Exact Probability of Erasure and a Decoding Algorithm for Convolutional Codes on the Binary Erasure Channel

Brian M. Kurkoski, Paul H. Siegel, and Jack K. Wolf[1]
Department of Electrical and Computer Engineering
University of California, San Diego, USA
{kurkoski,psiegel,jwolf}@ucsd.edu

***Abstract*— Analytic expressions for the exact probability of erasure for systematic, rate-1/2 convolutional codes used to communicate over the binary erasure channel and decoded using the soft-input, soft-output (SISO) and *a posteriori* probability (APP) algorithms are given. An alternative forward-backward algorithm which produces the same result as the SISO algorithm is also given. This low-complexity implementation, based upon lookup tables, is of interest for systems which use convolutional codes, such as turbo codes.**

I. Introduction

Error-correcting convolutional codes are widely used in communications, owing to their efficient encoding and optimal yet simple decoding algorithms. They are also the most common component code in turbo codes, which have excellent performance on a variety of channels models, including the erasure channel [1], [2]. Although the erasure channel is a simple communications model, there are applications where it is of interest, such as in computer networking and redundant arrays of disk drives.

In this paper, we will use Markov chain theory to investigate systematic, rate-1/2 convolutional codes, both recursive and non-recursive, when transmitted over the binary erasure channel. The received symbols are decoded by either the soft-input, soft-output (SISO) [3] or the *a posteriori* probability (APP) [4] algorithm. Both are considered instances of forward-backward algorithms; the SISO algorithm is of interest when it is used as a component in turbo decoders, and the APP algorithm is of interest when we are considering the performance of a single convolutional code. We give two results. First, we give analytic expressions for the exact probability of bit erasure for some convolutional codes on the binary erasure channel. Second, we show a simple algorithm which uses table lookups to produce the same results as the SISO algorithm, but with far less complexity.

Markov chain theory has been used to derive analytic expressions for the exact probability of error for a memory $\nu = 2$ recursive convolutional code, transmitted over the binary symmetric channel and decoded using the Viterbi algorithm [5]. This was an extension of the work in [6], which calculated the exact error probability for non-recursive codes. The use of Markov chains to derive the probability of decoding error goes back to the late 1960s [7]. Markov chain theory also has been used to analyze partial response channels [8].

[1]The authors are with the Center for Magnetic Recording Research at UCSD. This research was supported by NSF Grant CCR-0219582, part of the Information Technology Research Program.

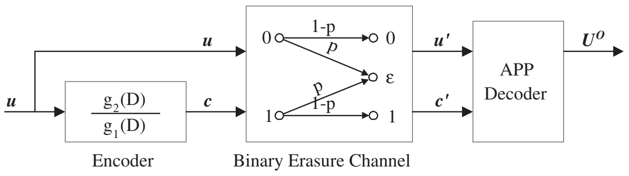

Fig. 1. System block diagram.

The remainder of this paper is organized as follows. In Section II we will give the forward-backward algorithm for both the SISO and APP variations. We will show that the possible state vectors in the forward and backward recursion steps are finite and reasonably small in number. In Section III, we use this observation to show that the forward and backward recursions form two Markov chains. We will compute their steady-state distribution, and give an analytic expression for the exact probability of bit erasure. In Section IV, we use the same observation to introduce an algorithm for decoding of convolutional codes. Section V is the conclusion.

II. The Channel, the Encoder and the Forward-Backward Algorithm

The binary erasure channel (BEC) is a two-input, three-output discrete memoryless channel. Each input bit is either erased with probability p, or received correctly with probability $1-p$, and the occurrences of erasures are independent events.

The block diagram for the system which we are considering is shown in Fig. 1. The input is a sequence of independent and identically distributed bits $u_n \in \{0,1\}$. It is encoded by a systematic, rate-1/2 convolutional encoder with transfer function $[1 \; \frac{g_2(D)}{g_1(D)}]$, where $g_2(D)$ is the "feedforward" polynomial, $g_1(D)$ is the "feedback" polynomial, D is the delay operator, and arithmetic is modulo 2. The memory length of the encoder is ν. The parity output is a sequence of bits c_n. Both the input stream $\mathbf{u} = \ldots u_{n-1}, u_n, u_{n+1}, \ldots$ and the output stream $\mathbf{c} = \ldots c_{n-1}, c_n, c_{n+1}, \ldots$ are bi-infinite. The sequences $\mathbf{u}$ and $\mathbf{c}$ are transmitted over the binary erasure channel and received as $\mathbf{u}'$ and $\mathbf{c}'$, with n^{th} elements u'_n and c'_n, respectively.

