

Convolutional-Code-Specific CRC Code Design

Chung-Yu Lou, *Student Member, IEEE*, Babak Daneshrad, *Member, IEEE*,
and Richard D. Wesel, *Senior Member, IEEE*

Abstract—Cyclic redundancy check (CRC) codes check if a codeword is correctly received. This paper presents an algorithm to design CRC codes that are optimized for the code-specific error behavior of a specified feedforward convolutional code. The algorithm utilizes two distinct approaches to computing undetected error probability of a CRC code used with a specific convolutional code. The first approach enumerates the error patterns of the convolutional code and tests if each of them is detectable. The second approach reduces complexity significantly by exploiting the equivalence of the undetected error probability to the frame error rate of an equivalent catastrophic convolutional code. The error events of the equivalent convolutional code are exactly the undetectable errors for the original concatenation of CRC and convolutional codes. This simplifies the computation because error patterns do not need to be individually checked for detectability. As an example, we optimize CRC codes for a commonly used 64-state convolutional code for information length $k=1024$ demonstrating significant reduction in undetected error probability compared to the existing CRC codes with the same degrees. For a fixed target undetected error probability, the optimized CRC codes typically require 2 fewer bits.

Index Terms—catastrophic code, convolutional code, cyclic redundancy check (CRC) code, undetected error probability.

I. INTRODUCTION

ERROR-detecting codes and error-correcting codes work together to guarantee a reliable link. The inner error-correcting code tries to correct any errors caused by the channel. If the outer error-detecting code detects any residual errors, then the receiver will declare a failed transmission.

Undetected errors result when an erroneously decoded codeword of the inner code has a message that is a valid codeword of the outer code. This paper designs cyclic redundancy check (CRC) codes for a given feedforward convolutional code such that the undetected error probability is minimized.

A. Background and Previous Work

A necessary condition for a good joint design of error-detecting and error-correcting codes, both using linear block codes, is provided in [1]. However, this condition is based on the minimum distances of the inner and outer codes and does not consider the detailed code structure.

Most prior work on CRC design ignores the inner code structure by assuming that the CRC code is essentially operating on a binary symmetric channel (BSC). We refer to this as the BSC assumption. The BSC assumption does not take

advantage of the fact that the CRC code will only encounter error sequences that are valid codewords of the inner code.

The undetected error probability of a CRC code under the BSC assumption was evaluated using the weight enumerator of its dual code in [2]. Fast algorithms to calculate dominant weight spectrum and undetected error probability of CRC codes under the BSC assumption were presented in [3], [4].

In [5], Koopman and Chakravarty list all standard and good (under the BSC assumption) CRC codes with up to 16 parity bits for information lengths up to 2048 bits. The authors recommend CRC codes given the specific target redundancy length and information length.

For more than 16 CRC parity bits, it is difficult to search all possibilities and find the best CRC codes even under the BSC assumption. Some classes of CRC codes with 24 and 32 bits were investigated under the BSC assumption in [6] and an exhaustive search for 32-bit CRC codes under the BSC assumption was performed later in [7].

Because all of these designs ignore the inner code by assuming a BSC, there is no guarantee of optimality when these CRC codes are used with a specific inner code.

A few papers do consider CRC and convolutional codes together. In [8], the CRC code was jointly decoded with the convolutional code and used to detect the message length without much degradation of its error detection capability. In [9], CRC bits were punctured to reach higher code rates. The authors noticed that bursty bit errors caused an impact on the performance of the punctured CRC code.

Recently, [10] and [11] have considered a CRC code used for error correction jointly with convolutional and turbo codes, respectively. However, in these cases, the error detection capability of the CRC code is degraded. Such undetected error probability as well as false alarm probability were analyzed in [12] under the BSC assumption. The authors of [12] also modeled the bursty error at the turbo decoder output using a Gilbert-Elliott channel.

B. Main Contributions

We propose two methods to compute the undetected error probability of a CRC code concatenated with a feedforward convolutional code. The *exclusion* method enumerates possible error patterns of the inner code and excludes them one by one if they are detectable. The *construction* method constructs a new convolutional code whose error events correspond exactly to the undetectable error events of the original concatenation of CRC and convolutional codes. With these two methods as tools, we design CRC codes for the most common 64-state convolutional code for information length $k = 1024$ and compare with existing CRC codes, demonstrating the performance benefits of utilizing the inner code structure.

C.-Y. Lou, B. Daneshrad, and R. D. Wesel are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: chungyulou@ucla.edu; babak@ee.ucla.edu; wesel@ee.ucla.edu).

This material is based upon work supported by the National Science Foundation under Grant Number 1162501. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

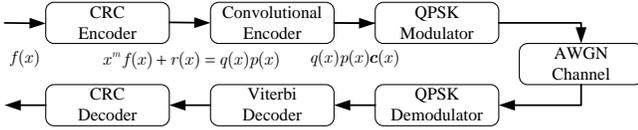


Fig. 1. Block diagram of a system employing CRC and convolutional codes.

This paper is organized as follows: Section II provides the system model. Section III presents the exclusion and construction methods for computing the undetected error probability of a CRC code concatenated with an inner convolutional code. Section IV describes how these two methods can be used to design a CRC code to minimize the undetected error probability for a specific feedforward convolutional code and information length. Section V applies this design approach to the most common 64-state convolutional code for information length $k = 1024$. Section VI concludes the paper.

II. SYSTEM MODEL

Fig. 1 shows the block diagram of our system model employing a CRC code concatenated with a convolutional code in an additive white Gaussian noise (AWGN) channel. The k -bit information sequence is expressed as a binary polynomial $f(x)$ of degree smaller than or equal to $k - 1$. The m parity bits are the remainder $r(x)$ of $x^m f(x)$ divided by the degree- m CRC generator polynomial $p(x)$. Thus the $n = k + m$ -bit sequence described by $x^m f(x) + r(x)$ is divisible by $p(x)$ producing the k -bit quotient $q(x)$ and a remainder of zero. The CRC-encoded sequence can also be expressed as $q(x)p(x)$. Thus $q(x)$ has a one-to-one relationship with $f(x)$. Note that $q(x)p(x)$ is the result of processing the sequence $x^m q(x)$ ($q(x)$ and m trailing zeros) by the CRC encoder circuit described by $p(x)$.

The transmitter uses a feedforward, terminated, rate- $\frac{1}{N}$ convolutional code having ν memories with generator polynomial $c(x) = [c_1(x), c_2(x), \dots, c_N(x)]$. The output $q(x)p(x)c(x)$ of the convolutional encoder is sent to the AWGN channel using quadrature phase-shift keying (QPSK) modulation.

Because the CRC bits are added at the end of the sequence, the highest degree term of $f(x)$ is the bit that is first in time and first to enter the convolutional encoder. Consistent with this convention and in contrast to common representations, the highest degree terms of $c(x)$ represent the most recent encoder input bits. Thus a convolutional encoder with the generator $G(D) = [1 + D^3 + D^4, 1 + D + D^2]$ will have $c(x) = [x^4 + x + 1, x^4 + x^3 + x^2]$.

The demodulated symbols are fed into a soft Viterbi decoder. The CRC decoder checks the n -bit sequence resulting from Viterbi decoding. An undetected error occurs when the receiver declares error-free decoding when the Viterbi decoder identified an incorrect codeword.

This work can be applied to feedback convolutional codes as well. Consider a feedback convolutional code with generator polynomial $c(x)/c_{FB}(x)$, where $c_{FB}(x)$ is its feedback connection polynomial. This feedback code has the same set of codewords as a feedforward code with generator polynomial $c(x)$. Assume the Viterbi decoder selects a wrong message $t(x)$ at the feedforward convolutional decoder output. In most

cases, the same received signal is decoded as $t(x)c_{FB}(x)$ by the feedback convolutional decoder. However, it is possible that the message decoded by the feedback decoder is not a multiple of $c_{FB}(x)$. Such trellis deviation must happen during the termination of the codeword and corresponds to either a long error sequence with large codeword Hamming distance or a short error sequence occurring only at the end of the codeword. Either of these cases should not dominate the overall codeword error performance and thus we only consider message errors of the form $t(x)c_{FB}(x)$.

Let the greatest common divisor of $p(x)$ and $c_{FB}(x)$ be $c_{gcd}(x)$. The polynomial $p(x)$ divides $t(x)c_{FB}(x)$ if and only if $p(x)/c_{gcd}(x)$ divides $t(x)$. Hence, the undetected error probability of this CRC code concatenated with the feedback convolutional code is approximated by that obtained by the CRC code $p(x)/c_{gcd}(x)$ concatenated with the feedforward convolutional code $c(x)$ and can be analyzed using the methods presented in this paper. Furthermore, a smart choice when using a CRC code concatenated with a feedback convolutional code is to pick $p(x)$ and $c_{FB}(x)$ relatively prime.

III. UNDETECTED ERROR PROBABILITY ANALYSIS

Let $e(x)$ be the polynomial of error bits in the Viterbi-decoded message so that the decoded n -bit sequence followed by ν zeros (for termination) is expressed as $q(x)p(x)x^\nu + e(x)$. If $e(x) \neq 0$ is divisible by $p(x)$, then this error is undetectable by the CRC decoder. This section presents two methods, the exclusion method and the construction method, to calculate the probability that a non-zero error $e(x)$ occurs that is undetectable.

A. Exclusion Method

The exclusion method enumerates the possible error patterns of the convolutional code and excludes the patterns detectable by the CRC code. The probability of the unexcluded error patterns is the undetected error probability. The exclusion method filters out part of the distance spectrum of the convolutional code through a divisibility test to create the distance spectrum of the undetectable errors of the concatenated code.

