Efficient Computation of Convolutional Decoder Reliability Without a CRC

A. Baldauf, A. Belhouchat, N. Wong, R. D. Wesel
University of California, Los Angeles, CA
{ambaldauf19,abelhouchat}@gmail.com {nsc.wong,wesel}@ucla.edu

***Abstract*—The reliability-output Viterbi algorithm (ROVA) of Raghavan and Baum computes the probability that the codeword selected by Viterbi decoding is in error, allowing unreliable decoding to be identified without the overhead of a cyclic redundancy check (CRC). ROVA can be used as a stopping criterion for variable length (VL) codes with feedback. Separately, Polyanskiy et al. proposed accumulated information density (AID) as a stopping criterion for VL codes for computation of random coding bounds. This paper compares the accuracy and complexity of ROVA and AID. It turns out that AID is far less accurate than ROVA. This paper proposes codeword information density (CID), which modifies AID to improve its accuracy and leads to a lower-complexity implementation of ROVA. The paper concludes with an analytical expression for the random variable describing the correct decoding probability computed by ROVA and uses this expression to characterize how the probabilities of correct decoding, undetected error, and negative acknowledgement behave as a function of the selected threshold for reliable decoding.**

I. Introduction

Cyclic redundancy checks (CRCs) are often used to detect errors in convolutional codewords [1]–[4]. CRCs play an important role in many incremental redundancy hybrid ARQs [4]–[7] but add overhead that can be significant for short block-lengths. One alternative is the reliability-output Viterbi algorithm (ROVA) [8]. ROVA computes the probability that a Viterbi decoding decision is in error. This allows the receiver to set a threshold on the ROVA-computed probability to achieve a target undetected (codeword) error rate (UER) without requiring a CRC.

ROVA was used in [9]–[11] to decide whether to request additional feedback in an incremental redundancy hybrid ARQ without the need for a CRC. For [9], ROVA was adapted as described in [12] for tail-biting convolutional codes. Although ROVA calculates codeword error probability exactly, it suffers from high complexity. Fricke and Hoeher [11] developed an approximation of ROVA that reduces complexity.

Another alternative to ROVA for controlling an incremental redundancy hybrid ARQ is information density [13], which computes an empirical estimate of the mutual information of the channel. Symbol-wise accumulated information density (AID) as proposed by Polyanskiy et al. [13] sums the information density of each received symbol in a prospective codeword, providing a metric of codeword reliability with a much lower complexity than ROVA.

This paper compares the accuracy of ROVA with that of symbol-wise AID. After observing the low accuracy of symbol-wise AID, we propose codeword information density (CID) as a modification to symbol-wise AID. The CID also computes an information density, but instead of adding the density of each symbol, it computes a single information density for the entire received codeword. CID gives better accuracy than AID and turns out to be equivalent to ROVA.

Sec. II reviews the ROVA algorithm of [8]. Sec. III reviews AID of [13] and shows that it is much less predictive of reliable decoding than ROVA. Sec. IV proposes CID as a modification of AID, shows that CID is equivalent to ROVA, and uses the CID perspective to compute ROVA with significantly less complexity than [8]. Sec. V presents an analytical expression for the random variable describing the correct decoding probability computed by ROVA and uses this expression to characterize how the probabilities of correct decoding, undetected error, and negative acknowledgement behave as a function of the selected threshold for reliable decoding. Sec. VI concludes the paper.

II. The Reliability Output Viterbi Algorithm

ROVA finds the probability that the n_c-symbol codeword $\hat{x}^{n_c}$ selected by maximum likelihood decoding is also the transmitted codeword $x_t^{n_c}$ (where x^{n_c} denotes the sequence of codeword symbols $x_1, \ldots, x_{n_c}$). Given a received noisy sequence $y^{n_c} = x^{n_c} + z^{n_c}$, the probability that $\hat{x}^{n_c} = x_t^{n_c}$ is

$$P(\hat{x}^{n_c} = x_t^{n_c} | y^{n_c}) = \frac{P(\hat{x}^{n_c}) f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{C}} P(x^{n_c}) f_{Y|X}(y^{n_c}|x^{n_c})} \quad (1)$$

$$= \frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{C}} f_{Y|X}(y^{n_c}|x^{n_c})} \quad (2)$$

where $\mathcal{C}$ is the set of valid codewords and f is the conditional pdf of the received sequence given the transmitted sequence. The simplification from (1) to (2) follows from the assumption that all possible codewords are *a priori* equally likely.

