272x^{10} + 36480x^{11} + 30840x^{12}
+ 18400x^{13} + 8720x^{14} + 1824x^{15} + 334x^{16}.
\]

IV. THE WEIGHT DISTRIBUTION IN $C_3(1,n)$

Let the characters $g_i$ be defined previously for $0 \leq j < 2^n$ and define vectors $\mathbf{u}_i = (g_i(e_1), \ldots, g_i(e_n))$ for $0 \leq i < n$. In particular, $\mathbf{u}_0$ is the vector $(1, \ldots, 1)$ and $\{\mathbf{u}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is a basis for the code $C_3(1,n)$.

Let $V = \text{GF}(q)^n$ and for $a$ in $V$, let $|a|$ denote its Hamming weight. For $a = (a_1, \ldots, a_n)$ in $V$, set $a \cdot (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ equal to the dot product $\sum_{i=1}^{n} a_i \mathbf{v}_i$. The collection of vectors of the form $\sum_{i=1}^{n} a_i \mathbf{v}_i$ constitute a subcode $C_3(1,n)$ of codimension 1. We first describe the weight distribution in $C_3(1,n)$ and then treat the nontrivial cosets $a_0 \mathbf{u}_0 + C_3(1,n)$.

We begin with a series of lemmas which reduce the analysis of the weight distribution to the weights of codewords in subsets of $C_3(1,n)$.

Lemma 10: Let $a$ have Hamming weight $m$ in $V$. Then the vector $\mathbf{v} = a \cdot (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ has Hamming weight $2^n - 2^{n-m}[W]$, where
\[
W = \left\{ \mathbf{b} = (b_1, \ldots, b_n) \in V : b_i = \pm a_i \text{ for all } i \text{ and } \sum_{i=1}^{n} b_i = 0 \right\}.
\]

Proof: Clearly $a \cdot (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ has weight $2^n$ less than the number of $j$ for which
\[
\sum_{i=1}^{n} a_i g_j(e_i) = 0.
\]

For any such $j$, we associate the vector $\mathbf{b} = (a_1 g_j(e_1), \ldots, a_n g_j(e_n))$ in $W$. But for any $\mathbf{b}$ in $W$ there are $2^{n-m}$ indices $j$ for which (1) holds and has associated vector $\mathbf{b}$. Namely, if we let $j = \sum_{i=0}^{n-1} j_i 2^i$ be another such value, then $k = \sum_{i=0}^{n-1} k_i 2^i$ is another if and only if $k_i = j_i$ whenever $a_i \neq 0$.

Let $s = (q-1)/2$ and let $\{a_1, \ldots, a_{s-1}\}$ be an indexing of the elements of $\text{GF}(q)^s$ such that $a_{i+s} = -a_i$. For an element $a = (a_1, a_2, \ldots, a_m)$ of $V$, define for each $i$
\[
n_i(a) = \{|k : 1 \leq k \leq n \text{ and } a_k = \pm a_i\}.
\]

Lemma 11: For $a$ in $V$, let $n_1(a) = n_1(\mathbf{a}), \ldots, n_s(a) = n_s(\mathbf{a})$ be the associated multiplicities. Then the weight of the vector $\mathbf{v} = a \cdot (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ is
\[
2^n - 2^{n-m} \sum_{\substack{n_1 \leq n_2 \leq \cdots \leq n_s \\
\sum_{i=1}^{s} n_i = m \mod 2}} \prod_{i=1}^{s} \binom{n_i}{n_i/2}.
\]

In particular, the weight of $\mathbf{v}$ depends only on the multiplicities $n_1, \ldots, n_s$, and not on the ordering of the coordinates of $\mathbf{a}$.

Proof: By Lemma 10, it suffices to determine $[W]$. Let $\mathbf{b}$ be in $W$, and for each $i$ let $a_i$ be the number of occurrences of $a_i$ in the coordinates of $\mathbf{b}$. Then the number of occurrences of $-a_i$ in this vector is $n_i - a_i$. It follows that
\[
\sum_{i=1}^{n} b_i = \sum_{i=1}^{s} (u_i a_i - (n_i - u_i) a_i) = \sum_{i=1}^{s} (2u_i - n_i) a_i
\]
and we set $m_i = 2u_i - n_i$. Since there are $\prod_{i=1}^{s} \binom{m_i}{n_i}$ vectors $\mathbf{b}$ in $W$ with associated multiplicities $n_1, \ldots, n_s$, we obtain
\[
[W] = \sum_{0 \leq u_i \leq m_i} \prod_{i=1}^{s} \binom{n_i}{u_i}
\]
from which the lemma follows.

The following lemma gives the weight of vectors in the nontrivial coset of $C_3(1,n)$ in $C_3(1,n)$, and is proved similarly.

Lemma 12: For $a$ in $V$, let $n_1 = n_1(\mathbf{a}), \ldots, n_s = n_s(\mathbf{a})$ be the associated multiplicities. Then the weight of a vector $\mathbf{v} = a_0 \mathbf{u}_0 + a \cdot (\mathbf{v}_1, \ldots, \mathbf{v}_n)$, with $a_0 \neq 0$, is
\[
2^n - 2^{n-m} \sum_{\substack{n_1 \leq n_2 \leq \cdots \leq n_s \\
\sum_{i=1}^{s} n_i = m \mod 2}} \prod_{i=1}^{s} \binom{n_i}{n_i/2}.
\]

In particular, the weight of $\mathbf{v}$ depends only on the multiplicities $n_1, \ldots, n_s$, and not on the ordering of the coordinates of $\mathbf{a}$.

When $q = 3$ the product in Lemmas 11 and 12 consists of one term. In this case, we obtain a precise formula for the weight distribution in $C_3(1,n)$ and $C_2(1,n)$. First we prove a result concerning sums of binomial coefficients in progressions.

Lemma 13: Let $m$ be a positive integer. Define
\[
a = \sum_{0 \leq j \leq m \mod 3} \binom{m}{j} \quad b = \sum_{0 \leq j \leq m \mod 3} \binom{m}{j} \quad c = \sum_{0 \leq j \leq m \mod 3} \binom{m}{j}.
\]

1) If $m = 3k$, then
\[
a = \frac{2^m + (-1)^k 2}{3} \quad b = \frac{2^m}{3} \quad c = \frac{2^m - (-1)^k}{3}.
\]

2) If $m = 3k + 1$, then
\[
a = b = \frac{2^m + (-1)^k 2}{3} \quad c = \frac{2^m - (-1)^k}{3}.
\]

3) If $m = 3k + 2$, then
\[
b = \frac{2^m + (-1)^k 2}{3} \quad c = \frac{2^m - (-1)^k}{3}.
\]

Proof: Let $m$ be of the form $3k + i$, with $0 \leq i \leq 2$, and let $\epsilon$ be a primitive third root of unity. We note that $\epsilon$ has trace $-1$ and that $\epsilon + 1$ is a cube root of $-1$ whose square is $\epsilon$. Expanding the binomial expression we obtain
\[
(\epsilon + 1)^m = \sum_{j=0}^{m} \binom{m}{j} \epsilon^{m-j}.
\]

In the three cases $i = 0, 1, 2$, respectively, we obtain
\[
(-1)^k \epsilon \pm a + c \epsilon + b \epsilon^2 \quad \text{for } i = 1,
\]
\[
(-1)^k (\epsilon + 1) \pm b + a \epsilon + c \epsilon^2 \quad \text{for } i = 0,
\]
\[
(-1)^k \epsilon = c + b \epsilon + a \epsilon^2 \quad \text{for } i = 2.
\]
ary Reed–Solomon codes agree with that of the binary Reed–Muller code.

It remains an open problem for the class of codes $C_s(r, n)$ described here to determine the weight distribution for $r > 1$. The minimum nonzero weight codewords in $C_s(1, n)$ are a $(2^n, 2^{n-m}, n(n+1)/2)$ design. The set of supports of the minimum nonzero weight codewords in $C_3(1, n)$ is a $(2^n, 2^{n-m}, (n(n+1)/2)+1)$ design. We have described a class $C_s(r, n)$ of group character codes over $\mathbb{GF}(q)$ and determined their dimensions and minimum weights. For each $n$ and $r$, the length, dimension, and minimum weight of $C_s(r, n)$ agrees with that of the binary Reed–Muller code $R(r, n)$. For the codes $C_s(1, n)$ we have explicitly determined the weight distribution and proved that the minimum nonzero weight codewords give 1-designs.

### Acknowledgment

The authors wish to thank I. F. Blake for his encouragement and the referees for their constructive comments and suggestions that considerably improved this correspondence.

### References


### New Constructions of Superimposed Codes

Arkadii G. D’yachkov, Anthony J. Macula, Jr, and Vyacheslav V. Rykov

Abstract—Kautz-Singleton (1964) [1] suggested a class of binary superimposed codes which are based on the q-ary Reed–Solomon codes (RS codes) [2]. Applying a concatenation of the binary constant-weight error-correcting codes [2] and the shortened RS codes, we obtain new constructions of superimposed codes. Tables of their parameters are given. From the tables it follows that the rate of obtained codes exceeds the corresponding random coding bound [3].

Index Terms—Clone-library screening, concatenated codes, Reed-Solomon codes, superimposed codes.

### I. SUPERIMPOSED CODES

#### A. Notations and Definitions

Let $1 \leq s < t$, $N > 1$ be integers and $X = \|X_i(u)\|$, $i = 1, 2, \ldots, N$, $u = 1, 2, \ldots, t$, be a binary $(N \times t)$-matrix (code) of size $t$ and length $N$ with columns (codewords) $x(1)$, $x(2)$, $\ldots$, $x(t)$, where $x(u) = (x_1(u), x_2(u), \ldots, x_N(u))$. Let $\text{min}$ denote the equation by definition. For code $X$, let $w$ and $\lambda$ be defined by

$$w = \min_{u} \sum_{i=1}^{N} x_i(u), \quad \lambda = \max_{u, v} \sum_{i=1}^{N} x_i(u) x_i(v).$$

$w$ is the minimal weight of codewords and $\lambda$ is the maximal correlation of codewords.

We say that the binary column $x$ covers the binary column $y$ if the Boolean sum $x \lor y = x$. The code $X$ is called [1], [4]–[6], a super-

### V. CONCLUDING REMARKS

We have described a class $C_s(r, n)$ of group character codes over $\mathbb{GF}(q)$ and determined their dimensions and minimum weights. For each $n$ and $r$, the length, dimension, and minimum weight of $C_s(r, n)$ agrees with that of the binary Reed–Muller code $R(r, n)$. For the codes $C_s(1, n)$ we have explicitly determined the weight distribution and proved that the minimum nonzero weight codewords give 1-designs.
imposed \((s, N, t)\)-code, or \(s\)-disjunct code if the Boolean sum of any \(s\)-subset of columns of \(X\) covers only those columns of \(X\) which are the terms of the given Boolean sum.

One can consider an arbitrary fixed code \(X\) as the incidence matrix of a \(t\)-family of subsets of the \(N\)-set. In this interpretation, the \(s\)-disjunct code \(X\) one-to-one corresponds to the family in which no set is covered by the union of \(s\) others.

Superimposed codes were introduced by Kautz and Singleton [1] who worked out the constructive methods. D’yachkov and Rykov [3, 4] and Erdos, Frankl, and Furedi [7] obtained the best upper and lower bounds on the rate of superimposed codes. We remind the reader of the asymptotic form of these bounds in Section I–B. Some bounds on the rate were also derived in [8]–[10].

D’yachkov, Macula, and Rykov [11] investigated the development of constructions for superimposed codes (nonadaptive pooling designs) intended for the clone library screening problem. (See Knill, Bruno, and Torney [12].) In Section I–C, we give a brief introduction to the problem. The detailed discussions are given in [12] and [13].

In Sections II and III, we study the most important class of superimposed codes which are based on the \(q\)-ary Reed–Solomon codes (RS codes) [2]. The given class was invented by Kautz and Singleton [1]. We introduce some generalizations of the Kautz–Singleton codes and identify the parameters of the best known superimposed codes.

### B. Bounds on the Rate of Superimposed Codes

Let \( t(s, N) \) be the maximal possible size of \( s\)-disjunct code. For an arbitrary fixed integer \( s = 2, 3, \ldots \), define the function

\[
R(s) = \lim_{N \to \infty} \frac{\log_2 t(s, N)}{N}
\]

which can be interpreted as a rate of the optimal \( s\)-disjunct code. The best lower bound \( R(s) \leq \bar{R}(s) \) on the rate \( R(s) \) had been proved [3] with the help of the random coding method for the fixed-composition code ensemble. The best upper bound \( \bar{R}(s) \geq R(s) \) on the rate \( \bar{R}(s) \), called the recurrent bound, was obtained in [4]. The important ideas of the upper bounding method were worked out in [7]. In [6], one can find a detailed summary of the existing bounds. Here we show only the descriptive asymptotic \((s \to \infty)\) form of the best bounds

\[
R(s) \sim \frac{1}{s^2 \log_2 e} \approx 0.693147 \frac{s^2}{s^2} = \frac{\log_2 s}{s^2},
\]

\[
\bar{R}(s) \sim \frac{2 \log_2 s}{s^2}.
\]

In Table I, we give the numerical values of \( R(s) \) and \( \bar{R}(s) \), when \( s = 2, 3, \ldots, 8 \).