1) *Undetectable Single Error*: An error event occurs when the decoded trellis path leaves the encoded trellis path once and rejoins it once. Let $e_{d,i}(x)$ be the polynomial of message error bits associated with the i^{th} error event that leads to a codeword distance d from the transmitted codeword, where the range of i is later specified in (1). Note that in this paper, the term “distance” always refers to the convolutional code output Hamming distance. Let this error event have length $l_{d,i}$. Both $e_{d,i}(x)$ and $l_{d,i}$ are obtained through computer search of the given convolutional code. The first (highest power) term of $e_{d,i}(x)$ is $x^{l_{d,i}-1}$ and the last (lowest power) term is x^ν because every error event starts with a one and ends with a one followed by ν consecutive zeros.

If the received data frame contains only one error event, then the polynomial of message error bits can be expressed as $e(x) = x^g e_{d,i}(x)$, where $g \in [0, n + \nu - l_{d,i}]$ indicates the possible locations where this error event may appear. If $e_{d,i}(x)$

is divisible by $p(x)$, this error event, including all of its offsets g , will be undetectable.

The union bound of such error probability is given by

$$P_{UD,1} \leq \sum_{d=d_{\text{free}}}^{\infty} \sum_{i=1}^{a_d} \mathbb{I}(p(x) | e_{d,i}(x)) \max\{0, n + \nu - l_{d,i} + 1\} P(d), \quad (1)$$

where d_{free} is the free distance of the convolutional code, a_d is the number of error events with output distance d , the indicator function $\mathbb{I}(\cdot)$ returns one when $e_{d,i}(x)$ is divisible by $p(x)$ and zero otherwise, and $P(d)$ is the pairwise error probability of an error event with distance d . The \max operator ensures that the number of possible locations is always nonnegative even when $l_{d,i}$ is large. Note that while (1) does not explicitly use the generator polynomial of the convolutional code, it does implicitly depend on the generator polynomial because the generator polynomial determines the valid trellis error events $e_{d,i}(x)$. The subscript “1” in $P_{UD,1}$ means that this probability only includes undetectable errors that are single error events. We call this type of error an *undetectable single error*.

For a QPSK system operated in an AWGN channel, $P(d)$ can be computed using the tail probability function of standard normal distribution, i.e. Gaussian Q-function, as [13]

$$P(d) = Q(\sqrt{2d\gamma}) \leq Q(\sqrt{2d_{\text{free}}\gamma}) e^{-(d-d_{\text{free}})\gamma}, \quad (2)$$

where $\gamma = E_s/N_0$ is the signal-to-noise ratio (SNR) of a QPSK symbol, and E_s and $N_0/2$ denote the received symbol energy and one-dimensional noise variance, respectively. Note that the accuracy of (2) comes in part from a knowledge of d_{free} . A useful Q-function approximation when knowledge of d_{free} is not available is presented in [14]. To generalize (2) to a higher-order quadrature amplitude modulation (QAM) system with bit-interleaved coded modulation using a random interleaver, one can multiply γ with a modulation-dependent factor to obtain an approximation. Details can be found in [14] as well.

To compute $P_{UD,1}$ each error event $e_{d,i}$ must first be identified as either divisible by $p(x)$ or not. One approach is to truncate (1) at \tilde{d} to get an approximation, in which case all error events with distance $d \leq \tilde{d}$ can be stored and this set of error events can be tested for divisibility by the CRC polynomial $p(x)$. The choice of \tilde{d} is based on the computational and storage capacity available to implement an efficient search such as [15]. The required memory size to store the error events is proportional to $\sum_{d=d_{\text{free}}}^{\tilde{d}} \sum_{i=1}^{a_d} l_{d,i}$.

The approximation of (1) can be quite tight if the probability of the terms with $d > \tilde{d}$ is negligible. However, assuming that all error events with $d > \tilde{d}$ are undetectable provides

$$P_{UD,1} \leq \sum_{d=\tilde{d}+1}^{\infty} n a_d P(d) + \sum_{d=d_{\text{free}}}^{\tilde{d}} \sum_{i=1}^{a_d} \{P(d) \cdot \mathbb{I}(p(x) | e_{d,i}(x)) \max\{0, n + \nu - l_{d,i} + 1\}\}, \quad (3)$$

where $l_{d,i}$ for $d > \tilde{d}$ is replaced with $\nu+1$ because the shortest error event has length $\nu+1$. Note that (3) can be computed because the error pattern $e_{d,i}(x)$ is only required for $d \leq$

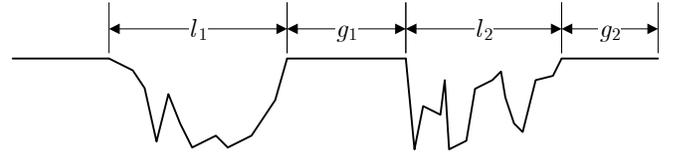


Fig. 2. An illustration of two error events.

\tilde{d} , and the distance spectrum a_d for $d > \tilde{d}$ provided by the transfer function [16] can be used for the first term of (3) as in the frame error rate (FER) bounds of [17].

In addition to the undetectable single error events discussed above, an undetectable error could consist of two or more error events, even though each of the error events itself is detectable. We will first discuss the case with two error events, and then generalize it to multiple error events.

2) *Undetectable Double Error*: A double error involves two error events $e_{d_1,i_1}(x)$ and $e_{d_2,i_2}(x)$ with respective lengths l_{d_1,i_1} and l_{d_2,i_2} . To simplify notation, for $u \in \{1, 2\}$ let $e_u(x)$ and l_u refer to $e_{d_u,i_u}(x)$ and l_{d_u,i_u} , respectively. In a data frame with two error events, the polynomial of error bits in the message can be expressed as $e(x) = x^{g_1+g_2+l_2}e_1(x) + x^{g_2}e_2(x)$, where the exponents of two x 's tell the locations of the two error events. Furthermore, $g_1 \geq 0$ represents the interval of symbols (gap) between two error events and satisfies $g_1 + g_2 + l_1 + l_2 \leq n + \nu$. If $x^{g_1+l_2}e_1(x) + e_2(x)$ is divisible by $p(x)$, the error is an *undetectable double error*. Its length is $l_1 + l_2 + g_1$ and its offset is g_2 .

An upper bound of the probability of an undetectable double error occurring in the codeword is given by

$$P_{UD,2} \leq \sum_{d_1=d_{\text{free}}}^{\infty} \sum_{d_2=d_{\text{free}}}^{\infty} \sum_{i_1=1}^{a_{d_1}} \sum_{i_2=1}^{a_{d_2}} \sum_{g_1=0}^{n+\nu-l_1-l_2} P(d_1 + d_2) \cdot \mathbb{I}(p(x) | x^{g_1+l_2}e_1(x) + e_2(x)) \cdot (n + \nu - l_1 - l_2 - g_1 + 1). \quad (4)$$

The distance of the double error event is simply the sum of the individual distances because the error events are completely separated as shown in Fig. 2. Two error events that overlap are simply a longer single error event, which was treated in Section III-A1.

Computation of (4) exactly is problematic because it requires infinite search depth. Replacing the l_1 and l_2 of large-distance terms in (4) with $\nu+1$ yields a more computation-friendly upper bound similar to (3) as follows:

$$P_{UD,2} \leq \sum_{(d_1,d_2) \in \mathbf{D}_{\tilde{d},2}} \sum_{i_1=1}^{a_{d_1}} \sum_{i_2=1}^{a_{d_2}} \sum_{g_1=0}^{n+\nu-l_1-l_2} P(d_1 + d_2) \cdot \mathbb{I}(p(x) | x^{g_1+l_2}e_1(x) + e_2(x)) \cdot (n + \nu - l_1 - l_2 - g_1 + 1) + \sum_{(d_1,d_2) \notin \mathbf{D}_{\tilde{d},2}} \frac{1}{2} (n - \nu) (n - \nu - 1) \cdot a_{d_1} a_{d_2} P(d_1 + d_2), \quad (5)$$

where $\mathbf{D}_{\tilde{d},2} = \{(d_1, d_2) \mid d_1, d_2 \geq d_{\text{free}}, d_1 + d_2 \leq \tilde{d}\}$.

Because computational complexity limits the *single* error event distance we can search, it is feasible to replace \tilde{d} in $\mathbf{D}_{\tilde{d},2}$ with $\tilde{d} + d_{\text{free}}$. This is not particularly helpful because we have already assumed errors with distance $d > \tilde{d}$ are negligible or undetectable during the calculation of undetectable single errors. We can also replace all l_1 and l_2 in (5) with $\nu + 1$ and provide another upper bound which does not require any length information.

As described in Appendix A, the number of g_1 values at which to check the divisibility of $x^{g_1+l_2}e_1(x) + e_2(x)$ by $p(x)$ can be significantly reduced from $n + \nu - l_1 - l_2 + 1$.

3) *Total Undetected Error Probability*: In general, s error events could possibly form an undetectable s -tuple error, whether each of them is detectable or not. The error bits in the message can be expressed as

$$e(x) = \sum_{u=1}^s \left(\prod_{v=u+1}^s x^{g_v+l_v} \right) x^{g_u} e_u(x). \quad (6)$$

This combined error will be undetectable if $e(x)$ is divisible by $p(x)$. Therefore, the probability of undetectable s -tuple errors $P_{\text{UD},s}$ can be approximated or bounded in the same way as (5). For simpler notation, define sets

$$\begin{aligned} \mathbf{D}_{\tilde{d},s} &= \left\{ (d_1, \dots, d_s) \mid d_u \geq d_{\text{free}} \quad \forall u, \sum_{u=1}^s d_u \leq \tilde{d} \right\} \\ \mathbf{I}_s &= \{ (i_1, \dots, i_s) \mid i_u \in [1, a_{d_u}] \quad \forall u \in [1, s] \} \\ \mathbf{G}_s &= \left\{ (g_1, \dots, g_s) \mid g_u \geq 0 \quad \forall u, \sum_{u=1}^s g_u \leq n + \nu - \sum_{u=1}^{s+1} l_u \right\}. \end{aligned}$$

Note that \mathbf{G}_s is determined by all d_u and i_u , and \mathbf{I}_s is determined by all d_u .