The natural application of ROVA is to set a threshold on the ROVA value (2) computed as in [8] and consider codewords with a ROVA value below the threshold as erasures because they are not sufficiently reliable. Fig. 1 shows how varying the threshold can control the UER. The empirical UER achieved by Viterbi with a ROVA threshold is shown for each threshold value from $P(\hat{x}^{n_c} = x_t^{n_c}|y^{n_c}) = 0.7$ to $1-10^{-4}$ in increments of 10^{-4}.

Also shown is the expected UER associated with the threshold, which is computed as the UER implied by the empirical average of observed ROVA values. There is excellent

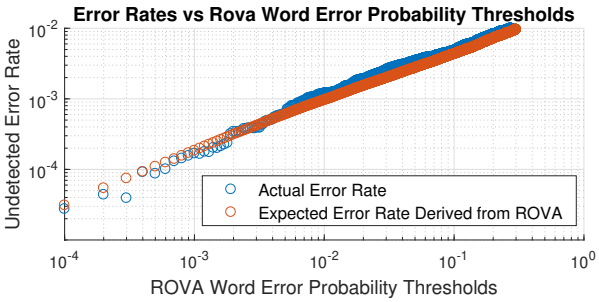

Fig. 1: Graph of empirical undetected (codeword) error rate (UER) as a function of ROVA threshold on UER for 100,000 decodings of a 4-state, rate-1/2 convolutional code with $k = 128$ message bits at SNR 4.5 dB.

agreement between the observed and expected UER. The average $P(\hat{x}^{n_c} = x_t^{n_c}|y^{n_c})$ will be substantially higher than the threshold because the threshold is the lowest acceptable value. As a result, the UER achieved when applying a threshold is significantly below the the UER that corresponds to codewords that have $P(\hat{x}^{n_c} = x_t^{n_c}|y^{n_c})$ exactly equal to the threshold.

A. Computing ROVA as in [8]

Algorithm 1 describes the procedure for computing the ROVA value (2) computed as in [8]. Consider a trellis with 2^v states defined by the set $\mathcal{S} = \{0, 1, \ldots 2^v - 1\}$. Let s_m be the trellis state after the m^{th} transmitted symbol. Let the trellis be initialized to state $s_0 = 0$ and assume that terminating input bits drive the state back to $s_n = 0$ at the last (n^{th}) transmitted symbol x_n. For each received symbol y_m for $m \in \{1, 2, \ldots n_c\}$ and for every possible value i of the the trellis state s_m in $\mathcal{S}$, two probabilities are computed in [8]: P_i^m and $\bar{P}_i^m$. We now define these two probabilities.

Let $\hat{x}^m(i)$ to be the symbol sequence corresponding to the Viterbi survivor path terminating at state i after the m^{th} transmitted symbol. P_i^m is $P(s_m = i, \hat{x}^m(i) = x_t^m|y^m)$, which is the probability that i is the correct state after the m^{th} transmitted symbol and that the Viterbi algorithm has correctly identified the survivor path to state i so that $\hat{x}^m(i) = x_t^m$. $\bar{P}_i^m$ is $P(s_m = i, \hat{x}^m(i) \neq x^m|y^m)$, which is the probability that i is the correct state at symbol m but the Viterbi algorithm has not correctly identified the survivor path to state i so that $\hat{x}^m(i) \neq x^m$. Thus $\bar{P}_i^m$ is the probability that Viterbi decoding has incorrectly pruned away the transmitted sequence x_t^m, which is a path to state i, after the m^{th} transmitted symbol.