### C. Application to DNA Library Screening

To understand what a DNA library is, think of several copies of an identical but incredible long word (of length \( \sim 10^6 \), e.g., a chromosome) from letters of the quaternary alphabet \{A, C, G, T\}. Each copy of the word has been cut in thousands of contiguous pieces (of length \( \sim 10^4 \), e.g., chromosome fragments). Take those pieces and copy their letter strings onto their own separate small piece of paper. The thousands of little pieces of paper (i.e., clones) that result essentially constitute a DNA library. In other words, each clone represents some contiguous subpiece of a contiguous superpiece of DNA. The DNA library, or the clone library consists of thousands separate clones.

A unique and contiguous subsuppiece of DNA (of length \( \sim 10^3 \)) is called a sequenced tagged site (STS). For a fixed STS, a clone is called positive (negative) for that STS if it contains (does not contain) that given STS.

Example: Let the following \( s = 4 \) copies of the DNA superpiece be given and \( \{C_1, C_2, C_3, C_4, C_5\} \) be the library of five clones.

\[
\begin{array}{c}
\text{AAAA} \\
\text{GCGTCT} \\
\text{TAA} \\
\text{CCGATAGGCAAATTG} \\
\end{array}
\]

\[
\begin{array}{c}
\text{AAAA} \\
\text{GCGTCT} \\
\text{TAA} \\
\text{CCGATAGGCAAATTG} \\
\end{array}
\]

\[
\begin{array}{c}
\text{AAAA} \\
\text{GCGTCT} \\
\text{TAA} \\
\text{CCGATAGGCAAATTG} \\
\end{array}
\]

\[
\begin{array}{c}
\text{AAAA} \\
\text{GCGTCT} \\
\text{TAA} \\
\text{CCGATAGGCAAATTG} \\
\end{array}
\]

\[
\begin{array}{c}
\text{AAAA} \\
\text{GCGTCT} \\
\text{TAA} \\
\text{CCGATAGGCAAATTG} \\
\end{array}
\]

Clones \( \{C_1, C_3\} \) could be taken from the same copy of the DNA superpiece. Clones \( \{C_1, C_3\} \) are taken from the different copies. Let \( \text{STS}_1 = \text{AAAA} \) and \( \text{STS}_2 = \text{TAA} \). Then \( C_1 \) is positive for \( \text{AAAA} \) and \( C_1, C_2 \) and \( C_1 \) are positive for \( \text{TAA} \). Note that \( C_1 \) is positive for both \( \text{AAAA} \) and \( \text{TAA} \) and \( C_3, C_5 \) are negative for both \( \text{AAAA} \) and \( \text{TAA} \).

A pool is a subset of clones. Each pool is tested as a group by exposing that entire group to a chemical probe (e.g., polymerase chain reaction) which can detect a given STS. A pool is called positive for the STS if the probe indicates that some member of that group contains the given STS. In other words, if the tests are error-free, then a pool is positive for an STS if that pool contains at least one clone that contains the given STS.

Let \( 1 \leq s < t, N > 1 \) be integers. Mathematically, clone-library screening for positive clones is modeled by searching a \( t \)-set of objects (clone-library) for a particular \( s \)-subset, \( p \leq s \), called a subset of positive clones. A nonadaptive pooling design is a series of \( N \) a priori group tests that can often be carried out simultaneously. Every parallel pooling design is nonadaptive. The pooling design can be described by a binary \( N \times t \)-matrix \( X = [x_{i,j}] \). The elements \( x_{i,j} = 1 \) if the \( j \)th clone is in the \( i \)th pool and \( x_{i,j} = 0 \), otherwise. A pool outcome (result of the group test) is said to be positive if one of the pool’s clones is positive, negative otherwise. Using this binary \( N \)-sequence of outcomes, an investigator has to identify the \( p \)-subset, \( p \leq s \), of positive clones.

Let \( p \leq s \) be the number of positive clones in a clone-library of size \( t \). To identify an unknown \( p \)-subset of positive clones, we apply the pooling design \( X \) which is the superimposed \((s, N, t)\)-code. Obviously, the binary \( N \)-sequence \( y \) of the pool outcomes is the Boolean sum of the unknown \( p \)-subset of columns of \( X \). The definition of \((s, N, t)\)-code means that the unknown \( p \)-subset is represented by all columns which are below \( y \). Thus we need to carry out \( t \) successive comparisons of the Boolean sum \( y \) with codewords of \( X \). Hence, the identification complexity of \((s, N, t)\)-code does not exceed \( t \).

### II. Generalized Kautz–Singleton Codes

Let \( P \) be the set of all primes or prime powers \( \geq 2 \), i.e.,

\[
P := \{2, 3, 4, 5, 7, 8, 9, 11, 13, 16, 17, 19, 23, 25, 27, 29, 31, 32, 37, \ldots \}.
\]

Let \( q_0 \in P \) and \( 2 \leq k \leq q_0 + 1 \) be fixed integers for which there exists the \( q_0 \)-ary Reed–Solomon code (RS code) \( B \) of size \( q_0^k \), length \( (q_0 + 1) \), and distance \( d = q_0 - k + 2 = q_0 + 1 - (k - 1) \). We will identify the code \( B \) with a \((q_0 + 1) \times q_0^k\)-matrix whose columns
From (1) it follows that for binary code $X$, the maximal correlation of codewords is $\lambda = \lambda_0 = k - r - 1$.

Let $X$ be a binary code with parameters $w$ and $\lambda$. Kautz and Singleton [1] suggested the following evident sufficient condition of the $s$-disjunct property:

$$s \lambda \leq w - 1. \quad (2)$$

Hence, by virtue of (1) and (2), code $X$ is the $s$-disjunct code if

$$s(k - r - 1) = s\lambda_0 \leq w - 1 = q_0 - r. \quad (3)$$

For the particular case $r = 0$, this construction of $s$-disjunct codes was suggested in [1].

Denote by $[b]$ the least integer $\geq b$. Let $m \geq 1$ and $2 \leq s < 2^m$ be arbitrary fixed integers. Define

$$\mathcal{P}(m, s) \triangleq \left\{ q : q \in \mathbb{P}, s \left( \frac{m}{\log_2 q} \right) - 1 \leq q \right\}. \quad (4)$$

Consider a binary code $X$ identified by parameters $0 \leq r \leq k - 1 \leq q_0$, $q_0 \in \mathcal{P}(m, s)$. The sufficient condition (3) for $s$-disjunct property
TABLE II

PARAMETERS FOR 2- AND 3-DISJUNCT CODES BASED ON \((N, D, w)\)-CODES

<table>
<thead>
<tr>
<th>Parameters of ((2, N, t))-codes</th>
<th>Parameters of ((3, N, t))-codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N)</td>
<td>(D)</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>17</td>
<td>6</td>
</tr>
<tr>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>27</td>
<td>8</td>
</tr>
<tr>
<td>29</td>
<td>8</td>
</tr>
<tr>
<td>31</td>
<td>8</td>
</tr>
<tr>
<td>33</td>
<td>8</td>
</tr>
<tr>
<td>35</td>
<td>8</td>
</tr>
<tr>
<td>37</td>
<td>8</td>
</tr>
<tr>
<td>39</td>
<td>8</td>
</tr>
<tr>
<td>41</td>
<td>8</td>
</tr>
<tr>
<td>43</td>
<td>8</td>
</tr>
<tr>
<td>45</td>
<td>?</td>
</tr>
<tr>
<td>46</td>
<td>8</td>
</tr>
<tr>
<td>48</td>
<td>?</td>
</tr>
</tbody>
</table>

of \(X\) could be written in the form \(k - r \leq (q_0 + 1 - k/s - 1) + 1\). 
Hence, if

\[
\left[ \frac{m}{\log_2 q_0} \right] \leq k - r \leq q_0 - \left( \frac{m}{\log_2 q_0} \right) + 1
\]  

(5)

then code \(X\) has \(s\)-disjunct property and its size \(t = q_0^{k-r} \geq 2^m\).

For fixed value \(q_0 \in \mathcal{P}(m, s)\), denote by

\[
N(q_0, s, m) = \min_{\{r\}} \{q_0(q_0 + 1 - r)\}
\]

(6)

the minimal possible length of \(s\)-disjunct codes of size \(\geq 2^m\) which are based on the \(q_0\)-ary shortened RS codes. The minimum in (6) is taken over all parameters \(0 \leq r \leq k - 1 \leq q_0\) for which (5) is true.

**Lemma:** The minimum in (6) is achieved at

\[
k = q_0 + s - (s - 1) \left[ \frac{m}{\log_2 q_0} \right]
\]

(7)

\[
r = k - \left[ \frac{m}{\log_2 q_0} \right] = q_0 - s \left( \left[ \frac{m}{\log_2 q_0} \right] - 1 \right) \geq 0
\]

(8)

and the optimal length is given by

\[
N(q_0, s, m) = q_0 \left[ s \left( \left[ \frac{m}{\log_2 q_0} \right] - 1 \right) + 1 \right].
\]

(9)

**Proof:** Fix an arbitrary \(q_0 \in \mathcal{P}(m, s)\). Let the integer \(k\) be defined by (7). One can easily check that \(k\) is the root of equation

\[
\left[ \frac{m}{\log_2 q_0} \right] \leq k - r \leq q_0 - s \left( \left[ \frac{m}{\log_2 q_0} \right] - 1 \right) + 1
\]

\[
\left[ \frac{m}{\log_2 q_0} \right]
\]

\[
\left[ \frac{m}{\log_2 q_0} \right] \leq k - r \leq q_0 - s \left( \left[ \frac{m}{\log_2 q_0} \right] - 1 \right) + 1
\]

\[
\left[ \frac{m}{\log_2 q_0} \right]
\]

\[
\left[ \frac{m}{\log_2 q_0} \right] \leq k - r \leq q_0 - s \left( \left[ \frac{m}{\log_2 q_0} \right] - 1 \right) + 1
\]

\[
\left[ \frac{m}{\log_2 q_0} \right] \leq k - r \leq q_0 - s \left( \left[ \frac{m}{\log_2 q_0} \right] - 1 \right) + 1
\]
TABLE III
PARAMETERS OF CONSTANT-WEIGHT CONCATENATED \((2, N, t)\)-CODES OF WEIGHT \(w\), LENGTH \(N\), AND SIZE \(t\), \(2^m \leq t < 2^{m+1}\), \(10 \leq m \leq 32\)

<table>
<thead>
<tr>
<th>((2, q', q))-code</th>
<th>(q)-ary shortened RS-code</th>
<th>((2, N, t))-code</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m)</td>
<td>(q)</td>
<td>(q')</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>42</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>42</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>48</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>48</td>
</tr>
<tr>
<td>11</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>17</td>
<td>68</td>
</tr>
<tr>
<td>12</td>
<td>17</td>
<td>68</td>
</tr>
<tr>
<td>12</td>
<td>19</td>
<td>76</td>
</tr>
<tr>
<td>12</td>
<td>19</td>
<td>76</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>84</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>84</td>
</tr>
<tr>
<td>13</td>
<td>21</td>
<td>120</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>15</td>
<td>23</td>
<td>253</td>
</tr>
<tr>
<td>15</td>
<td>23</td>
<td>253</td>
</tr>
<tr>
<td>16</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>17</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>68</td>
</tr>
<tr>
<td>18</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>18</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>19</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>21</td>
<td>12</td>
<td>20</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>23</td>
<td>23</td>
<td>253</td>
</tr>
<tr>
<td>24</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>28</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>29</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>31</td>
<td>23</td>
<td>253</td>
</tr>
<tr>
<td>32</td>
<td>11</td>
<td>17</td>
</tr>
</tbody>
</table>

\(m/\log_2 q_0\) \(\equiv (q_0 + 1 - k)/(s - 1) + 1\). Hence, the given value of \(k\) is the maximal possible integer satisfying (5). From the left-hand side of (5) it follows \(r \leq k - [m/\log_2 q_0]\). It means that the maximal possible value of \(r\) is given by (8). Therefore, the minimum in (6) is achieved at \(r\) defined by (8) and the corresponding minimal code length is equal to the right-hand side of (9).

We can summarize as follows.