Since an undetectable error may consist of any number of error events, the probability of having an undetectable error in the codeword P_{UD} is upper bounded by $\sum_{s=1}^{\infty} P_{\text{UD},s}$. Using the computation-friendly bound of each term such as (3) and (5), we obtain

$$\begin{aligned} P_{\text{UD}} &\leq \sum_{s=1}^{\infty} P_{>\tilde{d},s} + \sum_{s=2}^{\infty} \left\{ \sum_{(d_1, \dots, d_s) \in \mathbf{D}_{\tilde{d},s}} P \left(\sum_{u=1}^s d_u \right) \right. \\ &\quad \cdot \sum_{(i_1, \dots, i_s) \in \mathbf{I}_s} \sum_{(g_1, \dots, g_{s-1}) \in \mathbf{G}_{s-1}} I(p(x) \mid e(x)) \\ &\quad \cdot \left. \left(n + \nu - \sum_{u=1}^s l_u - \sum_{u=1}^{s-1} g_u + 1 \right) \right\} + \sum_{d_1 \in \mathbf{D}_{\tilde{d},1}} \{P(d_1) \\ &\quad \cdot \sum_{i_1 \in \mathbf{I}_1} I(p(x) \mid e_1(x)) \max\{0, n + \nu - l_1 + 1\}\}, \quad (7) \end{aligned}$$

where the composition of $e(x)$ depends on the number of error events s as in (6) and $P_{>\tilde{d},s}$ is the sum of probability of all s -tuple errors whose distances are greater than \tilde{d} .

The probability sum of all large-distance s -tuple errors is

$$P_{>\tilde{d},s} = \sum_{(d_1, \dots, d_s) \notin \mathbf{D}_{\tilde{d},s}} \binom{n + \nu - s\nu}{s} \left(\prod_{u=1}^s a_{d_u} \right) P \left(\sum_{u=1}^s d_u \right), \quad (8)$$

where the combinatorial number represents the number of ways to have s length- $(\nu + 1)$ error events in a length- $(n + \nu)$ sequence. Using (2) $P_{>\tilde{d},1}$ can be upper bounded by

$$P_{>\tilde{d},1} \leq Q \left(\sqrt{2d_{\text{free}}\gamma} \right) e^{d_{\text{free}}\gamma} \left\{ \bar{P} - \sum_{d_1 \in \mathbf{D}_{\tilde{d},1}} n a_{d_1} e^{-d_1\gamma} \right\}, \quad (9)$$

where \bar{P} is defined as $\bar{P} \triangleq nT(D, L)|_{D=e^{-\gamma}, L=1}$ using the transfer function [16]

$$T(D, L) = \sum_{d=d_{\text{free}}}^{\infty} \sum_{i=1}^{a_d} D^d L^{L_{d,i}}. \quad (10)$$

Therefore, the sum of $P_{>\tilde{d},s}$ terms can be upper bounded by

$$\begin{aligned} \sum_{s=1}^{\infty} P_{>\tilde{d},s} &\leq \sum_{s=1}^{\infty} \sum_{(d_1, \dots, d_s) \notin \mathbf{D}_{\tilde{d},s}} \frac{n^s}{s!} \left(\prod_{u=1}^s a_{d_u} \right) P \left(\sum_{u=1}^s d_u \right) \\ &\leq Q \left(\sqrt{2d_{\text{free}}\gamma} \right) e^{d_{\text{free}}\gamma} \sum_{s=1}^{\infty} \left\{ \frac{n^s}{s!} \right. \\ &\quad \cdot \left. \sum_{(d_1, \dots, d_s) \notin \mathbf{D}_{\tilde{d},s}} \left(\prod_{u=1}^s a_{d_u} e^{-d_u\gamma} \right) \right\} \quad (11a) \end{aligned}$$

$$\begin{aligned} &= Q \left(\sqrt{2d_{\text{free}}\gamma} \right) e^{d_{\text{free}}\gamma} \sum_{s=1}^{\infty} \left\{ \frac{1}{s!} \bar{P}^s \right. \\ &\quad \left. - \frac{n^s}{s!} \sum_{(d_1, \dots, d_s) \in \mathbf{D}_{\tilde{d},s}} \left(\prod_{u=1}^s a_{d_u} e^{-d_u\gamma} \right) \right\} \quad (11b) \end{aligned}$$

$$\begin{aligned} &= Q \left(\sqrt{2d_{\text{free}}\gamma} \right) e^{d_{\text{free}}\gamma} \left\{ e^{\bar{P}} - 1 \right. \\ &\quad \left. - \sum_{s=1}^{\infty} \frac{n^s}{s!} \sum_{(d_1, \dots, d_s) \in \mathbf{D}_{\tilde{d},s}} \left(\prod_{u=1}^s a_{d_u} e^{-d_u\gamma} \right) \right\}. \quad (11c) \end{aligned}$$

The bound of Gaussian Q-function (2) is used in (11a), and the transfer function is used to evaluate the sum of all s -tuple errors in (11b). Using (7) and (11c), a bound of P_{UD} can be calculated. In fact, when an undetectable error occurs in a codeword, the receiver may still detect an error if a detectable error happens somewhere else in the codeword. Therefore, P_{UD} , which is the probability of having an undetectable error in the codeword, is an upper bound of the undetected error probability. When channel error rate is low, having a detected error occur along with the undetected error is a rare event so that our bound will be tight.

Due to the limitation of searchable depth \tilde{d} of error events, the exclusion method is only useful when undetectable errors with distances $d \leq \tilde{d}$ dominate. However, this requirement could be violated by a powerful CRC code that detects the short-distance errors. The next subsection introduces the construction method, which allows the search depth to increase to distance $\hat{d} > \tilde{d}$.

B. Construction Method

The construction method utilizes the fact that all undetectable errors at the CRC decoder input are multiples of the

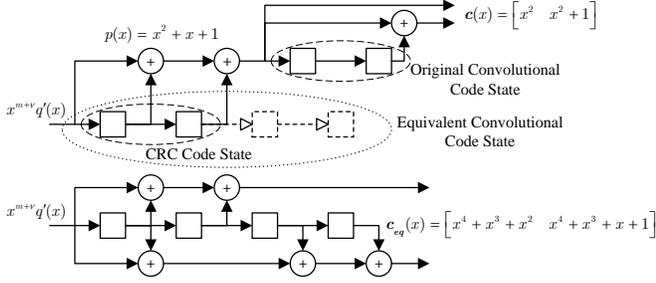


Fig. 3. An example of CRC code, original convolutional code, and equivalent convolutional code.

CRC generator $p(x)$. This method constructs an equivalent convolutional encoder $c_{eq}(x) = p(x)c(x)$ to isolate these errors. The set of non-zero codewords of $c_{eq}(x)$ is exactly the set of erroneous codewords (given an all-zeros transmission) that lead to undetectable errors for the concatenation of the CRC generator polynomial and the original convolutional encoder. Thus the probability of undetectable error is exactly the FER of $c_{eq}(x)$.

Fig. 3 shows an example of how $c_{eq}(x)$ is constructed from $c(x)$ and $p(x)$, where $q'(x)$ with $m + \nu$ trailing zeros is the input that generates the undetectable erroneous codeword. The input/output behavior of $c_{eq}(x)$ is exactly the same as the concatenation of $p(x)$ and $c(x)$. For $p(x)$ with m memory elements and $c(x)$ with ν memory elements, $c_{eq}(x)$ has $m + \nu$ memory elements. At a given time, the state of the original encoder $c(x)$ can be inferred from the state of $c_{eq}(x)$ because the state of $c_{eq}(x)$ contains the last $m + \nu$ inputs to $p(x)$ which are sufficient to compute the last ν outputs of $p(x)$, which exactly comprise the state of $c(x)$.

1) *States of the Equivalent Encoder:* Define the all-zero state \mathbf{S}^Z to be the state where all memory elements of the equivalent encoder are zero. When the equivalent encoder is in \mathbf{S}^Z , then the original encoder is also in its zero state. Avoiding trivial cases by assuming that the x^{m-1} and x^0 coefficients of $p(x)$ are 1, there are $2^m - 1$ equivalent encoder states in addition to \mathbf{S}^Z that correspond to the all-zero state of the original encoder. To see this, consider the top diagram in Fig. 3 in which the equivalent encoder state is shown as the state of the CRC encoder extended with ν additional memories. Note that whatever state the equivalent encoder is in, there is a sequence of ν bits that will produce ν zeros at the output of the CRC encoder that will drive the original encoder state to zero. The specific set of ν bits is a function of the m -bits of state in the CRC encoder. Thus for any m -bits pattern there is a corresponding $(m + \nu)$ -bit state of the equivalent encoder that corresponds to the original encoder being in the zero state.

We call the $2^m - 1$ non-zero equivalent encoder states for which the original encoder state is zero *detectable-zero states*, forming a set \mathbf{S}^D , because they correspond to the termination of a detectable error event in the original encoder. The remaining $2^{m+\nu} - 2^m$ states of the equivalent code are called *non-zero states*, forming a set \mathbf{S}^N , because the corresponding states of the original code are not zero. To terminate an error event in the original encoder, the trellis of the equivalent code transitions from a state in \mathbf{S}^N to \mathbf{S}^Z or to a

TABLE I
AN EXAMPLE OF STATE TYPES WITH $p(x) = x^2 + x + 1$ AND $\nu = 2$.