For each m, Algorithm 1 makes use of the scaling factor

$$\Delta_m = \frac{\sum_{\mathcal{T}} \gamma_1\gamma_2 \cdots \gamma_m}{\sum_{\mathcal{T}} \gamma_1\gamma_2 \cdots \gamma_{m-1}}, \tag{5}$$

where γ_m is the branch metric for the m^{th} symbol associated with one of the paths in the trellis $\mathcal{T}$ so that $\gamma_1\gamma_2 \ldots \gamma_m$ is a path metric for one of the paths in the trellis $\mathcal{T}$ and $\sum_{\mathcal{T}} \gamma_1\gamma_2 \ldots \gamma_m$ is the sum of all the path metrics in the first m stages of trellis $\mathcal{T}$ regardless of whether they are survivors in the Viterbi algorithm.

When $m = n_c$, the state has been forced to zero by terminating input bits so that $P_0^m + \bar{P}_0^m = 1$, and $\bar{P}_0^{n_c} = 1 - P_0^{n_c}$ is the probability that the codeword selected by Viterbi is incorrect. For $m < n_c$ and a particular state $i \in \mathcal{S}$, $P_i^m + \bar{P}_i^m$ will generally be less than one. These values must be summed over all states to account for all the probability:

$$\sum_{i=0}^{2^v-1} \left(P_i^m + \bar{P}_i^m\right) = 1. \tag{6}$$

Algorithm 1: The ROVA Algorithm as described in [8].

Initialization: For $i, j \in \mathcal{S}$, let $\mathcal{T}_m$ be the set of valid trellis branches possible during transmission of the m^{th} symbol. Each such trellis branch is defined by the ordered pair (i, j) where i is the origin state and j is the destination state. Note that this set is smallest at $m = 1$ when there are only 2^k branches emanating from $s_0 = 0$ to s_1 and at $m = n_c$ when there are only 2^k branches entering $s_n = 0$. Initialize $m = 0$, $P_0^0 = 1$ and $\bar{P}_0^0 = 0$

Iterations: The calculation of (2) in [8] proceeds as follows:

1) $m = m + 1$
2) For each valid branch $(i, j) \in \mathcal{T}_m$ compute metrics
$$\gamma_m(i, j) = f\left(y_m|x_m(i, j)\right) \tag{3}$$
where $x_m(i, j)$ is the symbol transmitted on (i, j).
3) Compute the scaling factor
$$\Delta_m = \sum_{(i,j)\in\mathcal{T}_m} \gamma_m(i, j)(P_i^{m-1} + \bar{P}_i^{m-1}). \tag{4}$$
4) For each $j \in \mathcal{S}$ with branches $(i, j) \in \mathcal{T}_m$ where Viterbi has identified branch (i^*, j) to be the survivor branch to j compute
$$P_j^m = \Delta_m^{-1}\gamma_m(i^*, j)P_{i^*}^{m-1}$$
$$\bar{P}_j^m = \Delta_m^{-1} \sum_{(i,j)\in\mathcal{T}_m} \gamma_m(i, j)(P_i^{m-1} + \bar{P}_i^{m-1}) - P_j^m$$
5) if $m = n_c$ conclude by reporting the ROVA value of $P_0^{n_c}$ and the probability of codeword error as $\bar{P}_0^{n_c} = 1 - P_0^{n_c}$, otherwise, go to step 1.

III. Accumulated Information Density

Polyanskiy et al. [13] used a threshold on information density to decide when to terminate random codes to derive bounds on throughput for codes with finite blocklength. The information density of a received symbol y_i with respect to a selected codeword symbol $\hat{x}_i$ is computed as

$$i(y_j, \hat{x}_j) = \log_2 \frac{f_{Y|X}(y_j|\hat{x}_j)}{f_Y(y_j)}. \tag{7}$$

In (7), $f_Y(y_j)$ is computed assuming that each possible symbol $x \in \mathcal{X}$ is drawn i.i.d. according to an input distribution, either a pdf $f_X(x)$ or a pmf $P_X(x)$. For practical communication systems in which a convolutional code is used in conjunction with a constellation of possible transmitted symbols, the input alphabet $\mathcal{X}$ is finite and is exactly the constellation. For a typical encoder (without shaping), each constellation point is equally likely so that $P_X(x) = |\mathcal{X}|^{-1}$.