**Proposition 1:** Fix an arbitrary \(q_0 \in \mathcal{P}(m, s)\) where the subset \(\mathcal{P}(m, s) \subset \mathcal{P}\) is defined by (4). For the binary code \(X\) based on the \(q_0\)-ary shortened RS code with parameters (7) and (8), we have

1. \(X\) is the \(s\)-disjunct constant-weight code of size \(t\), length \(N\), weight \(w\), and the maximal correlation of codewords \(\lambda\), where
   \[
   \lambda = \lambda_0 = k - r - 1 = \left\lfloor \frac{m}{\log_2 q_0} \right\rfloor - 1
   \]
   \[
   w = q_0 + 1 - r = s\lambda_0 + 1
   \]
   \[
   t = q_0^{k-r} = q_0^{\lambda_0+1} = q_0^{[m/\log_2 q_0]} \geq 2^m
   \]
   \[
   N = q_0(s\lambda_0 + 1)
   \]
2. The length \(N\) of code \(X\) coincides with the minimal possible length \(N(q_0, s, m)\) defined by (9).

Denote by \(N(s, m)\) the minimal possible length of \(s\)-disjunct codes of size \(t \geq 2^m\) which are based on the shortened RS codes, i.e.,

\[
N(s, m) = \min_{\mathcal{P}(m, s)} N(q, s, m)
\]

\[
= \min_{\mathcal{P}(m, s)} \left\{ q \left[ s \left( \left\lfloor \frac{m}{\log_2 q} \right\rfloor - 1 + 1 \right) \right] \right\}
\]

In Table I, we give numerical values of \(N(s, m)\), \(q_0\) and \(\lambda_0\), when \(s = 2, 3, \ldots, 8, m = 5, 6, \ldots, 30\), and the code size \(t = q_0^{\lambda_0+1}\) satisfies the inequalities \(2^m \leq t < 2^{m+1}\). The optimal parameters are identified as follows:

\[
q_0 \geq s\lambda_0 \quad r = q_0 - s\lambda_0 \quad k = r + \lambda_0 + 1
\]

\[
\lambda_0 = \left\lfloor \frac{m}{\log_2 q_0} \right\rfloor - 1 \quad w = n_0 = s\lambda_0 + 1
\]

\[
N = q_0(s\lambda_0 + 1) \quad t = q_0^{\lambda_0+1} \quad 2^m \leq t < 2^{m+1}
\]

(10)
Table IV

<p>| Parameters of Constant-Weight Concatenated ((3, q, t))-Codes of Weight (w). Length (N) and Size (t &lt; 2^{n+1}), (m = 8, 9, \ldots, 62) |
|---|---|---|---|---|---|---|---|</p>
<table>
<thead>
<tr>
<th>(m)</th>
<th>(q)</th>
<th>(q')</th>
<th>(w')</th>
<th>(q_0)</th>
<th>(k)</th>
<th>(t = q_0^{k-1})</th>
<th>(\lambda_0)</th>
<th>(N = q_0\lambda_0)</th>
<th>(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>7(^3)</td>
<td>7</td>
<td>2</td>
<td>49</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>8(^3)</td>
<td>7</td>
<td>2</td>
<td>56</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>1</td>
<td>11</td>
<td>8</td>
<td>5</td>
<td>11(^3)</td>
<td>7</td>
<td>2</td>
<td>77</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>9(^4)</td>
<td>10</td>
<td>3</td>
<td>90</td>
</tr>
<tr>
<td>13</td>
<td>11</td>
<td>1</td>
<td>11</td>
<td>6</td>
<td>2</td>
<td>11(^4)</td>
<td>10</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>1</td>
<td>13</td>
<td>8</td>
<td>4</td>
<td>13(^4)</td>
<td>10</td>
<td>3</td>
<td>130</td>
</tr>
<tr>
<td>15</td>
<td>22</td>
<td>37</td>
<td>4</td>
<td>37</td>
<td>34</td>
<td>31</td>
<td>37(^3)</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>1</td>
<td>16</td>
<td>11</td>
<td>7</td>
<td>16(^4)</td>
<td>10</td>
<td>3</td>
<td>160</td>
</tr>
<tr>
<td>18</td>
<td>13</td>
<td>1</td>
<td>13</td>
<td>6</td>
<td>1</td>
<td>13(^5)</td>
<td>13</td>
<td>4</td>
<td>169</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>30</td>
<td>4</td>
<td>29</td>
<td>24</td>
<td>20</td>
<td>29(^4)</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>21</td>
<td>16</td>
<td>20</td>
<td>4</td>
<td>19</td>
<td>12</td>
<td>7</td>
<td>19(^5)</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>23</td>
<td>19</td>
<td>25</td>
<td>4</td>
<td>25</td>
<td>18</td>
<td>13</td>
<td>25(^5)</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>16</td>
<td>20</td>
<td>4</td>
<td>19</td>
<td>10</td>
<td>4</td>
<td>19(^6)</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>26</td>
<td>22</td>
<td>37</td>
<td>4</td>
<td>37</td>
<td>30</td>
<td>25</td>
<td>37(^3)</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>29</td>
<td>16</td>
<td>20</td>
<td>4</td>
<td>19</td>
<td>1</td>
<td>19(^7)</td>
<td>19</td>
<td>6</td>
<td>304</td>
</tr>
<tr>
<td>31</td>
<td>22</td>
<td>37</td>
<td>4</td>
<td>37</td>
<td>28</td>
<td>22</td>
<td>37(^4)</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>37</td>
<td>19</td>
<td>25</td>
<td>4</td>
<td>25</td>
<td>12</td>
<td>4</td>
<td>25(^5)</td>
<td>22</td>
<td>7</td>
</tr>
<tr>
<td>41</td>
<td>19</td>
<td>25</td>
<td>4</td>
<td>25</td>
<td>10</td>
<td>1</td>
<td>25(^6)</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>46</td>
<td>22</td>
<td>37</td>
<td>4</td>
<td>37</td>
<td>22</td>
<td>14</td>
<td>37(^5)</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>52</td>
<td>22</td>
<td>37</td>
<td>4</td>
<td>37</td>
<td>20</td>
<td>10</td>
<td>37(^5)</td>
<td>28</td>
<td>9</td>
</tr>
<tr>
<td>62</td>
<td>22</td>
<td>37</td>
<td>4</td>
<td>37</td>
<td>16</td>
<td>4</td>
<td>37(^7)</td>
<td>34</td>
<td>11</td>
</tr>
</tbody>
</table>

Table I shows the solutions of (10), i.e., \(\lambda_0\) and \(q_0\), yielding the minimal length \(N = N(s, m)\) if for the given integers \(s\) and \(m\), these solutions exist. The optimal solutions were calculated with the help of a computer program.

**Example:** For the case \(s = 3, m = 10\), Table I gives \(q_0 = 11\), \(\lambda_0 = 2\), \(N = 77\). It means that there exists 3-disjunct constant-weight code with parameters

\[
\lambda = \lambda_0 = 2, \quad w = s\lambda_0 + 1 = 7, \quad t = q_0^{\lambda_0 + 1} = 11^3 = 1331, \quad N = 11 \cdot 7 = 77.
\]

This code is obtained from shortened RS code with parameters \(q_0 = 11, k = 7, r = 4\).

**Remark 1:** By the boldface type, we marked two examples of the superimposed code parameters which were known from [1] and [5].

**Remark 2:** In [14], we give the detailed investigation of superimposed codes with parameter \(\lambda = 1\).

**Discussion:** Table I contains the values \(R(s)\) (lower bound on the rate \(R(s)\) of the optimal code), the values \(\bar{R}(s)\) (upper bound on the rate \(R(s)\) of the optimal code), and the values of the rate for several obtained codes, namely, the values of fraction \(m/N\), \(m = 12, 20, 25, 29\). The comparison yields the following conclusions.

- If \(s = 2\) and \(m \leq 15\), then the values \(m/N\) exceed the random coding rate \(R(2) = 0.182\).

- If \(s \geq 3\) and \(m \leq 30\), then the values \(m/N\) exceed the random coding rate \(R(s)\).

**III. SUPERIMPOSED CONCATENATED CODES**

A further extension of the Kautz–Singleton superimposed \((s, N, t)\) codes is based on the following concatenated codes which were suggested in [1] and also were discussed in [15] and [16].

Consider the \(q_0\)-ary shortened Reed–Solomon code with parameters (10) where \(q_0\) is a prime power. Let \(q_0\)-ary symbols of this code be substituted, i.e., be coded, for the binary codewords of a known constant-weight \(s\)-disjunct code of size \(q' \geq q_0\), length \(q \leq q_0\), and weight \(w' < q\). Denote this binary superimposed code as an \((s, q, q')\)-code. We will apply \((s, q, q')\)-codes which are standard constant-weight \((n, D, w')\)-codes of size \(A(n, D, w') = q',\) length \(n = q, \) weight \(w' = s\lambda + 1, \) distance \(D = 2(w' - \lambda'), \) and the maximal correlation \(\lambda'.\) It is easy to check that the result of Section II can be generalized as follows.

**Proposition 2:** The given substitution yields the concatenated code which is the binary constant-weight superimposed \((s, q_0n_0, q_0^{q_0 + 1})\)-code of weight \(w' = w_0\).

In Section II, we used only the trivial substitution, where \(q_0 = q = q'\) and \(w' = 1\).

**Remark 3:** If we apply the trivial substitution \(q_0 = q = q'\), i.e., \(w' = 1\), then we obtain the concatenated code which is a standard constant-weight \((N, D, w')\)-code of size \(t,\) where

\[
N = q_0n_0, \quad w = n_0, \quad t = q_0^{k-1}, \quad D = 2(n - \lambda_0) = 2(q_0 - k + 2) = 2d.
\]
Remark 4: If \( q < q_0 \) and \( w' > 1 \), then one knows only the weight \( w = w' n_0 \) of the constant-weight concatenated code. We cannot identify its distance \( D \) and the maximal correlation \( \lambda \).

Let \( D = 2, 4, 6, \ldots, D \leq n \), and \( w \leq n \) be arbitrary integers. Denote by \( A(n, D, w) \) the maximal size of the corresponding constant-weight code which is known up to now. The tables of \( A(n, D, w) \) [17], [18] are called standard tables (ST).

Tables II–IV give the numerical values of the best known parameters for superimposed concatenated \((s, N, t)\)-codes, when \( s = 2 \) and \( s = 3 \).

Description of Table II: We tabulate the values of \( t > N \) for \((s, N, t)\)-superimposed codes of strength \( s = 2, 3 \) and length \( N = 9, 10, \ldots, 47 \). The symbol \( t = A(N, D, w) \) denotes the code size, i.e., the number of codewords (columns), \( N \) is the code length, \( w = s\lambda + 1 \) is the code weight, and \( D = 2(w - \lambda) \) is the code distance. Table II is calculated as follows. Let \( N = 9, 10, \ldots, 47 \) be fixed. In ST, we are looking for the pair \((D, w)\) such that

\[
 w = s\lambda + 1, \quad D = 2(w - \lambda) \iff w(s - 1) = \frac{D}{2} s - 1
\]

and the value of size \( A(N, D, w) \) is maximal. Table II gives these maximal values if \( s = 2, 3 \). For \( s = 2, N = 45, 48 \), and \( s = 3, N = 49 \), we use the boldface type because the corresponding codes are concatenated. We will repeat them in Tables III and IV. For \( s = 2, 3 \) and \( N = 49 \), the concatenated codes are the standard constant-weight codes because they are obtained by the trivial substitution. These codes are new for the standard tables and should be included in ST.

Description of Tables III and IV: We tabulate the parameters of the best known concatenated \((s, N, t)\)-codes of strength \( s = 2, 3 \) and length \( N \geq 45 \). For values \( N \) of the form \( N = q_0 n_0 \), we give the parameters of the \((s, q, q')\)-code and the shortened RS code yielding the maximal possible size \( t \) of the form \( t = q_0^{-m} \). To calculate the parameters of the shortened RS code, we apply the particular cases of (10) corresponding to \( s = 2 \) or \( s = 3 \). The groups of \( t \) values for which \( 2^m \leq t < 2^{m+1}, w = 8, 9, \ldots \), are separated. The Reed–Solomon codes are a known class of maximum-distance separable codes (MDS codes) [19], [2] which could be applied for our concatenated construction. In Table III, we give the parameters of several concatenated codes, which could be obtained with the help of the Latin square \( q_0 \times q_0 \), when \( q_0 \geq 2 \) is an arbitrary integer. We remind [19], [2], that any Latin square \( q_0 \times q_0 \) yields the \( q_0 \)-ary MDS-code of size \( q_0^2 \), length \( n_0 = 3 \), and distance \( d = 2 \), i.e., \( \lambda_0 = 1 \).

Discussion: The preliminary conclusions could be formulated as follows.

- For fixed integers \( s = 2, 3, \ldots \) and \( m \geq 1 \), we constructed the family \( \mathcal{F} = \mathcal{F}(s, m) \) of binary concatenated \( s \)-disjunct constant-weight codes of size \( t \), \( 2^m \leq t < 2^{m+1} \), based on the \( q_0 \)-ary shortened RS codes \( q_0 \) satisfies the inequality in (10)) and the standard binary constant-weight codes.