Time	State Type	Equivalent Code		Original Code	
		State	Input	State	Input
0	\mathbf{S}^Z	0000	1	00	1
1	\mathbf{S}^N	0001	1	01	0
2	\mathbf{S}^N	0011	0	10	0
3	\mathbf{S}^D	0110	1	00	0
4	\mathbf{S}^D	1101	0	00	1
5	\mathbf{S}^N	1010	0	01	1
6	\mathbf{S}^N	0100	0	11	0
7	\mathbf{S}^N	1000	0	10	0
8	\mathbf{S}^Z	0000		00	

state in \mathbf{S}^D . If the transition is to \mathbf{S}^D the cumulative errors are detectable because the portion of $q'(x)p(x)$ till this moment is not divisible by $p(x)$. If the transition is to \mathbf{S}^D , the cumulative errors are detectable because the portion of $q'(x)p(x)$ till this moment is divisible by $p(x)$.

Table I shows an example using the set-up shown in Fig. 3 with $p(x) = x^2 + x + 1$, $q'(x) = x^3 + x^2 + 1$, and $\nu = 2$. The original encoder $c(x)$ does not need to be specified for the results in Table I because any feedforward encoder will produce the same state sequence. The zero state of the original code is visited at time 0, 3, 4, and 8. At time 0, the state begins from \mathbf{S}^Z . At time 3, the first $c(x)$ error event ends. This error event is detectable if the codeword ends at time 3 because the input to the original code, i.e. x^2 , is not divisible by $p(x)$. The state remains in \mathbf{S}^D at time 4. At time 5 a second $c(x)$ error event begins. At time 8, the $c_{eq}(x)$ state returns to \mathbf{S}^Z , which is not only the end of the second $c(x)$ error event but also the end of an undetectable double error.

Note that $c_{eq}(x)$ is catastrophic because its generator polynomials have a common factor $p(x)$. The catastrophic behavior is expressed through the zero distance loops that occur as the equivalent encoder traverses a sequence of states in \mathbf{S}^D while the original convolutional encoder stays in the zero state. Thus time spent in \mathbf{S}^D between $c(x)$ error events can lengthen an undetectable error without increasing its distance.

2) *Error Events in the Equivalent Encoder:* Since the all-zero codeword is assumed to be sent, the correct path remains in \mathbf{S}^Z forever. An error event in the equivalent encoder occurs if the trellis path leaves \mathbf{S}^Z and returns \mathbf{S}^Z without any visits to \mathbf{S}^Z in between. We will classify these error events according to the number of times it enters \mathbf{S}^D from \mathbf{S}^N during the deviation. If a trellis path enters \mathbf{S}^D from \mathbf{S}^N $s - 1$ times, then it is classified as an undetectable s -tuple error. In an undetectable s -tuple error, there are exactly s segments of consecutive transitions between states in \mathbf{S}^N , which correspond to s error events of the original encoder. These segments are separated by segments of consecutive transitions between states in \mathbf{S}^D .

Fig. 4 illustrates an undetectable triple error in a system with $\nu = 2$ and $m = 2$. The three error events of the original code are separated by visits to \mathbf{S}^D . The path can leave \mathbf{S}^D right after entering it as shown between the first and the second error events; the path can also stay in \mathbf{S}^D for a while and then leave \mathbf{S}^D from a state different from where it enters \mathbf{S}^D as shown between the second and the third error events. Note that \mathbf{S}^Z is

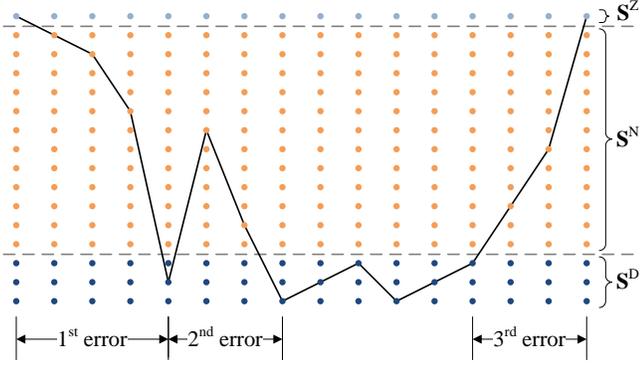


Fig. 4. A trellis diagram of an equivalent convolutional code with $\nu = 2$ and $m = 2$ having an undetectable triple error. The states are reordered such that the all-zero state is at the top and the detectable-zero states are at the bottom.

never directly connected to any state of \mathbf{S}^D .

The probability of undetectable single errors is bounded in a similar way as (3) by

$$P_{\text{UD},1} \leq P_{>\hat{d},1} + \sum_{d \in \mathbf{D}_{\hat{d},1}} \sum_{i=1}^{a_d^{\text{ZZ}}} \max\{0, n + \nu - l_{d,i}^{\text{ZZ}} + 1\} P(d), \quad (12)$$

where a_d^{ZZ} is the number of error events with distance d starting from \mathbf{S}^Z and ending at \mathbf{S}^Z while never traversing a state in \mathbf{S}^D . The length $l_{d,i}^{\text{ZZ}}$ is the length of the i^{th} error event counted in a_d^{ZZ} . Note that this expression requires the distance spectrum a_d^{ZZ} and length spectrum $l_{d,i}^{\text{ZZ}}$ for $d \in [d_{\text{free}}, \hat{d}]$ obtained using computer search.

Unlike (3), (12) does not need to check the divisibility of error events. Hence, there is no need to store the error patterns anymore but only their distances and lengths. Consequently the search depth \hat{d} can go beyond \tilde{d} .

Let \mathbf{S}_i^D be the i^{th} state in \mathbf{S}^D , where $i \in [1, 2^m - 1]$. Define Δ_i as the subset of \mathbf{S}^D composed of all states connected with \mathbf{S}_i^D through a path that only includes states in \mathbf{S}^D . Appendix B shows that Δ_i and the x -cyclotomic coset modulo $p(x)$ discussed in Appendix A are equivalent. Thus if $p(x)$ is a primitive polynomial, all sets Δ_i are identical and $|\Delta_i| = 2^m - 1$ is maximized. This may lead to fewer undetectable double errors as shown below and is desirable in the design of a CRC code.

Similar to (5), the probability of undetectable double errors can be bounded by

$$P_{\text{UD},2} \leq P_{>\hat{d},2} + \sum_{(d_1, d_2) \in \mathbf{D}_{\hat{d},2}} P(d_1 + d_2) \sum_{\phi=1}^{2^m-1} \sum_{\mathbf{S}_\phi^D \in \Delta_\phi} \sum_{i_1=1}^{a_{d_1, \phi}^{\text{ZD}}} \sum_{i_2=1}^{a_{d_2, \phi}^{\text{DZ}, \psi}} \sum_{t_1=0}^{[(n+\nu-l^{\min})/|\Delta_\phi|]} (n + \nu - l^{\min} - |\Delta_\phi| t_1 + 1), \quad (13)$$

where

$$l^{\min} = l_{d_1, i_1}^{\text{ZD}, \phi} + l_{d_2, i_2}^{\text{DZ}, \psi} + \delta_{\phi, \psi} \quad (14)$$

is the shortest possible length of the undetectable double error specified by $(d_1, d_2, \phi, \psi, i_1, i_2)$.

In (13), ϕ is the index of the state at which the trellis enters \mathbf{S}^D at the end of the first error event, and ψ is the index of the state at which the trellis leaves \mathbf{S}^D at the beginning of the second error event. In (14), $\delta_{\phi, \psi} < |\Delta_\phi|$ is the number of hops required to go from \mathbf{S}_ϕ^D to \mathbf{S}_ψ^D without leaving \mathbf{S}^D for $\mathbf{S}_\psi^D \in \Delta_\phi$, where $\delta_{\phi, \psi} = 0$ if $\phi = \psi$. The number of error events starting at \mathbf{S}^Z and ending at \mathbf{S}_ϕ^D with distance d_1 is $a_{d_1}^{\text{ZD}, \phi}$, and the i_1^{th} error event of them has length $l_{d_1, i_1}^{\text{ZD}, \phi}$, where both numbers are obtained by computer search. The variables $a_{d_2}^{\text{DZ}, \psi}$ and $l_{d_2, i_2}^{\text{DZ}, \psi}$ are defined in a similar way while the error event starts in \mathbf{S}_ψ^D and ends in \mathbf{S}^Z . Furthermore, t_1 specifies the number of cycles the trellis stays in Δ_ϕ and its upper limit makes sure that the total length of the undetectable double error does not exceed $n + \nu$, the number of trellis stages in the codeword. As in (5) $P_{>\hat{d},2}$ can often be neglected because terms with $d_1 + d_2 > \hat{d}$ have negligible probability.

Although the number and lengths of error events connecting \mathbf{S}^Z and states in \mathbf{S}^D are obtained by computer searches, the required search depth is only $\hat{d} - d_{\text{free}}$. Moreover, $a_{d_1}^{\text{ZD}, \phi}$ and $l_{d_1, i_1}^{\text{ZD}, \phi}$ for all ϕ can be found while searching for a_d^{ZZ} and $l_{d,i}^{\text{ZZ}}$ because these two types of error events both start from \mathbf{S}^Z . Regarding $a_{d_2}^{\text{DZ}, \psi}$ and $l_{d_2, i_2}^{\text{DZ}, \psi}$, the associated error events start from $2^m - 1$ different states and can be found through $2^m - 1$ separated searches. Nevertheless, since these error events end at \mathbf{S}^Z , only one backward search is necessary to capture all of them. In the backward search, bits in shift registers move backward. One can simply treat x as x^{-1} in polynomial representations and apply the same search algorithm.