Accumulated information density (AID) sums (7) for each symbol in the codeword to produce $i_{\text{AID}}(y^{n_c}, \hat{x}^{n_c})$ as follows:

$$i_{\text{AID}}(y^{n_c}, \hat{x}^{n_c}) = \sum_{j=1}^{n_c} i(y_j, \hat{x}_j) \tag{8}$$

$$= \sum_{j=1}^{n_c} \log_2 \left(\frac{f_{Y|X}(y_j|\hat{x}_j)}{f_Y(y_j)} \right) \tag{9}$$

$$= \log_2 \left(\frac{\prod_{j=1}^{n_c} f_{Y|X}(y_j|\hat{x}_j)}{\prod_{j=1}^{n_c} f_Y(y_j)} \right) \tag{10}$$

$$= \log_2 \left(\frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{X}^{n_c}} |\mathcal{X}|^{-n} f_{Y|X}(y^{n_c}|x^{n_c})} \right) \tag{11}$$

where $\mathcal{X}^{n_c}$ is the set of all sequences of n_c symbols. For AID, the denominator in (11) includes every possible sequence of n_c symbols from the alphabet (constellation) $\mathcal{X}$. However, only sequences that are actually codewords could have been transmitted. Including all possible sequences allows the computation of AID to be symbol-wise and thus much simpler than ROVA, but it introduces an inaccuracy.

Algorithm 2 below provides a procedure for computing i_{AID}. Let N_s be the number of states $|\mathcal{S}|$ and N_b be the number of branches entering each state. When $N_b = 2$, Algorithm 1 requires about $6N_s$ multiplications per trellis stage, but Algorithm 2 requires only about N_s multiplications.

Algorithm 2: Computation of i_{AID}.

Initialization: For $i, j \in \mathcal{S}$, let $\mathcal{T}_m$ be the set of valid trellis branches as defined in *Algorithm 1*. Initialize $m = 0$, $\Gamma_0^0 = 1$, $\Pi(0) = 1$.

Iterations:

1) $m = m + 1$
2) Compute branch metrics $\gamma_m(i, j)$ as in Algorithm 1.
3) For each $j \in \mathcal{S}$ with branches $(i, j) \in \mathcal{T}_m$ where Viterbi has identified survivor branch (i^*, j) compute

$$\Gamma_m^j = \Gamma_{m-1}^{i^*} \gamma_m(i^*, j)\,. \tag{12}$$

4) Compute $f_Y(y_m) = \sum_{x \in \mathcal{X}} |\mathcal{X}|^{-1} f(y_m|x)$ and

$$\Pi(m) = \Pi(m-1) f_Y(y_m) \tag{13}$$

5) if $m = n_c$ conclude by reporting

$$i_{\text{AID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2 \frac{\Gamma_n^0}{\Pi(n)}\,, \tag{14}$$

otherwise, go to step 1.

Figs. 2 and 3 compare the efficacy of ROVA and AID by plotting histograms of metric values for correctly and incorrectly decoded sequences. For AID, the sequences are organized by their accumulated information density. For ROVA, they are organized by word-error probability. The better separation (smaller overlap area) seen in Fig. 2 as compared to Fig. 3 between the histograms for correctly and incorrectly decoded sequences suggests that ROVA is more effective at distinguishing correct codewords from incorrect codewords.

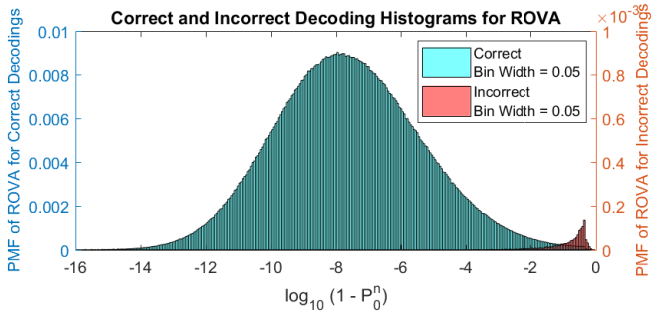

Fig. 2: Normalized histograms of ROVA error values for correct and incorrect decodings for 1 million decodings for a 64-state, rate-1/3 convolutional code with $k = 32$ message bits and SNR $= 1.0$dB. The minimal overlap between the incorrect and correct decodings indicates that setting a decision threshold using ROVA is an effective way to reduce undetected errors.