- For \( s = 2 \), Table III shows the stability of the code rate, i.e., \((m/N) \approx 0.2 > 0.182, m = 11, 12, \ldots \). It means that for \( s = 2 \), the code rate of the family \( \mathcal{F} \) exceeds the random coding rate \( R(2) = 0.182 \) [3].

- For \( s = 3 \) and \( m = 15, 16, \ldots, 21 \), the stable code rate \((m/N) \approx 0.1 \) also exceeds the corresponding random coding rate \( R(3) = 0.079 \) [3].

- It seems to us that the calculations of the best parameters of \( \mathcal{F} = \mathcal{F}(3, m) \), \( m \geq 8 \), are not completed and some values from Table IV could be improved.

Example: For the length \( N = 45 = 3 \cdot 15 \), the \((s = 2, N = 45, t = 4^2 = 1764)\)-code is the concatenated code based on the \( \text{binary constant-weight} (n = 15, D = 6, w = 5)\)-code of size \( A(15, 6, 5) = 42 \) and the Latin square \( 42 \times 42 \) is 1764.

References


The Viterbi Algorithm and Markov Noise Memory
Aleksandar Kavčić, Member, IEEE, and
José M. F. Moura, Fellow, IEEE

Abstract—This work designs sequence detectors for channels with intersymbol interference (ISI) and correlated (and/or signal-dependent) noise. We describe three major contributions. i) First, by modeling the noise as a finite-order Markov process, we derive the optimal maximum-likelihood sequence detector (MLSD) and the optimal maximum a posteriori (MAP) sequence detector, extending to the correlated noise case the Viterbi algorithm. We show that, when the signal-dependent noise is conditionally Gauss–Markov, the branch metrics in the MLSD are computed from the conditional second-order noise statistics. We evaluate the branch metrics using a bank of finite impulse response (FIR) filters. ii) Second, we characterize the error performance of the MLSD and MAP sequence detector. The error analysis of these detectors is complicated by the correlation asymmetry of the channel noise. We derive upper and lower bounds and computational efficient approximations to these bounds based on the banded structure of the inverses of Gauss–Markov covariance matrices. An experimental study shows the tightness of these bounds. iii) Finally, we derive several classes of suboptimal sequence detectors, and demonstrate how these and others available in the literature relate to the MLSD. We compare their error rate performance and their relative computational complexity, and show how the structure of the MLSD and the performance evaluation guide us in choosing a best compromise between several types of suboptimal sequence detectors.

Index Terms—Correlated noise, Gauss–Markov processes, intersymbol interference, Markov channel noise, Markov memory, maximum-likelihood sequence detection, Viterbi algorithm.

I. INTRODUCTION

In digital communications, the Viterbi algorithm [1] is the maximum-likelihood sequence detector (MLSD), [2], [3], for channels with intersymbol interference (ISI) and memoryless noise. Under these conditions, if \( \{x_k\} \) is the state-channel input sequence, and \( \{z_k\} \) is the observable sequence, we have that

\[
 f (z_k | z_{k-1}, \ldots, z_{-\infty}, x_{-\infty}, \ldots, x_{-\infty}) = f (z_k | x_{k-1}, x_k)
\]

(1)

where \( f \) denotes the conditional probability density function (pdf) of the observable channel output, conditioned on the infinite past observable sequence and on the whole channel input sequence. Equation (1) states that under an appropriate definition of the channel state-sequence, the ISI white Gaussian noise channel is memoryless [4].

In many applications, (1) is not an appropriate model, because, besides ISI, the channel noise is correlated, with correlation statistics that are possibly signal-dependent. Our research is motivated by detection in high-density magnetic recording [5] where, for example, the statistics of percolation effects between transitions depend strongly on the particular data sequence recorded: percolation is more likely in closely separated transitions (i.e., successive 1’s in the nonreturn to zero inverted (NRZI) recorded data sequence) than in widely separated recorded transitions (i.e., successive data symbols are 0’s). Reference [6] illustrates with experimental evidence this type of signal-dependent noise in high-density magnetic recording. Due to percolation (but also due to nonlinear transition shifts), the noise is correlated across transitions, with the correlation statistics depending on the signal recorded.

This signal-dependent correlated noise may overwhelm the white-noise component, becoming the dominant noise feature, and may severely degrade the performance of the sequence detector designed for a white Gauss noise ISI channel. Generalizing the Viterbi algorithm to these more complex channels raises the difficulty that, due to the noise correlation, it is no longer valid to truncate the conditioning in (1) to a finite sequence. In other words, with general correlated noise there is no appropriate definition for the state sequence under which the channel becomes memoryless, and so the model (1) becomes invalid. Early work on this problem includes [7]–[9]. More recently, the noise prediction method [10]–[12], and the so-called \( K \)-step Viterbi detector [13] are attempts to combat the noise correlation in magnetic recording.

In this work, we extend the Viterbi algorithm to channels with ISI and signal-dependent correlated noise: we develop the optimal maximum-likelihood sequence detector (MLSD) and study its error performance. The key to our approach is our modeling of the noise: we consider ISI channels with correlated noise, which goes well beyond white noise ISI channels, but we impose structure on the statistical properties of the correlated noise. We describe the noise, not as a general Gaussian correlated noise, but as a Gauss–Markov correlated noise whose second-order statistics are signal-dependent. We use the Markov assumption to reduce the memory of the noise to finite length. We show in Section II that under the noise Markovian property and with an appropriate definition for the input state sequence, the following holds:

\[
 f (z_k | z_{k-1}, \ldots, z_{-\infty}, x_{-\infty}, \ldots, x_{-\infty}) = f (z_k | x_{k-1}, x_k)
\]

(2)

where \( L \) is the so-called Markov memory length of the noise. In Section II, we describe the basic Markovian-memory channel model, and then develop examples of channels with such characteristics. For these Markov channels, the derivation of the Viterbi algorithm becomes a simple exercise in application of Bayes’ law. We present the Viterbi algorithm in Section III as the solution to maximum a posteriori (MAP) and maximum-likelihood (ML) sequence detection in Markovian-memory channels. In this section, we consider the general case of Markov noise, not necessarily Gauss. The recent paper [14] also notices that the Viterbi algorithm solves the MLSD in channels with additive Markov noise. In Section IV, we derive explicit expressions for the branch metric when the noise in the channel is signal-dependent additive Gauss–Markov. To compute efficiently the branch metrics, we use the properties of Gauss–Markov processes, namely, the structure of the inverses of the covariance matrices of such processes [15]: we compute the branch metrics by using finite impulse response (FIR) filters. We also discuss in Section IV adaptive implementations of these FIR filters that track the signal-dependence characteristics of the noise statistics.

A second contribution of this research is the error analysis we develop for the MLSD and MAP sequence detector for ISI channels with finite memory correlated noise. The error performance of the MLSD

Manuscript received December 22, 1997; revised February 17, 1999.

A. Kavčić is with the Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: kavcic@hl.harvard.edu).

J. M. F. Moura is with LIDS and the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA, on sabbatical leave from the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: moura@mit.edu).

Communicated by E. Soljanin, Associate Editor for Coding Techniques.

Publisher Item Identifier S 0018-9448(00)00033-X.
is considered in Section V. We derive upper and lower bounds for the error probability and by exploiting the structure of the inverse of the covariance matrix of a Gauss–Markov process we develop computationally efficient approximations to the bounds. We demonstrate the tightness of these bounds with a simulation study in realistic scenarios.

The third major issue we address is the derivation of suboptimal sequence detectors. Guided by the structure of the optimal MLSD and its error performance, we introduce in Section VI new suboptimal sequence detectors and show how these and other suboptimal detectors, including the noise prediction and K-step detectors, relate to the MLSD. We illustrate their error performance and determine within a certain class of suboptimal sequence detectors the best compromise from an error performance and computational complexity perspective. Finally, Section VII concludes the work.

Notation: Before leaving this section, we introduce the basic notation used below. Throughout, we assume that the reader is familiar with the Viterbi algorithm as presented in [3]. We attempt to use the notation in [3] to the extent possible. Column vectors are denoted by underlined characters, matrices are denoted by boldface characters, and the superscript T is used to denote matrix and vector transposition. If \( z_k \) is a discrete-time indexed sequence then \( k \) denotes the time, the column vector of sequence samples at time \( k \) through \( k + 1 \) is denoted by \( z_{k:k+1} \). The notation \( \{a, B\} \sim \mathcal{N}(m, \Sigma) \) denotes that the random vector \( \{a, B\} \), conditioned on an event \( B \), has a normal (joint Gaussian) distribution with mean \( m \) and covariance matrix \( \Sigma \). The probability of an event \( A \) is denoted by \( P(A) \). Further notation is introduced in Section V.

II. PROCESS MODEL

In this section, we establish the appropriate model for ISI channels with signal-dependent correlated noise. We structure the noise so that the resulting channel has finite memory. In Section II-A, we discuss the general Markov chain model with finite memory, and in Section II-B we illustrate this model with examples that are relevant to the ISI channel with correlated noise.

A. The Model

Let \( x_k \) represent the state of a finite-state machine at time instant \( k \). The number of possible states \( M \) is finite. We shall assume that the sequence of states \( x_k \) is a Markov chain in the sense that the probability of the machine being in state \( x_k \), conditioned on all states up to the state at time \( k - 1 \), depends only on the state \( x_{k-1} \), i.e.,

\[
P(x_k | z_{k-1}^\infty) = P(x_k | x_{k-1}).
\] (3)

The transition probabilities \( P(x_k | x_{k-1}) \) may be time-dependent, as originally presented by Forney [3]. This is assumed throughout, although not explicitly indicated.

Let the transition between states \( x_{k-1} \) and \( x_k \) produce an observable output random variable \( z_k \), where the statistics of \( z_k \) may depend on the transition \( (x_{k-1}, x_k) \), and time \( k \). We assume that the sequence of output variables \( z_k \) is a Markov sequence with a finite memory length \( L \), whose memory is dependent only on the state transition \( (x_{k-1}, x_k) \), i.e., the conditional probability density function (pdf) obeys

\[
f(z_k | z_{k-1}^\infty, x_{k-1}, x_k) = f(z_k | x_{k-1}, x_k).
\] (4)

Again, these pdf’s may also depend on time, which we will not show explicitly.

B. Examples

Example 1. Channel with ISI and Data-Dependent Gauss–Markov Noise: Let \( \alpha_k \) be a sequence of transmitted symbols (bits). We model a channel with intersymbol interference (ISI) of length \( I \geq 0 \) as

\[
z_k = y(z_k^{-1}) + n_k.
\] (5)

In (5), \( y(q_k^{-1}) \) is the noiseless channel output dependent only on the \( I + 1 \) latest transmitted bits. For linear channels, \( y(q_k^{-1}) \) may be represented as the convolution of the channel response with the transmitted sequence. We consider the channel to be nonlinear. The additive noise term \( n_k \) is considered to be a signal-dependent Gauss–Markov noise process with Markov memory length \( L \), i.e.,

\[
n_k = b(a_k^{-1})^T \eta_k + \sigma(a_k^{-1}) w_k.
\] (6)

The vector

\[
b(a_k^{-1}) = [b_0(a_k^{-1}), \ldots, b_L(a_k^{-1})]^T
\]

collects the \( L \) coefficients of an autoregressive filter whose values are dependent on the transmitted symbols \( a_k^{L-k-1} \), \( w_k \) is a zero-mean unit-variance white Gaussian noise process, and \( \sigma(a_k^{-1}) \) is a signal-dependent standard deviation.

Fig. 1 depicts the channel described by (5) and (6). As we will derive in Section IV, (4) holds for this channel if the state is chosen as a collection of \( I + L \) consecutive transmitted symbols \( x_k = a_k^{L-k-1} \). If we further have \( P(x_k, a_k|z_{k-1}^\infty) = P(x_k|a_k|z_{k-1}^\infty) \), then (3) holds too and the channel may be represented by a finite-state machine. If \( a_k \) are binary symbols, then there are \( 2^{L+I} \) states and \( 2^{I+L+1} \) transition branches in this finite-state machine.

Example 2. Noncausal ISI and Noncausal Data-Dependent Gauss–Markov Noise: The following is a generalization of Example 1 to noncausal ISI and noncausal data-dependent Gauss–Markov noise.

\[
z_k = y(z_k^{I+L}) + n_k
\] (7)

\[
n_k = \alpha \left( a_k^{L+1} \right)^T \eta_k + \sigma(a_k^{L+1}) w_k.
\] (8)

Here \( I_0 \geq I_1, A_2 \geq A_1, B_2 \geq B_1, L_1 \geq 0, \) and \( L_2 \geq 0 \). For this channel the state is chosen as \( x_k = a_k^{L_1} \), where

\[
C_1 = \min(I_1 - L_2 - L_1, A_1 - L_2, B_1 - L_2)
\]

and

\[
C_2 = \max(I_0, A_2 - L_2, B_2 - L_2).
\]

Example 1 is a special case of this example when \( I_0 = A_2 = B_2 = L_2 = 0 \), \( I_1 = A_1 = B_1 = -I, \) and \( L_1 = L \). Note that in this example the noise model is noncausal Markov, i.e., the noise at the current time may be dependent on past and future noise values. This is the appropriate model, for example, in magnetic recording, where time stands for the spatial variable, and the “past” and “future” times stand for the immediately adjacent bit-slots on a physical medium.
Example 3. Non-Markov Observables: Often the observed process is a deterministic part corrupted by additive noise. The additive noise may not be Markov, but often it may be approximated by a Markov process if the memory length \( L \) is taken long enough [16]. In this case, the Viterbi algorithm which we present in the next section is the asymptotically optimal detector of the finite-state machine state sequence.