3) *Undetected Error Probability*: As shown in Fig. 4, an undetectable s -tuple error is composed of several parts: one error event from \mathbf{S}^Z to a state in \mathbf{S}^D , $s-2$ error events from a state in \mathbf{S}^D to a state in \mathbf{S}^D with visits to \mathbf{S}^N in between, the final error event from a state in \mathbf{S}^D to \mathbf{S}^Z , and also $s-1$ paths inside \mathbf{S}^D connecting consecutive error events. Note that in the example of Fig. 4, where $s = 3$, where the path connecting the first and the second error events has a length of zero. Let ϕ_u and ψ_u be the indices of the start and end states of the u^{th} transitions in \mathbf{S}^D , respectively. In Fig. 4, we have $\phi_1 = \psi_1 = 2$ for the first transition; the second transition starts from $\mathbf{S}_{\phi_2}^D$ and ends at $\mathbf{S}_{\psi_2}^D$, where $\phi_2 = 3$ and $\psi_2 = 1$. Also, let the number of error events started at $\mathbf{S}_{\psi_{u-1}}^D$ and ended at $\mathbf{S}_{\phi_u}^D$ with distance d_u be $a_{d_u}^{\text{DD}, \psi_{u-1}, \phi_u}$ and the length of the i_u^{th} error event of them be $l_{d_u, i_u}^{\text{DD}, \psi_{u-1}, \phi_u}$, where both numbers are obtained by computer search. Although we need to perform $2^m - 1$ separated searches to obtain all $a_{d_u}^{\text{DD}, \psi_{u-1}, \phi_u}$ and $l_{d_u, i_u}^{\text{DD}, \psi_{u-1}, \phi_u}$, a search depth of $\hat{d} - (s-1)d_{\text{free}}$ is sufficient.

To simplify the notation, define the following sets:

$$\Phi_s = \{(\phi_1, \dots, \phi_s) \mid \phi_u \in [1, 2^m - 1] \quad \forall u \in [1, s]\}$$

$$\Psi_s = \{(\psi_1, \dots, \psi_s) \mid \psi_u \in \Delta_{\phi_u} \quad \forall u \in [1, s]\}$$

$$\mathbf{I}'_s = \left\{ (i_1, \dots, i_s) \mid i_1 \in [1, a_{d_1}^{\text{ZD}, \phi_1}], i_s \in [1, a_{d_s}^{\text{DZ}, \psi_{s-1}}], \right.$$

$$\left. i_u \in [1, a_{d_u}^{\text{DD}, \psi_{u-1}, \phi_u}] \quad \forall u \in [2, s-1] \right\}$$

$$\mathbf{T}_s = \left\{ (t_1, \dots, t_s) \mid t_u \geq 0 \quad \forall u \in [1, s], \right.$$

$$\left. \sum_{u=1}^s |\Delta_{\phi_u}| t_u \leq n + \nu - l_{s+1}^{\min} \right\},$$

where l_s^{\min} is the shortest possible length of the undetectable s -tuple error specified in a similar way as (14) and given by

$$l_s^{\min} = l_{d_1, i_1}^{\text{ZD}, \phi_1} + \sum_{u=2}^{s-1} l_{d_u, i_u}^{\text{DD}, \psi_{u-1}, \phi_u} + l_{d_s, i_s}^{\text{DZ}, \psi_{s-1}} + \sum_{u=1}^{s-1} \delta_{\phi_u, \psi_u}. \quad (15)$$

Similar to (7), the probability of having an undetectable error is now bounded by

$$\begin{aligned} P_{\text{UD}} &\leq \sum_{d_1 \in \mathbf{D}_{\hat{d}, 1}} \sum_{i=1}^{a_{d_1}^{\text{ZZ}}} \max\{0, n + \nu - l_{d_1, i}^{\text{ZZ}} + 1\} P(d_1) \\ &+ \sum_{s=2}^{\infty} \sum_{(d_1, \dots, d_s) \in \mathbf{D}_{\hat{d}, s}} P\left(\sum_{u=1}^s d_u\right) \sum_{(\phi_1, \dots, \phi_{s-1}) \in \Phi_{s-1}} \\ &\sum_{(\psi_1, \dots, \psi_{s-1}) \in \Psi_{s-1}} \sum_{(i_1, \dots, i_s) \in \mathbf{I}'_s} \sum_{(t_1, \dots, t_{s-1}) \in \mathbf{T}_{s-1}} \\ &\left(n + \nu - l_s^{\min} - \sum_{u=1}^{s-1} |\Delta_{\phi_u}| t_u + 1\right) + \sum_{s=1}^{\infty} P_{>\hat{d}, s}. \quad (16) \end{aligned}$$

The last term is the probability sum of all large-distance errors, which are assumed to be undetectable, and can be calculated using (11c). By letting $P_{>\hat{d}, s} = 0$, we obtain an approximation. By letting all l^{ZZ} , l^{ZD} , l^{DD} , and l^{DZ} equal to $\nu + 1$, we obtain a looser bound which does not require any length information. These two techniques are applicable to every $P_{\text{UD}, s}$, including (12) and (13).

C. Comparison

The main benefit of the construction method is that it is often able to search deeper than the exclusion method because the output pattern is not required. However, the required memory size scales with the number of states $2^{m+\nu}$ rather than 2^ν so this approach can encounter difficulty in analyzing high-order CRC codes. In contrast, the error events searched in the exclusion method belong to the original convolutional code, whose number of states is just 2^ν and is independent of the degree of the CRC code. In fact, both methods create the dominant parts (till \hat{d} or \hat{d}) of the distance spectrum of the equivalent catastrophic convolutional code with finite length. As explained in Section IV, we found it useful to draw on both approaches as we searched for optimal CRC polynomials for a specific convolutional code. The exclusion method is suitable for short-distance ($d \leq \hat{d}$) undetectable single errors, and the construction method is preferred when we search for longer undetectable single errors and double errors.

Fig. 5 compares the simulated undetected error probability to the bounds produced by the exclusion and construction methods. We consider the CRC code $p(x) = x^3 + x + 1$ concatenated with the memory size $\nu = 4$ convolutional code with generator polynomial $(23, 35)_8$ in octal and $d_{\text{free}} = 7$. The information length is $k = 1021$ bits and thus the CRC codeword length is $n = 1024$ bits. The FER of this original convolutional code is plotted as a reference. The equivalent

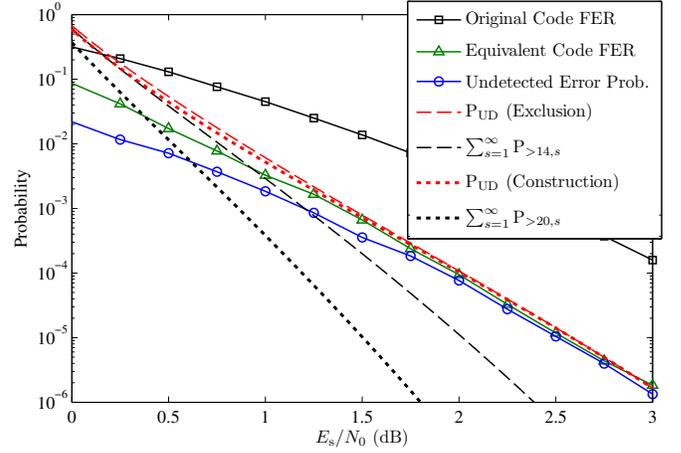


Fig. 5. A comparison of the simulated undetected error probability with the simulated frame error rate of the equivalent code and the analyses from the exclusion and construction methods. The system is equipped with the CRC code $p(x) = x^3 + x + 1$ and the convolutional code $(23, 35)_8$. The CRC codeword length is $n = 1024$ bits.

catastrophic convolutional code is $(255, 317)_8$ and its FER is also simulated.

We can see from Fig. 5 that the undetected error probability is upper bounded by the FER of the equivalent code, which equals P_{UD} , the probability of having an undetectable error in the CRC codeword, and this bound gets tighter as SNR increases. The equivalent code FER is above the probability of undetected error because it is possible that a frame has *both* an undetectable error event and a detectable error event, which causes a frame error in the equivalent code but does not cause an undetected error in the concatenated CRC and convolutional codes.

The upper bounds of P_{UD} are computed using the exclusion method (7) and construction method (16). In our calculation, the search depth limit of the exclusion method is $\tilde{d} = 14$, and $\hat{d} = 20$ is the depth limit for the construction method. Since $d_{\text{free}} = 7$, only undetectable single and double errors are considered. The probability sum of all large-distance terms given by (11c) are plotted to verify that they are negligible, except for the exclusion method at SNR below 0.75 dB. Although the construction method used a deeper search, it is still quite close to the exclusion method even in the low SNR region. It can be seen that these analysis methods deliver accurate bounds at high SNR for both the undetected error probability and the FER of the catastrophic code.

IV. SEARCH PROCEDURE FOR OPTIMAL CRC CODES

In this section, we will present an efficient way to find the optimal degree- m CRC code for a targeted convolutional code and information length k . Note that the performance of a CRC code depends on the information length [5]. A CRC code may be powerful for short sequences but have numerous undetectable long errors that are produced by a specific convolutional code.

Since the coefficients of x^m and x^0 terms are both one, there are 2^{m-1} candidates of degree- m CRC generator polynomials $p(x)$. In principle, either the exclusion method or

the construction method can produce the undetected error probability for each candidate allowing selection of the best $p(x)$. However, this process is time-consuming if m is large. Both exclusion and construction methods need to compute the distance spectrum of undetectable errors up to some distance d . We can reduce the computation time by skipping the distance spectrum searches of suboptimal CRC codes.

When the FER is low, the undetectable error rate of a CRC code is dominated by the undetectable errors with the smallest distance. Let the smallest distance of the undetectable errors be d_{\min} . We can evaluate a CRC-convolutional concatenated code by its distance spectrum at around d_{\min} . To be more precise, a polynomial should be removed from the candidate list if it has a smaller d_{\min} than the others or if it has more undetectable errors associated to the same d_{\min} . In a convolutional code, since the number of error events a_d grows exponentially as the associated distance d increases, the cost to find all undetectable errors grows exponentially as well. Hence, the CRC code search starts with $d = d_{\text{free}}$ and updates the candidate list by keeping only the CRC generator polynomials with the fewest undetectable errors. Next, repeat the procedure with the next higher d until only one polynomial remains in the list.