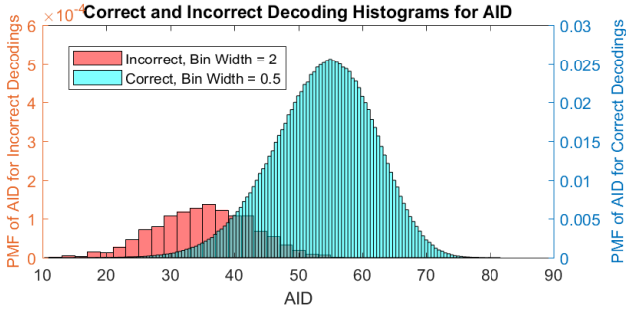

Fig. 3: Normalized histograms of correct and incorrect AID values for 1 million decodings for the same scenario as Fig. 2. There is considerable overlap between the incorrect and correct decodings.

Fig. 4 shows UER versus throughput for ROVA and AID, where the throughput is a function of the threshold that determines whether to accept the Viterbi result as reliable. Throughput is defined as the ratio of correctly decoded sequences that passed the threshold to the total number of received sequences. Fig. 4 confirms the poor performance of AID that was suggested in Figs. 2 and 3 . For a given target UER, AID supports a much lower throughput than ROVA. Despite its lower complexity, AID turns out to be too inaccurate to use as a decoder reliability metric in practice.

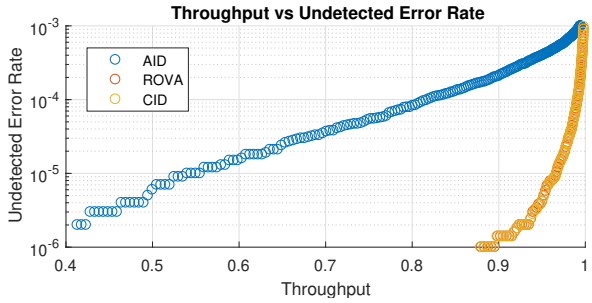

Fig. 4: Undetected error rate (UER) as a function of throughput for ROVA, AID, and CID showing the operating points of (throughput,UER) achievable with thresholds on ROVA, AID, and CID metrics. ROVA and CID give identical performance as expected by (19).

IV. Codeword Information Density: ROVA Redux

We propose a new metric, the codeword information density (CID), which is computed for the codeword selected by Viterbi as follows:

$$i_{\text{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2 \frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c}\in\mathcal{C}} P(x^{n_c}) f_{Y|X}(y^{n_c}|x^{n_c})} \quad (15)$$

CID operates on the complete sequences y^{n_c} and $\hat{x}^{n_c}$ and is limited to only consider valid codewords $x^{n_c} \in \mathcal{C}$. This gives a higher complexity, but higher accuracy relative to AID. Algorithm 3 below provides a procedure for computing i_{CID}.

Algorithm 3: Computation of proposed i_{CID} (and ROVA).

Initialization: For $i, j \in \mathcal{S}$, let $\mathcal{T}_m$ be the set of valid trellis branches as defined in *Algorithm 1*. Initialize $m = 0$, $\Gamma_0^0 = 1$, $S_0^0 = 1$.

Iterations:

1) $m = m + 1$
2) Compute branch metrics $\gamma_m(i, j)$ as in Algorithm 1.
3) Compute Γ_m^j as in Algorithm 2.
4) For each $j \in \mathcal{S}$ compute

$$\Sigma_m^j = \sum_{(i,j)\in\mathcal{T}_m} \Sigma_{m-1}^i \gamma_m(i, j)\,. \quad (16)$$

5) if $m = n_c$ conclude by reporting either

$$i_{\text{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2\left(\frac{\Gamma_n^0}{P(x^{n_c})\Sigma_n^0}\right), \text{or} \quad (17)$$

$$P_0^{n_c} = \frac{\Gamma_n^0}{\Sigma_n^0} \quad (18)$$

otherwise, go to Step 1.