III. VITERBI SEQUENCE DETECTION

A. MAP Sequence Detection

We study the estimation of the state sequence \( x_1, \ldots, x_K \) \((K > 0)\) given a sequence of observables \( z_1, \ldots, z_K \) when (3) and (4) hold. We assume that the initial state \( z_0 \) is known and that the initial \( L \) realizations \( z_{-1}, \ldots, z_0 \) of the observable random variable are also given, where \( L \) is the Markovian memory length of the observable sequence \( z_k \).

The maximum \textit{a posteriori} (MAP) sequence estimate of the state sequence \( \hat{x}_k^L = \{x_1, \ldots, x_K\}^T \) is the sequence \( \hat{x}_k^L = \{\hat{x}_1, \ldots, \hat{x}_K\}^T \) that maximizes the joint conditional pdf, i.e.,

\[
\hat{x}_k^L = \arg \max_{x} f(\hat{x}_k^L | x_0, z_{-1}^{L+1}). \quad (9)
\]

As shorthand notation, denote by \( f(x, z | \text{i.c.}) \) the conditional pdf on the right-hand side of (9), where \( \text{i.c.} \) stands for initial conditions. Using (3) and (4), we can factor \( f(x, z | \text{i.c.}) \) as

\[
f(x, z | \text{i.c.}) = \prod_{k=1}^{K} P(x_k | x_{k-1}, z_0^{L+1}) \prod_{k=1}^{K} f(z_k | z_{-1}^{L+1}, x_0^L) \]

\[
= \prod_{k=1}^{K} P(x_k | x_{k-1}, x_k) \prod_{k=1}^{K} f(z_k | z_{-1}^{L+1}, x_{k-1}, x_k). \quad (10)
\]

In (10), we assumed that \( P(x_k | z_0^{L+1}, z_{-1}^{L+1}) = P(x_k | z_{-1}^{L+1}) \), i.e., that the state transition probabilities are independent of the observables. Using the definition of conditional pdf’s, we may further substitute

\[
f(z_k | z_{-1}^{L+1}, x_{k-1}, x_k) = \int_{z_k} f(z_k | x_{k-1}, x_k) \frac{f(z_k | z_{-1}^{L+1}, x_0^L)}{f(z_k | z_{-1}^{L+1}, x_{k-1}, x_k)} \]

\[
= \int_{z_k} f(z_k | x_{k-1}, x_k) \frac{f(z_k | x_{k-1}, x_k)}{f(z_k | z_{-1}^{L+1}, x_{k-1}, x_k)} \]  

for the last term in (10). Maximizing (10) is the same as minimizing its negative logarithm. Taking the negative logarithm of \( f(x, z | \text{i.c.}) \), we get

\[
-\ln f(x, z | \text{i.c.}) = \sum_{k=0}^{K-1} \Lambda_{\text{MAP}} (z_{k-L}^L, x_{k-L}, x_k) \quad (12)
\]

where the MAP branch metrics are

\[
\Lambda_{\text{MAP}} (z_{k-L}^L, x_{k-L}, x_k) = -\ln P(x_k | x_{k-1}) - \int_{z_k}^{\infty} \ln f(z_k | x_{k-1}, x_k) \frac{f(z_k | z_{-1}^{L+1})}{f(z_k | z_{-1}^{L+1}, x_{k-1}, x_k)} dz_k. \quad (13)
\]

The MAP sequence estimate is thus that sequence \( \hat{x}_k^L \) for which the sum of the branch metrics in (12) is minimized.

B. ML Sequence Detection

The maximum-likelihood (ML) sequence estimate is that sequence \( \hat{x}_k^L \), for which the conditional pdf

\[
f(z | x, \text{i.c.}) = f(z_k | z_{-1}^{L+1}) \quad (14)
\]

is maximized. With arguments similar to those for MAP sequence detection, it follows that the ML estimate is that sequence \( \hat{x}_k^L \) for which the following sum of branch metrics is minimized:

\[
- \ln f(z | x, \text{i.c.}) = \sum_{k=0}^{K-1} \Lambda_{\text{ML}} (z_{k-L}^L, x_{k-L}, x_k). \quad (15)
\]

The ML branch metric is

\[
\Lambda_{\text{ML}} (z_{k-L}^L, x_{k-L}, x_k) = - \ln \int_{z_k}^{\infty} \frac{f(z_k | x_{k-1}, x_k)}{f(z_k | z_{-1}^{L+1}, x_{k-1}, x_k)} dz_k. \quad (16)
\]

C. Viterbi Trellis Implementation

The implementation of the minimization in (12) and (15) is very similar to the standard implementation described in [2] and [3]. The major difference is that the computation of the branch metrics relies now on a window of observed samples \( \hat{z}_{k-L}^L = [z_k-L, \ldots, z_k]^T \), instead of just one sample on \( z_k \). Depending on whether we are interested in MAP or ML sequence detection, the branch metrics are given by (13) and (16), respectively.

IV. GAUSSIAN STATISTICS AND BRANCH METRICS

We now consider the important case in applications, [16], of channels with intersymbol interference (ISI) and signal-dependent Gaussian–Markov noise as described in Example 1 in Section II and depicted in Fig. 1. The extension to the more general context of Example 2 is relatively easy. In the model of Example 1, \( I \geq 0 \) is the ISI length, and \( L \geq 0 \) is the Markovian memory length. We also assume that \( a_k \) is a binary symbol sequence.

A. Branch Metrics

We first define the state \( x_k \). We consider the ML sequence detection and work with (16). To find the metric \( \Lambda_{\text{ML}} (z_{k-L}^L, x_{k-L}, x_k) \), we must find the logarithm of the pdf \( f(z_k | x_{k-L}^L, x_k) \). The vector \( z_{k-L}^L \) in (16) involves observables \( z_k-L \) through \( z_k \), whose conditional means (ideal channel outputs) are \( y(z_{k-L}^L) \) through the channel outputs \( y(z_{k-L}^L) \), see (5). Therefore, the trellis branch \( f(x_{k-L}^L, x_k) \) should be chosen so that it collects all inputs \( a_{k-L} \) through \( a_k \) involved in determining the channel outputs \( y(z_{k-L}^L) \) through the channel outputs \( y(z_{k-L}^L) \). A simple choice is to label the state by grouping the incoming bits as

\[
x_k = a_k^{k-L+1}. \quad (17)
\]

We define the vector of \( L + 1 \) ideal channel outputs

\[
\Sigma(x_{k-L}, x_k) = [y(a_k^{k-L}), \ldots, y(a_k^{L})]^T
\]

and rewrite (5) as

\[
z_{k-L}^L = \Sigma(x_{k-L}, x_k) + n_k^{k-L}. \quad (18)
\]

Since \( n_k \) is a conditionally Gaussian process, we have

\[
(z_{k-L}^L | x_{k-L}, x_k) \sim N(\Sigma(x_{k-L}, x_k)C(x_{k-L}, x_k))
\]
where \( \mathbf{C}(x_{k-1}, x_k) \) is the \((L + 1) \times (L + 1)\) covariance matrix of \( \frac{1}{\sigma_k} \) conditioned on the pair of states \((x_{k-1}, x_k)\). Substituting this pdf into the expression for the ML branch metric (16), and cancelling constant terms common to all branches, we obtain the Gaussian ML branch metric

\[
\mathcal{M}_{\text{ML}} \left( \frac{1}{\sigma_k} x_{k-1}, x_k \right) = \ln \sigma^2 \left( \frac{1}{\sigma_k} x_{k-1} \right) + \frac{1}{\sigma^2} \left( \frac{1}{\sigma_k} x_{k-1} - \frac{1}{\sigma_k} x_{k-1} \right)^2.
\]  

(25)

The metric in (25) is obtained by filtering the observed vector \( \frac{1}{\sigma_k} x_{k-1} \) through an FIR filter with coefficients \( \mathbf{w}_o(x_{k-1}, x_k) \) given in (24). This filter is the inverse of the autoregressive filter in Fig. 1 and it uncorrelates the autoregressive (AR) process \( n_k \). Thus we may interpret the metric (25) as first uncorrelating the noise with an FIR filter, and then applying the Euclidean (square) metric to the output of the filter. This is illustrated in Fig. 2 where the branch metric is shown to be the processed output of a branch-dependent FIR filter. Since the FIR filter coefficients \( -\mathbf{b}(\mathbf{a}_k^{-1}) \) depend on only \( I + 1 \) symbols, and since there is a total of \( 2^{I+1} \) branches in each stage of the trellis, it turns out that there are \( 2^I + 1 \) distinct FIR filters in the detector, where \( 2^I \) branches may share a single filter.

A special case of the above FIR filter occurs when the covariance matrix is not signal-dependent, i.e., \( \mathbf{C}(x_{k-1}, x_k) = \mathbf{C} \). In this case, the filter \( \mathbf{w}_o(x_{k-1}, x_k) = \mathbf{w}_o \) is a state-invariant tap-delay line that whitens the stationary Gauss–Markov noise process. This is an intuitively pleasing result, since it is well known that a stationary (causal) Gauss–Markov AR sequence can be whitened by a time-invariant FIR filter. Notice also that, if \( \mathbf{w}_o \) is state-invariant, all branches share the same filter. We can then move the whitening filter to the front end of the Viterbi detector. This provides another explanation of why the Viterbi detector is optimal only when the channel noise is Markov. Suppose that the noise is not Markov, i.e., it is not the output of an autoregressive filter as in (6). Then to whiten this noise requires passing the observed channel outputs through an infinite impulse response (IR) filter, thus introducing infinite-length ISI. However, with infinite-length ISI, the Viterbi detector requires an infinite number of states in order to be the optimal detector, which is clearly unrealizable.

We make a distinction here with what is usually done in Kalman–Bucy filtering (KBF) [20] with Markov observation noise, see [21, Ch. 11]. In this case, the (vector) observations \( \frac{1}{\sigma_k} \) are modeled as

\[
\frac{1}{\sigma_k} = \mathbf{H}_k x_k + \mathbf{v}_k
\]  

(26)

where \( x_k \) is the state vector and \( \mathbf{v}_k \) is a first-order Markov vector following the model

\[
\mathbf{v}_{k+1} = \mathbf{A}_k \mathbf{v}_k + \mathbf{w}_k
\]  

(27)

where \( \mathbf{w}_k \) is a white Gauss vector sequence. Preprocessing the observations \( \frac{1}{\sigma_k} \) into the new observation

\[
\frac{1}{\sigma_k} = \frac{1}{\sigma_k} - \mathbf{A}_k \frac{1}{\sigma_k}
\]  

(28)

one can then apply standard KBF theory. Although similar, there is a distinct difference with Viterbi decoding. Due to the requirement of finite ISI, the Markov noise model (27) needs to be restricted to autoregressive processes (see (6)), while in KBF no such restriction is required.

Finally, we comment on the practical implementation of the branch metric computation. As seen from (19) or (25), implementing the

\[
\mathcal{M}_{\text{ML}} \left( \frac{1}{\sigma_k} x_{k-1}, x_k \right) = \ln \sigma^2 \left( \frac{1}{\sigma_k} x_{k-1} \right) + \frac{1}{\sigma^2} \left( \frac{1}{\sigma_k} x_{k-1} - \frac{1}{\sigma_k} x_{k-1} \right)^2.
\]  

(25)
branch label $a_0 \cdots a_t \cdots a_{t+L}$

filter $w_c$: addressed by $a_t$ through $a_{t+L}$

variance $\sigma^2$: addressed by $a_0$ through $a_{t+L}$

vector $Y$: addressed by $a_0$ through $a_{t+L}$

$D \quad z_{k-1} \quad D \quad z_{k-L} \quad 1 \quad -b_1 \quad \cdots \quad -b_{t'} = \text{filter } w_c$

$\sum \quad \frac{1}{\sigma} \quad \ln \sigma^2 \quad (\cdot)^2 \quad \mathcal{M}_{ML}$

Fig. 2. Block diagram of an FIR filter computing the ML branch metric for a channel with ISI of length $I$ and Gauss–Markov noise with memory length $L$. Since there are $2^{I+L+1}$ branches in each stage of the trellis, the branches may be labeled by $I + L + 1$ bits. Of these bits, only $I + 1$ are used to address the FIR filters. A bank of $2^{I+L}$ different FIR filters is needed for optimal detection, while $2^L$ branches share the same filter.