When $d < 2d_{\text{free}}$, only single errors are possible. The exclusion method can count the number of undetectable single errors of each candidate when $d \leq \tilde{d}$. We can perform a computer search for error events of the original code and check the divisibility of each of them. Note that if an error event is found to be undetectable, all of its possible offsets in the code-word should be counted. Since all candidates check the same set of error events, only one computer search with multiple divisibility checks (one for each CRC code) is sufficient. In contrast, using the construction method requires construction of equivalent encoders for each candidate separately. Hence, for the initial values of d near d_{free} , checking the divisibility via the exclusion method is preferred. Of course, once $d > \tilde{d}$, searching for undetectable single errors of the equivalent codes as in the construction method is the only approach.

When $d \geq 2d_{\text{free}}$, undetectable double errors need to be considered in addition to single errors. The divisibility test should be applied to all combinations of error event patterns $e_1(x)$, $e_2(x)$, and their gaps g_1 . Even if the concept of cyclotomic cosets discussed in Appendix A is utilized, we still need to construct all cyclotomic cosets through about 2^m divisions and also check if each of the remainder of $e_2(x)$ divided by $p(x)$ is in the same cyclotomic coset as the remainder of $e_1(x)$ divided by $p(x)$.

Alternatively, undetectable double errors can be directly created using the construction method. In the construction method, the error events connecting \mathbf{S}^Z and states in \mathbf{S}^D with distances between d_{free} and $d - d_{\text{free}}$ are found through computer searches. In fact, these events can be generated using the detectable error events of the original code previously found by exclusion if $d - d_{\text{free}} \leq \tilde{d}$. For example, since the detectable error pattern $e_1(x)$ is known, the corresponding error event in the equivalent encoder trellis starts from \mathbf{S}^Z and traverses the trellis with the input sequence given by the quotient of $x^m e_1(x)$ divided by $p(x)$. The state \mathbf{S}_ϕ^D , where it ends, is thus determined by the last $m + \nu$ input bits, and its

length $l_{d_1, i_1}^{\text{ZD}, \phi}$ has already been provided by the degree of $e_1(x)$.

For error events starting from states in \mathbf{S}^D and ending in \mathbf{S}^Z with pattern $e_2(x)$ previously obtained by exclusion, traverse the trellis in reverse from \mathbf{S}^Z with the input sequence given by the quotient of $x^{m+\nu+l'} e_2(x^{-1})$ divided by $x^m p(x^{-1})$, where $l' = l_{d_2, i_2}^{\text{DZ}, \psi} - 1$ is the degree of $e_2(x)$. Note that $x^{l'} e_2(x^{-1})$ and $x^m p(x^{-1})$ are the reverse bit-order polynomial representations of $e_2(x)$ and $p(x)$, respectively. The state \mathbf{S}_ψ^D where the error event begins is determined by the last $m + \nu$ input bits in reverse order.

According to the discussion at the end of Appendix A, d_{\min} is not likely to be much greater than $2d_{\text{free}}$ when information length k is long enough. In other words, the CRC code search algorithm is usually finished before reaching $3d_{\text{free}}$ and does not need to count the number of undetectable triple errors.

V. CRC DESIGN EXAMPLE FOR $\nu = 6$, $k = 1024$

As an example we present the best CRC codes of degree $m \leq 16$ specifically for the popular memory size $\nu = 6$ convolutional code with generator polynomial $(133, 171)_8$ with information length $k = 1024$ bits. Note that the proposed design method is applicable to all convolutional codes and information lengths, and not limited to the choices used for this example. The corresponding undetected error probability is also calculated and compared with existing CRC codes.

Table II shows the standard CRC codes listed in [5] and the best CRC codes found by the search procedure in Section IV. For degrees with no standard codes, those recommended by Koopman and Chakravarty in [5] are listed and called K&C. The notation of generator polynomials is in hexadecimal as used in [5]. For example, CRC-8 has generator polynomial $x^8 + x^7 + x^6 + x^4 + x^2 + 1$ expressed as 0xEA, where the most and least significant bits represent the coefficients of x^8 and x^1 terms, respectively. The coefficient of x^0 term is always one and thus omitted.

Table II also gives the distance spectrum of undetectable single errors a_d^{ZZ} of each CRC code up to $d = 22$. The distance spectrum of the original convolutional code a_d is given as a reference. Note that, since this convolutional code has $d_{\text{free}} = 10$, a smaller a_{20}^{ZZ} or a_{22}^{ZZ} does not mean fewer undetectable errors at distance $d = 20$ or 22 . Undetectable double errors should also be counted for $d \geq 20$ to judge a candidate.

During the search for the best CRC codes with degrees $m \leq 11$, only single errors need to be considered because one candidate will outperform all the others before looking at $d = 20$. Although the best degree-11 CRC code has $d_{\min} = 20$, all the other candidates have $d_{\min} < 20$ and are eliminated before the end of the $d = 18$ round. Since the lengths of single errors $l_{d,i}$ for $d < 20$, ranging from 7 to 43, are much shorter than $n + \nu$, a candidate that has fewer types of dominant undetectable error events will have fewer dominant undetectable errors in total. In other words, when undetectable single errors dominate and information length k is long enough, the best CRC code should possess the smallest a_d^{ZZ} .

When $d_{\min} \geq 2d_{\text{free}}$, the dominant undetectable errors include double errors. In this case, a smaller a_d^{ZZ} does not mean a better code because it only considers single errors.

TABLE II
STANDARD CRC CODES OR CRC CODES RECOMMENDED BY KOOPMAN
AND CHAKRAVARTY (K&C) [5], AND THE BEST CRC CODES FOR
CONVOLUTIONAL CODE $(133, 171)_8$ WITH $k = 1024$ BITS.

Name	Gen. Poly.	Undetectable Single Distance Spectrum a_d^{ZZ}							
		d	10	12	14	16	18	20	22
K&C-3	0x5		1	5	19	170	941	5050	29290
Best-3	0x7		0	7	24	169	879	5111	29363
CRC-4	0xF		1	2	11	79	464	2504	14719
Best-4	0xD		0	1	17	91	462	2537	14674
CRC-5	0x15		1	2	9	52	267	1378	8005
Best-5	0x11		0	0	4	52	230	1257	7275
CRC-6	0x21		0	1	4	21	124	572	3659
Best-6	0x29		0	0	1	22	124	641	3650
CRC-7	0x48		0	0	1	14	55	298	1877
Best-7	0x47		0	0	0	7	70	322	1867
CRC-8	0xEA		0	0	0	4	36	174	871
Best-8	0x89		0	0	0	1	29	177	938
K&C-9	0x167		0	0	0	4	13	73	477
Best-9	0x177		0	0	0	0	14	104	437
CRC-10	0x319		0	0	0	1	8	41	239
Best-10	0x314		0	0	0	0	3	49	223
CRC-11	0x5C2		0	0	0	0	7	17	107
Best-11	0x507		0	0	0	0	0	24	113
CRC-12	0xC07		0	0	0	0	3	12	48
Best-12	0xA10		0	0	0	0	0	4	66
K&C-13	0x102A		0	0	0	0	1	7	36
Best-13	0x1E0F		0	0	0	0	0	1	29
K&C-14	0x21E8		0	0	0	0	1	2	15
Best-14	0x314E		0	0	0	0	0	0	11
K&C-15	0x4976		0	0	0	0	1	1	6
Best-15	0x604C		0	0	0	0	0	0	3
CRC-16	0xA001		0	0	0	0	0	1	3
Best-16	0x8E61		0	0	0	0	0	0	1
Original Distance Spectrum a_d			11	38	193	1331	7275	40406	234969

For example, the degree-16 polynomials 0xF8F1 and 0x8E61 both have $d_{\min} = 22$. The former has $a_{22}^{ZZ} = 0$ while the latter has $a_{22}^{ZZ} = 1$. However, at $d = 22$, the former has so many (2860) undetectable double errors that the number is greater than the total count of undetectable single and double errors (1011 + 1424) of the latter, when the information length is $k = 1024$ bits. However, when $k = 512$ bits, the former has fewer undetectable errors and becomes optimal.

Therefore, different information lengths may lead to different optimal CRC designs. The authors of [5] have included the information length k as a key design parameter and proposed a methodology to determine “good” CRC codes for a range of k . The same rule can be applied here. First, find the CRC polynomials possessing the largest d_{\min} for the longest k . Then, consider shorter k and keep the CRC polynomials having the largest d_{\min} , which might increase as k decreases. Table III shows the best CRC codes for $k = 256, 512$, or 1024 bits and identifies the “good” codes for this range of information lengths. The bold numbers indicate the k for which the code is designed. For example, the code 0xA10 is the best degree-12 code at lengths $k = 256$ and 1024 bits. Note that the codes with degrees $m \leq 11$ are not shown since undetectable single errors dominate and thus the best CRC codes for these k are identical. In fact, the best CRC code for the largest k is usually “good” for shorter k . In our case,

TABLE III
THE BEST CRC CODES FOR CONVOLUTIONAL CODE $(133, 171)_8$ WITH
 $k = 256, 512$, OR 1024 BITS. BOLD NUMBERS INDICATE THE k FOR
WHICH THE CODE IS DESIGNED. THE “GOOD” CODES FOR THIS RANGE
OF k ARE SPECIFIED.