Comparing (2) and (15), we find that ROVA and CID have almost the same formula. Starting with (2), including a $P(x^{n_c})$ term in the denominator and taking a logarithm produces (15). Consequently, CID and ROVA have a one-to-one transformation given by

$$i_{\text{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2\left(\frac{P_0^{n_c}}{P(x^{n_c})}\right). \quad (19)$$

CID and ROVA turn out to be identical metrics, revealing a lower-complexity approach to computing ROVA. Considering the common case of a rate-$1/n$ convolutional code where $N_b = 2$, per trellis stage the original ROVA algorithm requires approximately $6N_s$ multiplications, AID requires only N_s multiplications, but is inaccurate. The CID inspired ROVA computation in Algorithm 3 requires only $2N_s$ multiplies and computes the identical ROVA value of $P_0^{n_c}$ as Algorithm 1.

V. Analytical Expression for the $P_0^{n_c}$ Distribution

An analytical expression for the distribution of $P_0^{n_c}$ reveals the relationship between the selected threshold and the induced UER (as shown in Fig. 1) and between the selected threshold and the induced throughput. The analysis below assumes BPSK symbols 1 and -1 are transmitted over an additive white Gaussian noise (AWGN) channel with noise variance σ^2.

Consider the computed conditional pdf $f_{Y|X}(y^{n_c}|\hat{x}^{n_c})$ in (2) as a random variable F and recall that for an AWGN channel it is computed as

$$F = \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-\hat{x}_i)^2}{2\sigma^2}} \quad (20)$$

where n_b is the number of binary symbols in x^{n_c}. For example, with a rate-1/3 convolutional code, $n_b = 3n_c$. If $\hat{x}^{n_c} = x_t^{n_c}$, using a subscript to denote the Hamming distance $d_H(\hat{x}^{n_c}, x_t^{n_c}) = 0$,

$$F_0 = \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \quad (21)$$

where z_i is the AWGN in the i^{th} symbol. If $d_H(\hat{x}^{n_c}, x_t^{n_c}) = 1$, with the one difference bit in the j^{th} symbol, then

$$F_1 = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z_j+2)^2}{2\sigma^2}} \prod_{i=1, i\neq j}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \quad (22)$$

$$= e^{-\frac{4+4z_j}{2\sigma^2}} \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \quad (23)$$

$$= e^{-\frac{4+4z_j}{2\sigma^2}} F_0 \quad (24)$$

where the mean of Gaussian describing the j^{th} symbol in (22) is shifted by the difference between the true and decoded values of x_j. For our BPSK modulation, this difference is always 2. This can be generalized to any Hamming distance. For $d_H(\hat{x}^{n_c}, x_t^{n_c}) = m$,

$$F_m = e^{-\frac{4m+\sum_{\ell=1}^{m} 4z_\ell}{2\sigma^2}} F_0 \quad (25)$$

Because Viterbi decoding only considers valid codewords $x^{n_c} \in \mathcal{T}$, the multiplicity of each possible value of $d_H(\hat{x}^{n_c}, x_t^{n_c})$ is a function of the specific convolutional code used to encode the message, and for a terminated trellis $d_H(\hat{x}^{n_c}, x_t^{n_c})$ has some maximum value D. Let A_m be the number of valid codewords $\hat{x}^{n_c}$ with $d_H(\hat{x}^{n_c}, x_t^{n_c}) = m$, which by linearity is the number of valid codewords with Hamming weight w:

$$A_w = \left|\{\hat{x}^{n_c} \in \mathcal{T} : d_H(\hat{x}^{n_c}, x_0^{n_c}) = w\}\right|, \quad (26)$$

where $x_0^{n_c}$ is the transmitted codeword for the all-zeros input.