The soft-input decoding algorithms we study in this paper require probabilistic inputs. If the transmitted bit u_n is received correctly, then the uncoded soft output channel information $U_n(i) = P[u_n = i|u'_n]$ is 0 or 1. Likewise for $C_n(i) = P[c_n = i|c'_n]$. If the transmitted bit is erased, then $U_n(i) = 1/2$ or $C_n(i) = 1/2$. Note that for the BEC, the soft output information is independent of the channel parameter, whereas for the binary symmetric channel and the additive white Gaussian noise channel this is not true.

Let $M = 2^\nu$ be the number of encoder states. Let $A_n(m)$ and $B_n(m), m = 1, \ldots, M$ be the state probabilities computed forward and backward, respectively. For each trellis edge e, the starting state is $s^S(e)$, the ending state is $s^E(e)$, the associated encoder input label is $u(e)$ and the associated encoder output label is $c(e)$. Let H^A_{n+1}, H^B_n, H_n and H^{APP}_n be normalization constants.

- *The forward-backward step.* Recursively compute $A_{n+1}(m)$ and $B_n(m)$,

$$A_{n+1}(m) = H^A_{n+1} \sum_{e:s^E(e)=m} A_n(s^S(e)) \cdot U_n(u(e))\, C_n(c(e)) \tag{1}$$

$$B_n(m) = H^B_n \sum_{e:s^S(e)=m} B_{n+1}(s^E(e)) \cdot U_n(u(e))\, C_n(c(e)). \tag{2}$$

- *SISO output.* The output of the SISO algorithm is $U^O_n(i) = P(u_n = i|\mathbf{u'}^{n-1}_{-\infty}, \mathbf{u'}^{\infty}_{n+1}, \mathbf{c}')$, and is computed as a function of $A_n(m), B_{n+1}(m)$ and C_n:

$$U^O_n(i) = H_n \sum_{e:u(e)=u_n} A_n(s^S(e)) \cdot C_n(c(e))\, B_{n+1}(s^E(e)), \tag{3}$$

We will define the APP algorithm in terms of the SISO algorithm, and unless noted otherwise, we are referring to the SISO algorithm.

- *APP output.* The output of the APP algorithm is given as a function of the SISO output:

$$U^{O,APP}_n(i) = H^{APP}_n\, U_n(i)\, U^O_n(i).$$

Since we are considering the behavior at some time n sufficiently far from any trellis termination that end effects can be neglected, we do not specify an initial condition for the decoder's forward and backward recursions.

We will write $\mathbf{A}_n = (A_n(1), \ldots, A_n(M))$ and $\mathbf{B}_n = (B_n(1), \ldots, B_n(M))$. The state metric vector $\mathbf{A}_n$ takes on values from the set $\Sigma_A = \{\boldsymbol{\sigma}^1_a, \ldots, \boldsymbol{\sigma}^{|\Sigma_A|}_a\}$ and $\mathbf{B}_n$ takes on values from the set $\Sigma_B = \{\boldsymbol{\sigma}^1_b, \ldots, \boldsymbol{\sigma}^{|\Sigma_B|}_b\}$. We observed that for the codes which we consider in this paper, the state metric vectors $\mathbf{A}_n$ and $\mathbf{B}_n$ are drawn from relatively small sets Σ_A and Σ_B.

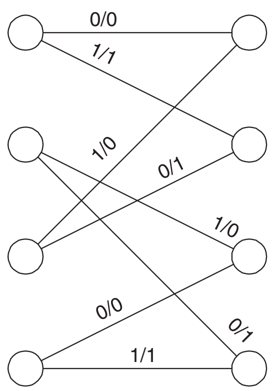

Fig. 2. Trellis for the code $\left[1\ \frac{1}{1+D+D^2}\right]$. The trellis labels are u_n/c_n.

Example Consider the code for which $\frac{g_2}{g_1} = \frac{1}{1+D+D^2}$. By inspecting the trellis (Fig. 2), it can be easily verified:

$$\begin{aligned}\Sigma_A = \{&(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1),\\ &(\tfrac{1}{2}, \tfrac{1}{2} 0, 0), (\tfrac{1}{2}, 0, \tfrac{1}{2} 0), (\tfrac{1}{2}, 0, 0, \tfrac{1}{2}), (0, \tfrac{1}{2}, \tfrac{1}{2} 0),\\ &(0, \tfrac{1}{2}, 0, \tfrac{1}{2}), (0, 0, \tfrac{1}{2}, \tfrac{1}{2}), (\tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4})\}.\end{aligned}$$

If we assume that the all-zeros codeword was transmitted, then $\mathbf{A}_n$ can only take on values from a subset of Σ_A, called Σ^*_A. The subset Σ^*_B is similarly defined.