ML branch metric requires knowledge of the second-order statistics $C(x_{k-1}, x_k)$. In practice, the second-order statistics are typically not known and need to be estimated from the data. Tracking the statistics adaptively is preferable in many applications because the covariance statistics may time-drift or be nonstationary for some other reason. Adaptive covariance tracking has been widely covered in the literature, see, e.g., [18] and [22]. For a method applied in magnetic recording signal detection see [5]. Notice also that, if the involved covariance matrices are signal-dependent, and so nonstationary and non-Toeplitz, many efficient AR estimation algorithms [18], [23] based on the Levinson–Durbin recursion will then not work.

V. ERROR ANALYSIS

We develop in this section the error analysis for the Viterbi sequence decoder when the noise in the ISI channel is signal-dependent and correlated. Our results extend the error analysis in [2] and [24] to correlated finite memory ISI channels. However, due to the channel nonlinearity and the signal-dependent noise correlation, the analysis must carefully treat the lack of symmetry. For example, in contrast with the linear channel with stationary white noise, if $a'_1, \cdots, a'_m$ and $a''_1, \cdots, a''_m$ are two valid binary sequences in the trellis, the probability of detecting $a'$ when $a''$ is sent is not the same as detecting $a''$ when $a'$ is sent. Also, unlike the linear channel with stationary white noise, we cannot formulate the upper bound using the flowgraph transfer function [2] because relative distances between trellis paths are not symmetric due to the asymmetric signal-dependent noise correlation.

To formulate the bounds, we introduce first the necessary notation. Then, in Section V-A, we derive the binary hypothesis error probability for asymmetric Gauss–Markov noise channels, followed in Section V-B by a computationally efficient approximation to this error probability. In Sections V-C and V-D, we formulate upper and lower bounds using the binary hypothesis error probability expressions. Finally, in Section V-E, we illustrate the error performance results by analyzing the detector performance of a simple ISI channel with finite memory Markov noise.

Notation: Let

$$\varepsilon_M = (\Psi_M', \Psi_M'')$$

denote a length-$M$ error event. By $\Psi_M'$, we denote the correct path through the trellis spanning $M + 1$ consecutive states $x'_k, \cdots, x'_{k+M}$. The erroneous path is denoted by $\Psi_M''$ and spans states $x''_k, \cdots, x''_{k+M}$. It is assumed that $x'_k = x''_k, x'_{k+1} = x''_{k+1}, \cdots, x'_{k+M} = x''_{k+M}$, for $1 \leq m \leq M - 1$. Since a state is defined as $x_k = x_k^L + x_k^{L+1}$, an equivalent notation is

$$\Psi_M' = a_k^{L+1}$$

and

$$\Psi_M'' = a_k^{L+1}$$

where

$$a_k^{L+1} \neq a_k^{L+1}, \quad \text{for } 1 \leq m \leq M - 1$$

and

$$a_k^{L+1} = a_k^{L+1}, \quad \text{for all other } m$$

A. Binary Hypothesis Error Probability

The binary hypothesis error probability $P_2(\varepsilon_M)$ is the probability of detecting $\Psi_M'$ when $\Psi_M'$ is the true path. This error occurs when the accumulated metric (25) along the path $\Psi_M'$ is greater than the accumulated metric along $\Psi_M''$, i.e.,

$$P_2(\varepsilon_M) = \sum_{m=1}^{M} \mathcal{M}_{ML} \left( \frac{b_{k+m-L} - a_{k+m-L}}{\sigma} \right) \mathcal{M}_{ML}^{(-b_{k+m-L} - a_{k+m-L})} \left( \frac{b_{k+m-L} - a_{k+m-L}}{\sigma} \right)$$

(29)

Notice that we can use an arbitrary $k$ in (29). For convenience, we shall use $k = L$ throughout this subsection.

To evaluate the expression on the right-hand side of (29), we use the special structural properties of Gauss–Markov covariance matrices [19] and rewrite (29) as

$$P_2(\varepsilon_M) = \sum_{m=1}^{M} \mathcal{M}_{ML} \left( \frac{b_{k+m-L} - a_{k+m-L}}{\sigma} \right) \mathcal{M}_{ML}^{(-b_{k+m-L} - a_{k+m-L})} \left( \frac{b_{k+m-L} - a_{k+m-L}}{\sigma} \right)$$

(30)

The matrices $C_{\Psi_M'}$ and $C_{\Psi_M''}$ are the covariance matrices of $x_{k+1}^{L+M}$ when $a'_1, \cdots, a'_m$ and $a''_1, \cdots, a''_m$ are the transmitted (or written on a memory medium) binary sequences, respectively. The innovation vector $X_{k+1}^{L+M}$ is defined as

$$X_{k+1}^{L+M} = x_{k+1}^{L+M} - y'$$

(31)

where the values $y'(a''_1) \cdots y'(a''_m)$ are the same as those used in the model equation (5). The vectors $X_{k+1}^{L+M}$ and $y''$ are defined similarly as in (31), where the superscript ‘$'$’ is replaced by ‘$''$. Next, we briefly describe how to obtain the inverses $C_{\Psi_M'}^{-1}$ and $C_{\Psi_M''}^{-1}$ of the covariance matrices from the model parameters in (6). Since the process $n_k$ is conditionally Gauss–Markov with Markov memory

\[1\]
length \( L \), it follows from [15] that \( C_{\varphi_M}^{-1} \) is an \( L \)-banded matrix. Its upper Cholesky decomposition is given by
\[
C_{\varphi_M}^{-1} = U_{\varphi_M}^T \mathbf{D}_{\varphi_M} U_{\varphi_M}.
\]
(32)

In (32), according to [19] and [25], the upper triangular \( L \)-banded matrix \( U_{\varphi_M} \) is
\[
U_{\varphi_M} = \begin{bmatrix}
U_{11} & 0 \\
-\frac{1}{\sigma^2} (d_{11}^{L-1}) & 1 \\
& & \ddots \\
& & & \frac{1}{\sigma^2} (d_{L-11}^{L-1}) \\
0 & 0 & \cdots & 1
\end{bmatrix}
\]
(33)

and the diagonal matrix \( \mathbf{D}_{\varphi_M} \) is
\[
\mathbf{D}_{\varphi_M} = \text{diag} \left[ 1/\sigma^2 (d_{11}^{L-1}), \ldots, 1/\sigma^2 (d_{L-11}^{L-1}) \right].
\]
(34)

The values \( b(d_{ij}^{L-1}) \) and \( \sigma (d_{ij}^{L-1}) \) in (33) and (34) are the same as those in the model equation (6). The upper triangular matrix \( U_{11} \) and the diagonal matrix \( \mathbf{D}_{11} \) are the upper Cholesky factors of \( C_{\varphi_M}^{-1} = U_{11} \mathbf{D}_{11} U_{11}^T \), where \( C_{11} \) is the conditional covariance matrix of \( z_i \) when the sequence of transmitted (written) symbols is \( a_{i-1} \).

The elements of \( C_{\varphi_M}^{-1} \) can be obtained by rearranging a signal-dependent set of Yule–Walker equations and solving a linear system of equations. Since computing \( C_{\varphi_M}^{-1} \) is tedious but straightforward, we refer the reader to an example provided in [26]. The Cholesky decomposition of \( C_{\varphi_M}^{-1} \) is obtained similarly.

Once we have computed the Cholesky decompositions (32) of both \( C_{\varphi_M}^{-1} \) and \( C_{\varphi_M}^{-1} \), we can rewrite the binary hypothesis error probability for the error event \( \varepsilon_M = (\Psi_M', \Psi_M'') \) as
\[
P_2(\varepsilon_M) = P \left\{ \bar{w}_T \bar{w}^T > 0 \right\} \det \Sigma_{\varepsilon M} - (\bar{w} - \bar{\mu}_M)^T \Sigma_{\varepsilon M}^{-1} (\bar{w} - \bar{\mu}_M) \}
\]
(35)

where \( \bar{w} \) is an \( (L + M) \times 1 \) Gaussian random vector with \( E[\bar{w}] = 0 \) and \( E[\bar{w}_T \bar{w}^T] = \mathbf{I} \). The vector \( \bar{\mu}_M \) is given by
\[
\bar{\mu}_M = QD_{\varphi_M}^{1/2} U_{\varphi_M} (y'' - y') = [d \ 0 \ \cdots \ 0]^T
\]
(36)

where \( d = ||D_{\varphi_M}^{1/2} U_{\varphi_M} (y'' - y')|| \), \( y' \) and \( y'' \) are defined in (31), and \( || \cdot || \) is the \( L_2 \) vector norm. \( Q \) is a unitary matrix (for example, the Householder reflector \( Q = 2v v^T / ||v||^2 - I \), where \( v = \bar{w} / ||\bar{w}|| + \varepsilon_1 \), \( \varepsilon_1 = [1, 0, \ldots, 0]^T \), see [27]), and \( \Sigma_{\varepsilon_M} \) satisfies
\[
\Sigma_{\varepsilon M}^{-1} = QD_{\varphi M}^{1/2} U_{\varphi M} C_{\varphi M}^{-1} U_{\varphi M}^T D_{\varphi M}^{1/2} Q^T
\]
and
\[
\det \Sigma_{\varepsilon M} = \det D_{\varphi M} \det C_{\varphi M} \det \left( U_{\varphi M}^T \right)
\]
(37)

In the general case, the expression on the right-hand side of (35) cannot be simplified by integrating a \( \chi^2 \)-like distribution. These simplifications are possible only in special cases, for example, when \( \Sigma_{\varepsilon M} \) is diagonal (for a list of all special cases where a simplification is possible, see [28]). In the general case, however, evaluation of (35) involves integrating a zero-mean \( I \)-covariance multivariate Gaussian pdf in the region where the condition on the right-hand side of (35) holds. Since this is a region outlined by a quadratic form, its shape must be determined on a case-by-case basis, which ultimately may be computationally impractical. A brute-force alternative is to evaluate (35) numerically by Monte Carlo, but this is also computationally intensive for low error rate (high signal-to-noise ratio (SNR)) scenarios. In the next subsection, we develop a computationally efficient \( Q \)-function approximation that leads to an accurate evaluation of (35) for high SNR scenarios where Monte Carlo numeric evaluations are not viable.

B. Approximate Binary Hypothesis Error Probability

Computing (35) requires integrating a zero-mean \( I \)-covariance multivariate Gaussian pdf over a region where
\[
f_{\varepsilon M}(w) = w^T \bar{w} - \ln \det \Sigma_{\varepsilon M} - (\bar{w} - \bar{\mu}_M)^T \Sigma_{\varepsilon M}^{-1} (\bar{w} - \bar{\mu}_M) > 0.
\]
(38)

Since a Gaussian function has a very fast decay, we approximate (35) by
\[
P_2(\varepsilon_M) \approx Q(d_2)
\]
(39)

where \( d_2 \) is the point on \( f_{\varepsilon M}(w) = 0 \), closest to the origin and
\[
Q(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-(t/2)^2} dt.
\]

To determine \( d_2 \), one may use any minimization technique (Lagrange multipliers, steepest descent, etc.). A particularly fast method is to find the point \( d_1 \) on the boundary \( f_{\varepsilon M}(w) = 0 \), between 0 and \( \bar{\mu}_M \) by solving a quadratic equation, see Fig. 3. For a given matrix \( \Sigma_{\varepsilon M} \), the point \( d_1 \), will be between 0 and \( \bar{\mu}_M \) if \( d_2 \) is large enough, i.e., if the SNR is large enough. Since \( d_2 \) is typically close to \( d_1 \), we can obtain \( d_2 \) in a few iterations, or we may simply set \( d_2 \approx d_1 \cos \theta \), see Fig. 3. We prefer the iterative method since it is not too computationally expensive.

C. Upper Bound

The bit-error probability \( P_1 \) can be upper-bounded by the union bound [2]. Since our channel is asymmetric (nonlinear with signal-dependent noise correlation) we need a union bound expression that generalizes the one in [2]. One can verify that the two expressions are equivalent for linear-symmetric channels. Let \( M \) denote the length (in branches) of an error event \( \varepsilon_M = (\Psi_M', \Psi_M'') \). For our binary channel, the minimum allowable error event length can be verified to be \( M_{\text{min}} = I + L + 1 \), where \( I \) is the ISI length, and \( L \) is the Markov memory length of noise. The set of all allowable error events of length \( M \) is denoted by \( E_M \). Denote by \( b_d(\varepsilon_M) \) the number of erroneous bits...
corresponding to the error event \( \varepsilon_M \). Let \( P(x) \) be the a priori probability of the trellis state \( x \). Similarly, let \( P(x_k, x_{k-1}) \) denote the conditional probability of transition from state \( x_{k-1} \) to state \( x_k \). The binary hypothesis error probability associated with the error event \( \varepsilon_M \) is \( P_2(\varepsilon_M) \) and can be computed by either Monte Carlo simulation or approximated by (39). The upper bound on bit-error probability is given by

\[
P_b \leq \sum_{M=M_{\min}}^{M_{\max}} \sum_{\Psi_M} \left[ P(x_k) \prod_{i=1}^{M} P(x_{k+i-1} | x_{k+i-1}) \right] \sum_{\Psi_M' \neq \Psi_M} b_\#(\varepsilon_M) P_2(\varepsilon_M).
\]

This bound accounts for both the ML and the MAP sequence detector, where the corresponding binary hypothesis error probability \( P_2(\varepsilon_M) \) needs to be substituted (see footnote to (29)). If all symbol sequences are equally likely, then \( P(x_k) = 2^{-i} \) and \( P(x_{k+i-1} | x_{k+i-1}) = 1/2 \).