Degree	Gen. Poly.	$(d_{\min}, \text{count of undetectable errors at } d_{\min})$			
		k	256 bits	512 bits	1024 bits
12	0xA10 (good)		(20, 1664)	(20, 5525)	(20, 17732)
	0x8DC (good)		(20, 1904)	(20, 4748)	(20, 19283)
13	0x18F6 (good)		(20, 169)	(20, 1474)	(20, 7452)
	0x1E0F (good)		(20, 289)	(20, 1187)	(20, 5301)
14	0x2E20		(22, 3196)	(20, 520)	(20, 2056)
	0x314E (good)		(22, 4698)	(22, 12324)	(20, 198)
15	0x6D80		(22, 962)	(20, 253)	(20, 765)
	0x76AD		(22, 1210)	(22, 2808)	(20, 1382)
	0x604C (good)		(22, 1767)	(22, 4414)	(22, 13329)
16	0xA219		(24, 7396)	(22, 316)	(20, 454)
	0xF8F1 (good)		(24, 9823)	(22, 219)	(22, 2860)
	0x8E61		(22, 243)	(22, 629)	(22, 2435)

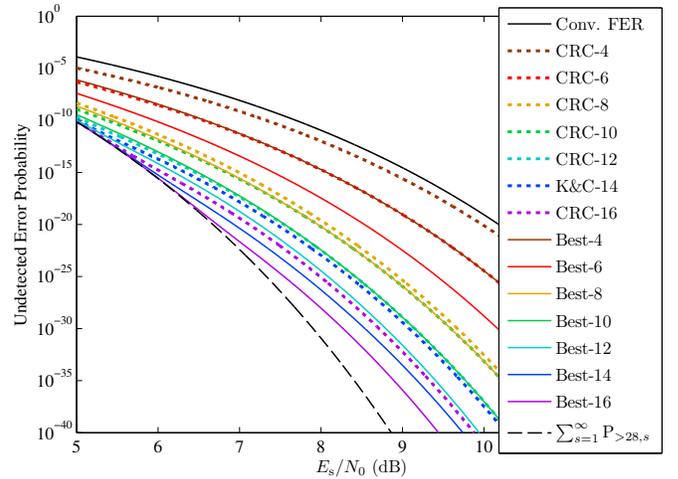


Fig. 6. The undetected error probability of the existing and best CRC codes for convolutional code $(133, 171)_8$ with information length $k = 1024$ bits computed using the construction method.

the best $k = 1024$ CRC codes have no more than 0.1dB loss compared to the best $k = 256$ and the best $k = 512$ CRC codes at information lengths $k = 256$ and 512 bits, respectively, except for the degree-16 codes.

In Fig. 6, the bounds of undetected error probability of the existing and best CRC codes for information length $k = 1024$ bits are shown. For clarity, only even degrees of the CRC codes are displayed. The upper bound of the original convolutional code FER without any CRC code, calculated using transfer function techniques [14], is plotted as a reference. In our calculation, the search depth limit of the exclusion method is $\hat{d} = 22$ and not enough for high degree CRC codes. Therefore, the construction method (16) was used with $\hat{d} = 28$.

The probability sum of all large-distance terms calculated using (11c) is also plotted to illustrate that the large-distance terms really are negligible even assuming they are all undetectable, except for the best degree-16 CRC code at SNR below 7 dB and some other codes at SNR below 6 dB. Note that since the operation $e^P - 1$ in (11c) causes non-negligible rounding errors in the high SNR region, it was approximated

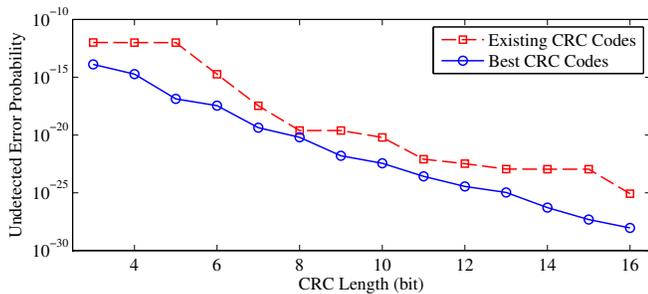


Fig. 7. The undetected error probability of the existing and best CRC codes for convolutional code $(133, 171)_8$ with information length $k = 1024$ bits at SNR = 8 dB.

by the first ten terms, which results a similar expression as (11b) but only carried out for $1 \leq s \leq 10$.

Since $\hat{d} < 3d_{\text{free}}$, it is not necessary to evaluate triple or higher order errors under this truncation. It is clearly seen from the figure that the best degree- m CRC codes found by our procedure outperform the existing degree- m codes for all m . Furthermore, their performance is either better than or similar to the existing degree- $(m + 2)$ code except for $m = 6$. In other words, the proposed design can typically save 2 check bits while keeping the same error detection capability.

We can compare the error detection capability of all codes at a fixed SNR. If we draw a vertical line at SNR = 8 dB on Fig. 6, the intersections can be plotted along with the associated CRC lengths m in Fig. 7. The largest reduction of undetected error probability is about five orders of magnitude at $m = 5$ where the existing CRC code has an undetected error probability of 1.01×10^{-12} and the newly designed CRC code has an undetected error probability of 1.36×10^{-17} , based on the analysis. Note that these numbers are calculated at 8 dB SNR, and the reduction gets more significant as SNR increases.

Since the existing CRC codes are not tailored to the convolutional code, a higher degree code does not necessarily have a better performance. We observe that there are three almost horizontal regions for the existing CRCs with degrees m ranging from 3 to 5, from 8 to 9, and from 13 to 15. The reason is that the codes in each region have similar number of dominant undetectable errors as can be seen in Table II. In contrast, the CRC codes designed using our procedure show steady improvement as the degree increases.

The existing and best CRC codes can also be compared in terms of the required CRC length to achieve a certain undetected error probability. Assume our target is to reach undetected error probability below 10^{-25} . This can be shown by drawing a horizontal line at the target probability level on Fig. 6 and plotting the CRC lengths associated to the crossed points as a function of SNR as in Fig. 8. For most of the SNR levels, the best CRC codes requires two fewer check bits than the existing CRC codes to achieve the same error detection capability. At SNR around 10.5 dB, the best CRC code requires three fewer check bits than the existing CRC code. Since the existing code required six check bits, this is a 50% reduction.

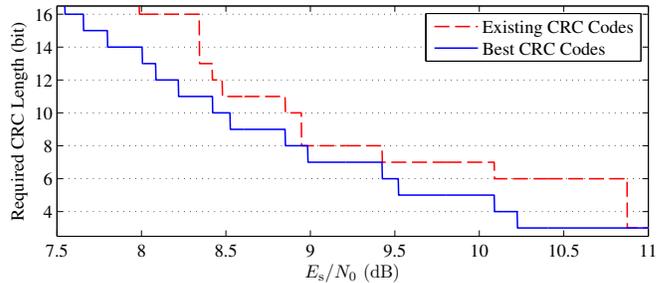


Fig. 8. The required CRC lengths for the existing and best CRC codes for convolutional code $(133, 171)_8$ with information length $k = 1024$ bits to achieve undetected error probability below 10^{-25} .

VI. CONCLUSION

A good CRC code in a convolutionally coded system should minimize the undetected error probability. To calculate this probability, two methods based on distance spectrum are proposed. The exclusion method starts with all possible single and multiple error patterns of the convolutional code and excludes them one by one by testing if they are detectable. In the construction method, undetectable errors are mapped to error events of an equivalent convolutional code, which is the combination of the CRC code and the original convolutional code. The computer search for error events in the construction method does not need to record the error patterns and thus can go deeper than the search in the exclusion method. However, the construction method could encounter difficulties while dealing with high-degree CRC codes. Moreover, the construction method is generally applicable to the performance analysis of catastrophic convolutional codes.

We also propose a search procedure to identify the best CRC codes for a specified convolutional encoder and information length. A candidate CRC code is excluded if it has more low-distance undetectable errors. Therefore, the best CRC polynomial is guaranteed to have the fewest dominant undetectable errors and minimizes the probability of undetected error when SNR is high enough. When undetectable double errors dominate, the choice of the best CRC polynomial is more dependent on the information length. In an example application of the design procedure for the popular 64-state convolutional code with information length $k = 1024$, new CRC codes provided significant reduction in undetected error probability compared to the existing CRC codes with the same degrees. With the proposed design, we are able to save two check bits in most cases while having the same error detection capability.

It is an open problem to generalize this work to other error-correcting codes, such as turbo or low-density parity-check (LDPC) codes. The construction method can only be applied when the encoder has the convolutional structure. We note that convolutional LDPC codes have such a structure. For turbo codes, the construction method can be used to analyze one of the recursive systematic convolutional (RSC) codes but analyzing the other RSC code is not straightforward due to the existence of the interleaver. Nevertheless, the exclusion method can be generalized to any other error-correcting codes

if the dominant error events are identified.

APPENDIX A

EFFICIENT SEARCH FOR UNDETECTABLE DOUBLE ERRORS

In (4) and (5), the indicator function is evaluated for every g_1 given $e_1(x)$ and $e_2(x)$. However, there is a more efficient way to check the divisibility and can save a lot of computation time when $n + \nu$ is large.

The remainder of any polynomial in $\mathbf{GF}(2)[x]$ divided by $p(x)$ forms a quotient ring $\mathbf{GF}(2)[x]/p(x)$. Two polynomials mapped to the same element of the quotient ring are called *congruent*. If $x^{g_1+l_2}e_1(x)$ is congruent to $e_2(x)$ modulo $p(x)$, then their combination forms an undetectable double error. To characterize the remainder of $x^{g_1+l_2}e_1(x)$ modulo $p(x)$, we apply the concept of cyclotomic coset [18], which is originally defined for integers, to polynomials. For polynomials in $\mathbf{GF}(2)[x]$, define x -cyclotomic coset modulo $p(x)$ containing $e(x)$ as

$$\mathbf{C}_{e(x)} = \{x^h e(x) \pmod{p(x)} \mid h = 0, 1, \dots\}, \quad (17)$$

which includes the remainder of all possible offsets of $e(x)$ divided by $p(x)$. One can verify that two cyclotomic cosets are either identical or disjoint, so all of the distinct x -cyclotomic cosets modulo $p(x)$ form a partition of quotient ring $\mathbf{GF}(2)[x]/p(x)$. For example, consider a degree-2 primitive polynomial $p(x) = x^2 + x + 1$, and we have $\mathbf{C}_0 = \{0\}$ and $\mathbf{C}_1 = \{1, x, x + 1\}$ forming a partition of $\mathbf{GF}(2)[x]/p(x)$; consider a degree-2 non-primitive polynomial $p(x) = x^2 + 1$, and we have $\mathbf{C}_0 = \{0\}$, $\mathbf{C}_1 = \{1, x\}$, and $\mathbf{C}_{x+1} = \{x + 1\}$ forming a partition of $\mathbf{GF}(2)[x]/p(x)$. In both cases, \mathbf{C}_0 is trivial since it only contains the “zero” element. In the primitive case, \mathbf{C}_1 is the only non-trivial cyclotomic coset and its cardinality is $|\mathbf{C}_1| = |\mathbf{GF}(2)[x]/p(x)| - 1 = 2^m - 1$. In the non-primitive case, there are multiple non-trivial cyclotomic cosets and their sizes are smaller than $2^m - 1$. In fact, there is only one unique non-trivial cyclotomic coset if $p(x)$ is a primitive polynomial.