Example For the same example code, $|\Sigma^*_A| = 5$ and,

$$\begin{aligned}\Sigma^*_A = \{&(1,0,0,0), (\tfrac{1}{2}, \tfrac{1}{2} 0, 0), (\tfrac{1}{2}, 0, \tfrac{1}{2} 0),\\ &(\tfrac{1}{2}, 0, 0, \tfrac{1}{2}), (\tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4})\}.\end{aligned}$$

Also, we observed that the output of the SISO and APP algorithms takes on only the values $\{0, \frac{1}{2}, 1\}$. In other words, just as the erasure channel produces only the correct symbol or an erasure, so too these algorithms only produce the correct symbol or an erasure. For this reason, we speak of the "probability of erasure" rather than the "probability of error."

III. Analytic expression for exact probability of bit erasure

To find an analytic expression for the probability of bit erasure of convolutional codes over the BEC, we will first show that the forward and backward recursions given by (1) and (2) form two Markov chains, and then determine the chains' steady-state distributions, π_A and π_B respectively. Once we have these, we can compute the probability of erasure $p_e = P(U^O_n = 1/2)$ given by (3). The technique that we give here is similar to one used in [2] to analyze turbo codes.

In this section we assume the all-zeros codeword was transmitted. Because of the linearity of convolutional codes, the analytic expressions that we derive will be valid for arbitrary codewords.

TABLE I

EXACT PROBABILITY OF ERASURE FOR SOME CONVOLUTIONAL CODES, UNDER APP DECODING.

Code	p_e^{APP}
$[1\ 1+D]$	$\frac{p^3}{(1-p+p^2)^2}$
$\left[1\ \frac{1}{1+D}\right]$	$\frac{2p^3-2p^4+p^5}{(1-p+p^2)^2}$
$\left[1\ 1+D+D^2\right]$	$\frac{3p^4-6p^5+10p^6-11p^7+6p^8+p^9-3p^{10}+p^{11}}{(1-p-p^2+5p^3-5p^4+3p^5-p^6)^2}$
$\left[1\ \frac{1}{1+D^2}\right]$	$\frac{2p^3-2p^4+p^5}{(1-p+p^2)^2}$
$\left[1\ \frac{1}{1+D+D^2}\right]$	$\frac{5p^4-10p^5+6p^6+10p^7-28p^8+38p^9-33p^{10}+18p^{11}-6p^{12}+p^{13}}{(1-p-p^2+5p^3-5p^4+3p^5-p^6)^2}$
$\left[1\ \frac{1+D^2}{1+D+D^2}\right]$	$\frac{3p^5-18p^6+56p^7-107p^8+132p^9-101p^{10}+42p^{11}-3p^{12}-4p^{13}+p^{14}}{(1-4p+7p^2-5p^3+3p^5-p^6)^2}$
$\left[1\ \frac{1+D+D^2}{1+D^2}\right]$	$\frac{2p^5-10p^6+26p^7-39p^8+31p^9-p^{10}-22p^{11}+21p^{12}-8p^{13}+p^{14}}{(1-4p+7p^2-5p^3+3p^5-p^6)^2}$

A. Steady-state distributions

The state metric vector $\mathbf{A}_n$ can take on one of $|\Sigma_A^*|$ values, and the sequence $\ldots, \mathbf{A}_{n-1}, \mathbf{A}_n, \mathbf{A}_{n+1}, \ldots$ generated by the forward recursion forms a Markov chain. This Markov chain is described by a probability transition matrix M_A, where the i, j entry represents the probability of making a transition from state $\mathbf{A}_n = \boldsymbol{\sigma}_a^i$ to state $\mathbf{A}_{n+1} = \boldsymbol{\sigma}_a^j$. The Markov transition probabilities depend upon the channel parameter p. The steady-state distribution of the Markov chain is the solution to:

$$\pi_A = M_A^t \pi_A, \text{ and} \tag{4}$$

$$\sum_{i=1}^{|\Sigma_A^*|} \pi_{A,i} = 1, \tag{5}$$

where $\pi_{A,i}$ represents the i^{th} element of the row vector π_A. Similarly, the steady-state distribution for the backward recursion π_B can be found, with the difference that rows of M_B correspond to $\mathbf{B}_{n+1}$ and columns correspond to $\mathbf{B}_n$.