A practical method for evaluating the bound in (40) is to truncate the first sum to values of \( M < M_{\max} \), where \( M_{\max} \) is a predetermined large enough constant. Notice that in (40), the binary hypothesis error probability \( P_2(\varepsilon_M) \) needs to be determined for every qualifying error event \( \varepsilon_M \), which makes the bound computationally expensive if \( P_2(\varepsilon_M) \) is computed by Monte Carlo simulations rather than by the approximation (39).

**D. Lower Bound**

Obviously, a lower bound is obtained by picking any single term of the union bound (40). This, however, may not be a tight bound. To get a tighter bound, we modify the genie-assisted bound [24].

**Modified Genie-Assisted Bound:** Denote by \( P(\varepsilon) \) the probability of an error event. Let \( \pi_i(P_e) \) be the probability that the input sequence \( a_k \) is such that there is a probability \( P \geq P_e \) of confusion with some allowable sequence \( a_k' \). Even with a genie-assisted detector that has to choose only between \( a_k \) and \( a_k' \), the probability of an error event \( P(\varepsilon) \) will be greater than \( \pi_i(P_e) \). Clearly, then

\[
P(\varepsilon) \geq \max_{0 \leq P \leq 1} \left[ \pi_i(P_e) \cdot P \right].
\]

The difference between this formulation and the one in [24] is that we assume \( P \geq P_e \), instead of \( P = P_e \). In fact, the bound in (41) will never be looser (and in many cases tighter) than the one in [24], even in linear-symmetric channels.

We use the result in (41) to find a lower bound for the bit-error probability. Consider only error events of length \( M_{\min} = I + L + 1 \). The number of these events for binary signaling is

\[
N = 2^{L+L+M_{\min}} = 2 \cdot 4^{L+L}.
\]

Label these error events as \( \varepsilon_{M_{\min}}(1), \varepsilon_{M_{\min}}(2), \ldots, \varepsilon_{M_{\min}}(N) \), where their order is such that

\[
P_2(\varepsilon_{M_{\min}}(1)) \geq P_2(\varepsilon_{M_{\min}}(2)) \geq \cdots \geq P_2(\varepsilon_{M_{\min}}(N))
\]

and \( P_2(\varepsilon_{M_{\min}}(i)) \) is the binary error probability of the error event \( \varepsilon_{M_{\min}}(i) \). The probability \( P_2(\varepsilon_{M_{\min}}(i)) \) can be computed either through Monte Carlo simulation or with the approximation in (39). In the binary signaling case, for every sequence \( \Psi_{M_{\min}}(i) \) of \( M_{\min} + 1 \) states, there exists only one erroneous sequence \( \Psi_{M_{\min}}(i) \). We therefore have

\[
\sum_{i=1}^{N} P(\Psi_{M_{\min}}(i)) = 1
\]

i.e., \( \Psi_{M_{\min}}(i) \) are disjoint sequences. If we choose \( P_e = P_2(\varepsilon_{M_{\min}}(i)) \) for some \( i \leq i \leq M_{\min} \), then

\[
\pi_i(P_e) \geq \sum_{j=1}^{i} P(\Psi_{M_{\min}}(j))
\]

because there may be error events of length \( M > M_{\min} \) whose binary hypothesis error probability is greater than \( P_e \). In the interest of keeping the bound computationally simple, we ignore these error events of length \( M > M_{\min} \). Since for all error events of length \( M_{\min} \), the number of erroneous bits is \( b_\#(\varepsilon_{M_{\min}}(i)) = 1 \), we have

\[
P_b \geq \pi_i(P_e) \cdot P_e \geq \sum_{j=1}^{i} P(\Psi_{M_{\min}}(j)) \cdot P_2(\varepsilon_{M_{\min}}(i))
\]

for any \( 1 \leq i \leq N \). Maximizing the right-hand side of (42), we obtain a tighter lower bound

\[
P_b \geq \max_{1 \leq i \leq N} \left\{ \sum_{j=1}^{i} P(\Psi_{M_{\min}}(j)) \cdot P_2(\varepsilon_{M_{\min}}(i)) \right\}.
\]
Thus the modified genie-assisted bound provides the lower bit-error probability bound

\[ P_b \leq 2^{\frac{1}{2} \sum_{t=0}^{L} \max_{1 \leq i \leq N} \{ \epsilon_{B}(\epsilon_{M}(i)) \} } \]  

Since the terms \( P_2(\epsilon_{M}(i)) \) have already been computed as part of (40), no further computation is needed for the lower bound. Notice also that while (44) is always a lower bound, it may not be the tightest of the modified genie-assisted bounds because we considered only the minimum-length error events. The following example shows, however, that this bound is relatively tight.

### E. Performance Evaluation Example

We study the performance of the maximum-likelihood sequence detector for a simple channel with ISI length \( I = 1 \) and Markov memory length \( L = 1 \). The channel is given by the model equations (5) and (6), also depicted in Fig. 1, where the values of the parameters are given in Table I. The channel represented in Table I can be viewed as a nonlinearly deviated \( 1 - D \) partial response channel with signal-dependent correlated noise.

Fig. 4 shows the performance of the ML sequence detector. The different values of the SNR shown in Fig. 4 are obtained by scaling the noiseless channel response (column denoted by \( y_{\epsilon M}(i) \)) in Table I while keeping the relative distance between the signal points the same. This corresponds to scaling the signal power while the noise constellation is kept the same. In Fig. 4, the solid line represents the simulated performance of the ML sequence detector (MLSD). We do not show the MLSD performance curve beyond SNR = 14 dB since the Monte Carlo simulations become expensive to compute. Fig. 4 also shows the simulated performance of the Euclidean detector applied to the channel in Table I. The Euclidean detector is a Viterbi detector that assumes the noise to be white. Fig. 4 shows that at the error rate of \( 10^{-5} \) the MLSD has a gain of 2.5 dB over the Viterbi Euclidean detector.

Fig. 4 also plots the bit-error probability bounds. The stars (*) in Fig. 4 show the bounds computed by evaluating the binary hypothesis error probability with Monte Carlo simulations. As the Monte Carlo simulations become too computationally intensive beyond SNR = 12 dB, we only show the stars (*) for SNR \( \leq 12 \) dB. On the other hand, the approximate bounds (dashed lines in Fig. 4) computed by approximating the binary hypothesis error probability with a Q-function, see (39), can be computed at a fractional computational cost for any value of the SNR. Fig. 4 also shows that the Q-function approximation is extremely close to the bounds computed by Monte Carlo simulations. As expected, the bounds are tightening as the SNR increases which helps us evaluate the MLSD performance at high SNR’s where Monte Carlo simulations of the MLSD are impractical. We point out that at extremely high error rates \( (10^{-1} \text{ and above}) \) the upper bound may fail to converge, but that is rarely a concern since at those error rates the MLSD performance can readily be evaluated through simulations.

### VI. Suboptimal Detectors

Being the optimal detector, the maximum-likelihood sequence detector (MLSD) bounds from below the error rate of any suboptimal detector of lower complexity. Furthermore, the structure of the signal-dependent MLSD gives new insight into developing suboptimal detectors. Here we cover several suboptimal schemes and discuss their relationship to the MLSD.

From Fig. 4 we see that there is a performance region between the MLSD and the Euclidean detector that can be filled with suboptimal detectors with complexities possibly lower than that of the MLSD. There are many ways to construct such suboptimal receivers. We limit ourselves to suboptimal receivers with a certain structure. We look for suboptimal receivers that are Viterbi-like, i.e., that operate on a trellis and use branch/path metric minimization as their detection strategy. We define the complexity of a Viterbi-like detector as the number of multiplications it has to perform per clock interval (symbol interval) in order to compute all its branch metrics. We ignore additions. In many realizations of the Viterbi detector this may not be an accurate representation of the detector complexity since there exist several practical finite-precision methods to simplify the multiplication operation. Nevertheless, for simplicity, we trust that the multiplications count provides at least a reasonable representation for the complexity of a detector.

When \( I \) is the ISI length and \( L \) the Markov memory length, the MLSD for the binary channel model in (5) and (6) has \( 2^{I+L} \) states and \( 2^{I+L+1} \) branches. Thereby, \( 2^{I+L+1} \) branches share a single FIR filter used to compute the branch metric, see Fig. 2. In Fig. 2, if we count each tap-multiplication by \( b_i(\epsilon_M^{-1}(k)) \), each division by \( \sigma(\epsilon_M^{-1}(k)) \), and each squaring operation as one multiplication, the count of multiplications needed to compute all the branch metrics of the MLSD in one clock interval is \( 2I + L + 1 \). Since the computational complexity of the Euclidean Viterbi detector is \( 2^{I+L+1} \) per clock interval, we have that the computational complexity increase factor of the MLSD over the Euclidean detector is

\[ F_{\text{MLSD}}(L) = 2^L + L + 1. \]  

We next consider suboptimal detectors with lower computational increase factors.

#### A. Low-Order Autoregressive (AR) Approximation (Q-AR Detection)

We introduce in this subsection a new suboptimal detector that we refer to as the low-order Q-AR detector.

A Gauss–Markov noise process of order \( L \) is the output of an autoregressive (AR) model of order \( L \). The inverse covariance matrix of an \( L \)-th-order Gauss–Markov process is \( L \)-banded, meaning that it has \( 2L + 1 \) nonzero diagonals, while the remaining are all-zero diagonals [15]. We obtain a suboptimal detector by using an AR approximation [16] of order \( Q \leq L \). Let \( C_Q \) be the covariance matrix of the \( Q \)-th-order AR process approximation. The matrix \( C_{Q-1} \) is then \( Q \)-banded. We can show that the \( Q \)-banded matrix \( C_{Q-1} \) that minimizes the Kullback–Leibler mean information loss due to approximating the original \( L \)-th-order AR process with a \( Q \)-th-order AR process is given by the following decomposition [19]:

<table>
<thead>
<tr>
<th>( k-1 )</th>
<th>( y(k-1) )</th>
<th>( \sigma(k-1) )</th>
<th>( b_1(k-1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>-0.1</td>
<td>1.0</td>
<td>-0.2</td>
</tr>
<tr>
<td>01</td>
<td>1.2</td>
<td>1.8</td>
<td>-0.6</td>
</tr>
<tr>
<td>10</td>
<td>-0.8</td>
<td>1.2</td>
<td>-0.6</td>
</tr>
<tr>
<td>11</td>
<td>0.1</td>
<td>1.0</td>
<td>-0.2</td>
</tr>
</tbody>
</table>
A th-order filter is
\[
C = C_Q + \begin{bmatrix}
\Delta \\
\Delta^T
\end{bmatrix}
\begin{bmatrix}
0_{2Q+1} \\
D_{2Q+1}
\end{bmatrix}
\]
and
\[
C_Q^{-1} = \begin{bmatrix}
0 & D_{2Q+1} \\
0 & 0
\end{bmatrix}
\]  \hspace{1cm} \text{(46)}
where \(0_{2Q+1}\) denotes a band of \(2Q+1\) all-zero diagonals, and \(D_{2Q+1}\) denotes a band of \(2Q+1\) nonzero diagonals. Note that the decomposition in (46) is unique since there exists only one matrix \(C_Q\) such that the \(Q\)-band of \(C_Q\) equals the \(Q\)-band of \(C\), and that \(C_Q^{-1}\) is \(Q\)-banded. The suboptimal Viterbi detector based on the low-order \(Q\)-AR approximation computes the branch metric using a \(Q\)-tap FIR filter instead of an \(L\)-tap FIR filter. Consequently, the detector has \(2^{2Q+Q}\) states and \(2^{2Q+Q+1}\) branches where \(2^{2Q+1}\) branches share a single \(Q\)-tap FIR filter. The computational complexity increase factor of the low-order \(Q\)-AR detector over the Euclidean Viterbi detector is
\[
F_{Q-AR}(Q) = 2^Q + Q + 1.
\]  \hspace{1cm} \text{(47)}
Obviously, if \(Q = L\), we have the MLSD. If \(Q = 0\), we obtain the suboptimal detector presented in [7] and [8] for which the complexity increase factor is \(F_{Q-AR}(0) = 2\).