Viterbi selects the correct codeword with high probability, and when it doesn't the selected $\hat{x}^{n_c}$ usually has similar value of $f_{Y|X}(y^{n_c}|\hat{x}^{n_c})$. Thus we can approximate $f_{Y|X}(y^{n_c}|\hat{x}^{n_c})$ with F_0 so that

$$P_0^{n_c} \approx \frac{F_0}{F_0 \sum_{w=0}^{W} A_w e^{-\frac{4w+\sum_{\ell=1}^{w} 4z_\ell}{2\sigma^2}}}, \quad (27)$$

which can also be expressed as

$$P_0^{n_c} \approx \left(1 + \sum_{w=1}^{W} A_w e^{-\frac{4w+\sum_{\ell=1}^{w} 4z_\ell}{2\sigma^2}}\right)^{-1}. \quad (28)$$

The expression for $P_0^{n_c}$ given in (28) includes a sum of W log-normal random variables. Because the magnitude of the

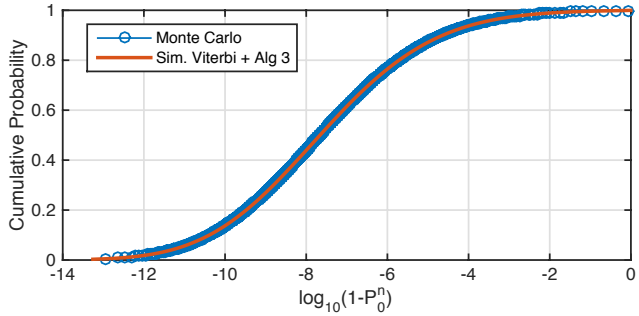

Fig. 5: Cumulative histogram of $\log_{10}(1-P_0^{n_c})$ computed by simulation of Viterbi/Algorithm 3 and by Monte Carlo of (28) with W truncated to 21 for 64-state, rate-1/3 convolutional code with $n = 32$ at SNR 1.0 dB.

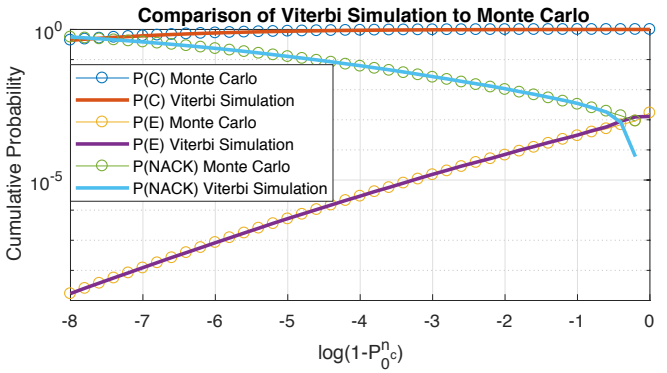

Fig. 6: Comparison of $P(\mathrm{C})$, $P(\mathrm{E})$, and $P(\mathrm{NACK})$ between ROVA probabilities obtained by simulation for the code and channel of Fig. 5 and Monte Carlo using (28) with $W = 21$.

terms decreases rapidly, it can be approximated by summing a few of the most significant terms. Fig. 5 compares the cumulative histogram of $P_0^{n_c}$ found by a simulation of Viterbi decoding with $P_0^{n_c}$ computed as described in Algorithm 3 with the cumulative histogram of $P_0^{n_c}$ given in (28) generated by Monte Carlo using $W = 21$, which involves active terms for $w = 15$ to $w = 21$ and neglects terms with $w > 21$. Seven active terms provides an excellent approximation in Fig. 5.

For the following analysis, $P(\mathrm{C})$ is the probability of Viterbi selecting the correct codeword, $P(\mathrm{E})$ is the probability of Viterbi selecting an incorrect codeword, i.e., UER, and $P(\mathrm{NACK})$ is probability of negative acknowledgement, i.e. rejecting the selected codeword because $P_0^{n_c} < T$.