It is obvious that if $e_1(x)$ and $e_2(x)$ belong to different cyclotomic cosets, there is no way to have a g_1 that makes $x^{g_1+l_2}e_1(x)$ congruent to $e_2(x)$ modulo $p(x)$. In other words, it is unnecessary to check whether any g_1 creates an undetectable double error with the specific $e_1(x)$ and $e_2(x)$. If $e_1(x)$ and $e_2(x)$ belong to the same cyclotomic coset $\mathbf{C}_{e_1(x)}$, only one proper $g_1 \in [0, |\mathbf{C}_{e_1(x)}| - 1]$ can make $x^{g_1+l_2}e_1(x) + e_2(x)$ divisible by $p(x)$. Denote this particular g_1 as g'_1 . Once we find g'_1 , all possible g_1 that create undetectable double errors are just $g_1 = g'_1 + u |\mathbf{C}_{e_1(x)}|$ for non-negative integer u satisfying $g_1 + l_1 + l_2 \leq n + \nu$.

Note that when $e_1(x) = e_2(x)$, they will belong to the same cyclotomic coset no matter what CRC generator polynomial $p(x)$ is. That is to say, no CRC code is able to detect such double error if these two error events have a proper gap g_1 . Fortunately, the smallest proper gap is $g'_1 = -l_2 \pmod{|\mathbf{C}_{e_1(x)}|}$ so the total length of the undetectable double error is $|\mathbf{C}_{e_1(x)}| + l_1$. Hence, when $n + \nu$ is small enough, such double error will never occur. On the other hand, when $n + \nu$ is large, d_{\min} , which is the shortest distance of the undetectable errors, will be upper bounded by $2d_{\text{free}}$.

APPENDIX B

THE RELATIONSHIP BETWEEN Δ_i AND $\mathbf{C}_{e(x)}$

Define the state of the equivalent code at time $n - g$ as $q'_g(x)$, which is a polynomial with maximum degree $m + \nu - 1$ representing consecutive $m + \nu$ bits from the $x^{g+m+\nu-1}$ term to the x^g term in $x^m q'(x)$ for $g \in [-\nu, n]$. The corresponding state of the original code is given by the coefficients of the terms from $x^{m+\nu-1}$ to x^m in polynomial $q'_g(x)p(x)$. Let $q'_{g,u}$ and p_u be the coefficients of x^u in $q'_g(x)$ and $p(x)$, respectively. Then the coefficient of x^u for $u \in [m, m + \nu - 1]$ in $q'_g(x)p(x)$ is given by

$$\sum_{v=0}^m q'_{g,u-v} p_v. \quad (18)$$

Assume that the all-zero codeword is sent, i.e. $q(x) = 0$, and the trellis path enters a detectable-zero state \mathbf{S}_i^D at time $n - g$. Also, let $e(x)$ be a polynomial of degree smaller than or equal to $n - g - 1$ representing the length- $(n - g)$ input sequence of the original convolutional encoder from the beginning to time $n - g$, and it is given by the coefficients from the x^{n-1} term to the x^g term in $q'(x)p(x)$. Since the state at time $n - g$ is \mathbf{S}_i^D , $e(x)$ must be non-divisible by $p(x)$.

The remainder of $e(x)$ divided by $p(x)$ is given by

$$e(x) \pmod{p(x)} = \sum_{u=0}^{m-1} x^u \sum_{v=u+1}^m q'_{g,m+u-v} p_v, \quad (19)$$

which is totally governed by $q'_g(x)$, or \mathbf{S}_i^D . If the remainder is known, the bits $q'_{g,v}$ for $v \in [0, m - 1]$ can be solved uniquely through back substitution for $u = m - 1, m - 2, \dots, 0$ because $p_m = 1$. Furthermore, the whole polynomial $q'_g(x)$, or \mathbf{S}_i^D , can be solved by letting (18) equal to zero for $u = m + \nu - 1, m + \nu - 2, \dots, m$ because the state of the original code is just ν zeros. Hence, the remainder of $e(x)$ divided by $p(x)$ determines a detectable-zero state \mathbf{S}_i^D , and vice versa. Furthermore, each of the $2^m - 1$ non-zero elements in $\mathbf{GF}(2)[x]/p(x)$ corresponds to a unique state in \mathbf{S}^D .

To find all states in Δ_i , we can specify $q'_{g-h,0}$, the input bit to the equivalent encoder at time $n - g + h$, for $h = 1, 2, \dots$ such that the input bits to the original convolutional encoder after time $n - g$ are all zeros and thus the following states are in \mathbf{S}^D . By doing so, we know that the polynomials $q'_{g-h}(x)$ will represent the states in Δ_i . This procedure is finished when certain $q'_{g-h}(x)$ represents \mathbf{S}_i^D again. During this procedure, the corresponding input sequence to the original encoder from the beginning to time $n - g + h$ is simply $x^h e(x)$ because the input bits after time $n - g$ are all zeros. By the definition given in (17), the remainder of $x^h e(x)$ divided by $p(x)$ is an element of $\mathbf{C}_{e(x)}$. In addition, we know that this remainder corresponds to a state in Δ_i . Therefore, Δ_i and $\mathbf{C}_{e(x)}$ contain the same elements but just represented in different forms.

ACKNOWLEDGMENT

The authors would like to thank Dr. A. R. Williamson with Communications Systems Laboratory (CSL) at University of California, Los Angeles (UCLA) for useful discussions and Mr. K. Vakilinia with CSL at UCLA for his kind help.

REFERENCES

- [1] T. Klove and M. Miller, "The detection of errors after error-correction decoding," *IEEE Trans. Commun.*, vol. 32, no. 5, pp. 511–517, May 1984.
- [2] S. Leung-Yan-Cheong, E. R. Barnes, and D. Friedman, "On some properties of the undetected error probability of linear codes," *IEEE Trans. Inform. Theory*, vol. 25, no. 1, pp. 110–112, Jan. 1979.
- [3] P. Kazakov, "Fast calculation of the number of minimum-weight words of CRC codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 3, pp. 1190–1195, Mar. 2001.
- [4] R.-D. Lin and W.-S. Chen, "Fast calculation algorithm of the undetected errors probability of CRC codes," in *Proc. 2005 IEEE 19th Int. Conf. Advanced Inform. Networking and Applicat. (AINA)*, vol. 2, Mar. 2005, pp. 480–483.
- [5] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *Proc. 2004 IEEE Int. Conf. Dependable Syst. and Networks (DSN)*, Jun. 2004, pp. 145–154.
- [6] G. Castagnoli, S. Brauer, and M. Herrmann, "Optimization of cyclic redundancy-check codes with 24 and 32 parity bits," *IEEE Trans. Commun.*, vol. 41, no. 6, pp. 883–892, Jun. 1993.
- [7] P. Koopman, "32-bit cyclic redundancy codes for internet applications," in *Proc. 2002 IEEE Int. Conf. Dependable Syst. and Networks (DSN)*, 2002, pp. 459–468.
- [8] S.-L. Shieh, P.-N. Chen, and Y. S. Han, "Flip CRC modification for message length detection," *IEEE Trans. Commun.*, vol. 55, no. 9, pp. 1747–1756, Sep. 2007.
- [9] M. Ghosh and F. LaSita, "Puncturing of CRC codes for IEEE 802.11ah," in *Proc. 2013 IEEE 78th Veh. Technology Conf. (VTC Fall)*, Sep. 2013, pp. 1–5.
- [10] R. Wang, W. Zhao, and G. Giannakis, "CRC-assisted error correction in a convolutionally coded system," *IEEE Trans. Commun.*, vol. 56, no. 11, pp. 1807–1815, Nov. 2008.
- [11] Y. Wei, M. Jiang, B. Xia, W. Chen, and Y. Yang, "A CRC-aided hybrid decoding algorithm for turbo codes," *IEEE Wireless Commun. Lett.*, vol. 2, no. 5, pp. 471–474, Oct. 2013.
- [12] M. El-Khamy, J. Lee, and I. Kang, "Detection analysis of CRC-assisted decoding," *IEEE Wireless Commun. Lett.*, vol. 19, no. 3, pp. 483–486, Mar. 2015.
- [13] A. J. Viterbi and J. K. Omura, *Principles of digital communication and coding*. Courier Dover Publications, 2009.
- [14] C.-Y. Lou and B. Daneshrad, "PER prediction for convolutionally coded MIMO OFDM systems—an analytical approach," in *Proc. 2012 IEEE Military Commun. Conf. (MILCOM)*, Oct. 2012, pp. 1–6.
- [15] M. Cedervall and R. Johannesson, "A fast algorithm for computing distance spectrum of convolutional codes," *IEEE Trans. Inform. Theory*, vol. 35, no. 6, pp. 1146–1159, Nov. 1989.
- [16] A. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. 19, no. 5, pp. 751–772, Oct. 1971.
- [17] E. Malkamaki and H. Leib, "Evaluating the performance of convolutional codes over block fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1643–1646, Jul. 1999.
- [18] S. Ling, *Coding theory: a first course*. Cambridge University Press, 2004.