B. Block Partitioning (K-Skip Detection)

In [13], Altekar and Wolf present a suboptimal detector by replacing the covariance matrix \(C\) by a matrix \(C'\) obtained by keeping only the \(K \times K\) block diagonal elements of \(C\), i.e.,
\[
C \approx \begin{bmatrix}
R & 0 \\
0 & R
\end{bmatrix} = C'
\]  \hspace{1cm} \text{(48)}
where \(R\) is a \(K \times K\) principal minor of \(C\). This contrasts with the \(Q\)-AR approximation of (46) where it is the inverse covariance that is banded rather than the covariance matrix itself, which is block-partitioned. In [13], the method is called the \(K\)-step Viterbi detector. In [13], the authors provide a metric computation method involving vectors of observed samples and covariance matrices \(R\). Using techniques presented in Section IV-B, we can show that the \(K\)-step detector may also be implemented with low-order FIR filters (which is also a computationally less expensive implementation than in [13]). However, the order \(Q\) of the FIR filters is cyclicly time-varying. For example, the three-step detector would use banks of \(2^{2+1}\) \(Q = 0\)-FIR filters for the first stage of the trellis, \((Q = 1)\)-FIR filters for the second stage, and \((Q = 2)\)-FIR filters for the third stage (for low-order signal-dependent AR processes, see [16] and [19]). In the fourth stage of the trellis, the detector goes back to the bank of \((Q = 0)\)-FIR filters and repeats the cycle. Thus generally, a \(K\)-step detector would cyclicly employ \((Q = 0)\)-AR through \((Q = K - 1)\)-AR approximations at different stages of the trellis. Thereby, the FIR filters used to calculate the branch metrics are the exact inverses of the \(Q\)-AR filter approximations, see Section IV-B. At trellis stages where \(Q \geq L\), the bank of FIR filters are the inverses of the bank of \(L\)-AR filters used in the channel model equations (5) and (6). Since the \(K\)-step detector uses nonuniform AR approximations at different trellis stages, the number of multiplications has to be averaged over a cycle of \(K\) trellis stages to get the average computational complexity per clock interval. The computational complexity increase factor of the \(K\)-step Viterbi detector over a Euclidean Viterbi detector is
\[
F_{K-skip}(K) = \begin{cases}
\frac{2^K - 1}{K} + \frac{K + 1}{2}, & \text{for } K \leq L \\
\frac{2^K - 1 + (K - L)(2^L + L - 1)}{K} + \frac{L(L + 1)}{2K}, & \text{for } K > L.
\end{cases}
\]  \hspace{1cm} \text{(49)}
Clearly, the \((K = 1)\)-step detector is the same as the \((Q = 0)\)-AR detector, which is the detector presented in [7] and [8].

C. Block Skipping (S-Skip Detection)

A drawback of the \(K\)-step detector is that it reaches the MLSD performance only in the limit as \(K \rightarrow \infty\). We propose here a new class of suboptimal detectors that reaches the MLSD performance for a finite order of suboptimality. The following equation shows the \(L\)-band of a covariance matrix \(C\) of an \(L\)-order Gauss–Markov process partitioned in contiguous principal blocks of size \((L+1) \times (L+1)\)
\[
C = \begin{bmatrix}
R & 0 & \cdots & 0 \\
0 & R & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & R
\end{bmatrix}
\]  \hspace{1cm} \text{(50)}
It can be shown, see for example [19], that the elements of the \(L\)-band of \(C\) determine in a unique way the covariance matrix \(C\) of the \(L\)-th order Gauss–Markov process, in particular, they determine the elements of the blocks \(\Delta\). We can conclude that the MLSD derives the branch metrics based on all covariance coefficients in this \(L\)-band of \(C\). As an alternative, out of the \((L + 1) \times (L + 1)\) blocks outlined on the right-hand side of (50), a suboptimal algorithm may ignore the blocks outlined by the dashed lines in (50). We refer to such a procedure as the \((S = 1)\)-skip detector. In general, if we skip \(S\) blocks at a time, we have the \(S\)-skip detector. Much like detailed in Section VI-B, the \(S\)-skip detector may be designed through cyclic implementation of nonuniform \(Q\)-AR approximations. Each cycle spans \(S + 1\) trellis stages and the detector cyclicly employs banks of \((Q = L - S)\)-FIR, \((Q = L - S + 1)\)-FIR, …, \((Q = L)\)-FIR filters. The bank of \((Q = L)\)-FIR filters is given by the inverses of the actual \(L\)-th order AR filters in (5) and (6). The computational complexity increase factor of the \(S\)-skip Viterbi detector over the Euclidean Viterbi detector is
\[
F_{S-skip}(S) = \frac{2^{L+1} - 2^{L-S}}{S + 1} + \frac{2L - S + 2}{2}, \hspace{1cm} \text{for } 1 \leq S \leq L.
\]  \hspace{1cm} \text{(51)}
The \((S = 0)\)-skip detector is clearly the MLSD. It is also apparent that the \((S = L)\)-skip detector is equivalent to the \(K\)-step block partitioning detector when \(K = L + 1\), where \(L\) is the Markov memory length of the noise.

D. Nonuniform AR Approximations

The \(K\)-step and \(S\)-skip detectors are subclasses of Viterbi detectors that apply different \(Q\)-th order AR approximations at different stages of
the trellis. One can easily think of numerous other Viterbi-like detectors based on this concept, where the different orders of AR approximations are not necessarily cyclicly employed. A typical scenario would be in nonstationary detection. Suppose that at some point during data transmission the data rate increases, i.e., the clock interval (symbol interval) shrinks. If the new clock interval is too short to perform Viterbi detection based on a high-order AR approximation, the detector may resort to a less computationally expensive lower order AR approximation, still retaining the communication link at the price of a less reliable communication. Combinations of different orders of AR approximations will lead to a large number of possible detector designs.

### E. Decision Feedback

Decision feedback mechanisms can be used to lower the computational complexity of Viterbi detectors. Reduced-state sequence estimators (RSE) [29] use premature decisions to reduce the ISI length and thus lower the number of states. Noise predictors [11], [12], [30] on the other hand, use premature decisions to guess the future noise sample. In the context of Markov noise models, this corresponds to lowering the Markov memory length. Hybrids between reduced-state sequence estimators and noise predictors are also possible. Furthermore, we can envision combining the nonuniform AR approximations with decision feedback strategies to form a larger subclass of Viterbi-like suboptimal detectors. A common feature to all decision feedback suboptimal detectors is error propagation, see results presented within [13]. Error propagation precludes the use of decision feedback detectors in many applications where larger blocks of errors cannot be tolerated, e.g., in block error-correction coding/decoding.

### F. Comparative Results

In this subsection we provide the error performance for a number of detectors and show that the $Q$-AR detectors provide usually the best compromise between error rate and computational complexity. Table II shows the parameters (for explanations, see (5) and (6)) of a very noisy real channel encountered in magnetic recording applications [16]. The channel in Table II is a nonlinearly distorted class-4 partial response (PR4) channel with ISI length $J = 2$ and signal-dependent correlated noise with Markov memory length $L = 3$. Fig. 5 shows for this channel the error rate versus the computational complexity increase factor $F$ for a number of detectors: the MLSD, the Euclidean Viterbi decoder, and several suboptimal $Q$-AR, $K$-step, and $S$-skip detectors. The computational complexity increase factors are normalized with respect to that of the Euclidean detector, whose operating point in Fig. 5 corresponds then to the value 1 on the horizontal axis. We do not consider decision feedback suboptimal detectors since they suffer from error propagation. The error rates were computed by Monte Carlo simulations with $10^5$ bits. The results confirm, as mentioned above, that the suboptimal detectors based on low-order AR approximations ($Q$-AR detectors) provide the best complexity-versus-error-rate tradeoff. The $K$-step detector is a reasonable alternative to the $Q$-AR detector only for low complexities and high error rates. On the other hand, the $S$-skip detector is a reasonable alternative to the $Q$-AR detector only for high complexities and low error rates.

### VII. Summary

We studied the maximum-likelihood sequence detector (MLSD) and the MAP sequence detector for intersymbol interference (ISI) channels with correlated noise whose statistics are possibly signal-dependent. We describe three major contributions to the problem. First, by modeling the correlated noise as a Markov process of finite order, we derive the optimal MLSD and MAP sequence detector as extensions of the Viterbi decoder for ISI channels with finite memory noise. Second, we provide an error performance study of the MLSD and MAP sequence detector, deriving tight upper and lower bounds for the probability of error. Finally, we present several classes of suboptimal sequence detectors, including new algorithms, and compare their computational complexity and their error performance.

The key to deriving a finite implementation for the optimal sequence detector is our modeling the correlated noise as a finite-order Markov process, see Section II. We showed in Section III that, as long as the memory of the noise is Markov, we can extend the Viterbi algorithm

<table>
<thead>
<tr>
<th>$\sigma_{k-2}$</th>
<th>$\mu_{k-2}$</th>
<th>$\beta_{k-2}$</th>
<th>$\gamma_{k-2}$</th>
<th>$\delta_{k-2}$</th>
<th>$\epsilon_{k-2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>-0.12</td>
<td>0.10</td>
<td>-0.3</td>
<td>-0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>001</td>
<td>0.8</td>
<td>0.20</td>
<td>-0.4</td>
<td>-0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>010</td>
<td>0.0</td>
<td>0.12</td>
<td>-0.2</td>
<td>-0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>011</td>
<td>0.8</td>
<td>0.17</td>
<td>-0.1</td>
<td>-0.3</td>
<td>-0.1</td>
</tr>
<tr>
<td>100</td>
<td>-0.62</td>
<td>0.11</td>
<td>-0.4</td>
<td>-0.2</td>
<td>-0.2</td>
</tr>
<tr>
<td>101</td>
<td>-0.12</td>
<td>0.14</td>
<td>-0.6</td>
<td>-0.5</td>
<td>-0.2</td>
</tr>
<tr>
<td>110</td>
<td>-0.62</td>
<td>0.10</td>
<td>-0.6</td>
<td>-0.4</td>
<td>-0.2</td>
</tr>
<tr>
<td>111</td>
<td>-0.12</td>
<td>0.12</td>
<td>-0.3</td>
<td>-0.1</td>
<td>0.0</td>
</tr>
</tbody>
</table>
to implement the maximum a posteriori (MAP) detector and the maximum-likelihood sequence detector (MLSD). The complexity of the MLSD is exponentially proportional to the sum of the ISI length $I$ plus the Markovian noise memory length $L$. The branch metrics of the detector are functions of $L + 1$ consecutive samples of the observable process. When the noise is Gaussian with Markovian memory (signal-dependent Gauss–Markov noise), see Section IV, the branch metrics are expressed in terms of the conditional second-order statistics of the noise. In the binary signaling case, there are $2^{I+L}$ states in the trellis and $2^{I+L+1}$ branches in each stage of the trellis. We presented an efficient FIR filter based computation of the branch metrics using a bank of $2^{I+L}$ 1-th-order FIR filters. The structure of the receiver resembles the structure of the detector in the memoryless scenario, in which we preprocess the received sequence to uncorrelate the noise and then apply the Viterbi detection. In the case of noise with memory, however, the preprocessing cannot be done up-front—it needs to be implemented separately in every branch of the Viterbi trellis. The constraint of finite ISI length requires that we use only FIR filters for the preprocessing, which in turn limits us to Markov (i.e., autoregressive) channel noise only. When the channel noise memory is not Markov, we find the best fitting Markov process to describe it and then apply the Viterbi algorithm to asymptotically reach optimality. In nonstationary environments, adaptive tracking of the conditional covariance statistics is required since the branch metrics depend on these statistics.

We presented in Section V the error analysis in terms of bounds on the probability of bit errors. While formally similar to the analysis for white and signal-independent noise, the bounds are computationally more complex due to the signal-dependent noise correlation asymmetry in the channel. We simplified the computation of the bounds by developing a method that exploits the banded structure of the inverses of Gauss–Markov covariance matrices. We have further simplified the bound computations using $Q$-function approximations. A simulation study was carried out showing the good quality of the bounds as predictors of the error performance. As expected, the bounds are tighter at higher values of the signal-to-noise ratio (SNR).

We use the structure of the extended Viterbi detector for Markovian memory channels as a guideline to derive new suboptimal detectors that are computationally simpler than the MLSD. We also showed how several suboptimal algorithms proposed in the literature are related to the MLSD. In Section VI, low-order autoregressive (AR) approximations were utilized to yield a whole class of suboptimal sequence detectors. An experimental study was carried out to show that, out of all these suboptimal methods, the strategy that provides the best complexity-versus-performance tradeoff is the one based on fully utilizing a $Q$-band of the noise covariance matrix, i.e., the $Q$-AR detector. We have shown how to exploit the banded form of the inverse of Markov covariance matrices to construct suboptimal receivers that asymptotically reach optimality with higher order approximations and that do not suffer from error propagation problems.

**REFERENCES**