The expression of (28) indicates a probability distribution f_P on the computed probability of correct decoding $P_0^{n_c}$. The corresponding computed probability of incorrect decoding is $1 - P_0^{n_c}$. Thus, with $P_0^{n_c} > T$ required to accept the Viterbi decoding result, we have the following expressions:

$$P(\mathrm{C}) = \int_{p=T}^{1} p f_P(p) dp \tag{29}$$

$$P(\mathrm{E}) = \int_{p=T}^{1} (1-p) f_P(p) dp \tag{30}$$

$$P(\mathrm{NACK}) = \int_{p=0}^{T} f_P(p) dp\,. \tag{31}$$

Fig. 6 compares the application of (29), (30), and (31) using the cumulative histogram of $P_0^{n_c}$ generated by Monte Carlo using $M = 21$ (shown in Fig. 5) to the values of $P(\mathrm{C})$, $P(\mathrm{E})$, and $P(\mathrm{NACK})$ obtained by simulation of Viterbi decoding with $P_0^{n_c}$ computed as described in Algorithm 3 and then applying the threshold T to decide if the codeword selected by Viterbi should be accepted.

VI. Conclusion

This paper compared the ROVA algorithm of [8] to AID from [13] and found that AID is less accurate because it considers all x^{n_c} sequences as possible rather than restricting attention only to valid codewords. When AID is modified to consider only valid codewords, it becomes equivalent to ROVA, and reveals a lower complexity approach to ROVA as compared to the algorithm presented in [8]. This paper also derived an expression for the random variable that describes the codeword reliability according to ROVA and showed how it can be used to accurately model the probabilities of correct decoding, undetected error, and negative acknowledgement that will result when ROVA is used to determine whether a codeword selected by Viterbi is sufficiently reliable.

References

[1] C.-Y. Lou, B. Daneshrad, and R. D. Wesel, "Convolutional-code-specific CRC code design," *IEEE Transactions on Communications*, vol. 63, no. 10, pp. 3459–3470, Oct 2015.

[2] H. Yang, S. V. S. Ranganathan, and R. D. Wesel, "Serial list Viterbi decoding with CRC: Managing errors, erasures, and complexity," in *IEEE Global Communications Conference*, Abu Dhabi, UAE., Dec 2018.

[3] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *Int. Conf. Dependable Systems and Networks*, Jun. 2004, pp. 145–154.

[4] European Telecommunications Standards Institute 3GPP TS 25.212 version 7.0.0 release 7. [Online]. Available: https://portal.3gpp.org

[5] C. Lott, O. Milenkovic, and E. Soljanin, "Hybrid ARQ: Theory, state of the art and future directions," in *IEEE Inf. Theory Workshop (ITW)*, Bergen, Norway, Jul. 2007.

[6] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of error control coding," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2531–2560, Oct. 1998.

[7] R. D. Wesel, N. Wong, A. Baldauf, A. Belhouchat, A. Heidarzadeh, and J. Chamberland, "Transmission lengths that maximize throughput of variable-length coding & ACK/NACK feedback," in *IEEE Global Communications Conference*, Abu Dhabi, UAE., Dec 2018.

[8] A. Raghavan and C. Baum, "A reliability output Viterbi algorithm with applications to hybrid ARQ," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1214–1216, May 1998.

[9] A. R. Williamson, T.-Y. Chen, and R. D. Wesel, "Variable-length convolutional coding for short blocklengths with decision feedback," *IEEE Trans. on Comm.*, vol. 63, no. 7, pp. 2389–2403, Jul 2015.

[10] J. Fricke and P. Hoeher, "Reliability-based retransmission criteria for hybrid ARQ," *IEEE Transactions on Communications*, vol. 57, no. 8, pp. 2181–2184, Aug 2009.

[11] J. C. Fricke and P. A. Hoeher, "Word error probability estimation by means of a modified Viterbi decoder," in *2007 IEEE 66th Vehicular Technology Conference*, Sep 2007.

[12] A. R. Williamson, M. J. Marshall, and R. D. Wesel, "Reliability-output decoding of tail-biting convolutional codes," *IEEE Transactions on Communications*, vol. 62, no. 6, pp. 1768–1778, Jun 2014.

[13] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Feedback in the non-asymptotic regime," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 4903 – 4925, August 2011.