Example Consider again the example code for which $\frac{g_2}{g_1} = \frac{1}{1+D+D^2}$. The first row and first column of M_A correspond to $(1,0,0,0)$, the second row and second column correspond to $(\frac{1}{2}, \frac{1}{2}, 0, 0)$, etc. M_A can be found by examining one stage of the trellis, shown in Fig. 2. For example, consider $\mathbf{A}_n = (1,0,0,0)$. If u_n and c_n are both erased (which occurs with probability p^2), then $\mathbf{A}_{n+1} = (\frac{1}{2}, \frac{1}{2}, 0, 0)$. If u_n and c_n are not both erased (which occurs with probability $1-p^2$), then $\mathbf{A}_{n+1} = (1,0,0,0)$. Recalling that the all-zeros codeword was transmitted, M_A is the $|\Sigma_A^*|$-by-$|\Sigma_A^*|$ matrix:

$$M_A = \begin{bmatrix} 1-p^2 & p^2 & 0 & 0 & 0 \\ (1-p)^2 & 0 & p-p^2 & p-p^2 & p^2 \\ 1-p & p & 0 & 0 & 0 \\ 0 & 0 & 1-p^2 & 0 & p^2 \\ 0 & 0 & 1-p & 0 & p \end{bmatrix}.$$

Using a technique similar to [9], we found the steady-state distribution:

$$\pi_A = \frac{1}{k(p)} \begin{bmatrix} 1-p-2p^2+3p^3-p^4 \\ p^2-p^3 \\ 2p^3-3p^4+p^5 \\ p^3-2p^4+p^5 \\ p^4+p^5-p^6 \end{bmatrix},$$

where $k(p) = 1-p-p^2+5p^3-5p^4+3p^5-p^6$. For the backwards Markov chain, computing M_B and π_B is similar.

B. Computing output erasure probability

The SISO output $U_n^O(i)$ is computed according to (3), a function which depends upon $\mathbf{A}_n$, $\mathbf{B}_{n+1}$ and $C_n(i)$. Using the steady-state distribution π_A and π_B, and the coded symbol erasure probability, p, we can compute the output erasure probability:

$$p_e = P(U_n^O(0) = \frac{1}{2}).$$

Under the all-zeros codeword assumption, $U_n^O(0)$ can take on either the value 1, which corresponds to decoding correctly, or the value $\frac{1}{2}$, which corresponds to decoding an erasure. We will write U_n^O to mean $U_n^O(0)$, and C_n to mean $C_n(0)$.

For every pair of $\mathbf{A}_n$, $\mathbf{B}_{n+1}$, one of the following cases will be true:

Case 1 $U_n^O = 1$ (knowledge of c_n is not required to correctly decode u_n),

Case 2 $U_n^O = 1$ if $C_n = 1$; otherwise, $U_n^O = 1/2$ (we know u_n if and only if we know c_n),

Case 3 $U_n^O = 1/2$ (u_n cannot be known, even if c_n is known).

Define a matrix T which has as its i, j entry the conditional probability of an output erasure:

$$\begin{aligned} T_{i,j} &= P(U_n^O = \frac{1}{2} | \mathbf{A}_n = \boldsymbol{\sigma}_a^i, \mathbf{B}_{n+1} = \boldsymbol{\sigma}_b^j) \\ &= \begin{cases} 0, & \text{Case 1} \\ P(C_n = \frac{1}{2}) = p, & \text{Case 2} \\ 1, & \text{Case 3} \end{cases} \end{aligned}$$

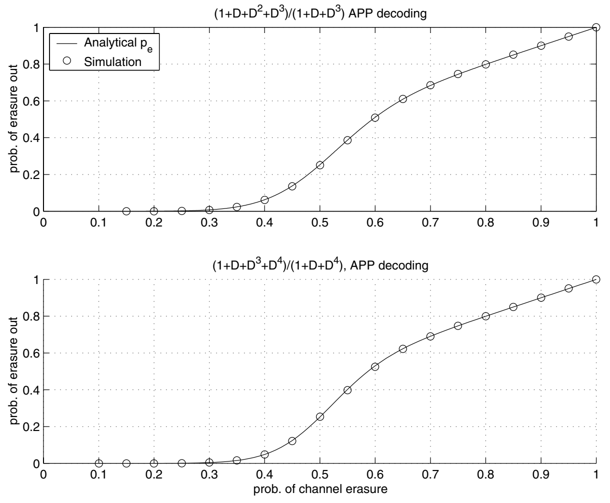

Fig. 3. Comparison of analytical and simulation results for APP decoding of the $\nu = 3$ code $\left[1 \ \frac{1+D+D^2+D^3}{1+D+D^3}\right]$ (top; see also Table II) and the $\nu = 4$ code $\left[1 \ \frac{1+D+D^3+D^4}{1+D+D^4}\right]$ (bottom).

Then, the probability of erasure $p_e = P(U_n^O = \frac{1}{2})$ is:

$$\begin{aligned} p_e &= \sum_{i=1}^{|\Sigma_A^*|} \sum_{j=1}^{|\Sigma_B^*|} P(U_n^O = \frac{1}{2} | \mathbf{A}_n = \boldsymbol{\sigma}_a^i, \mathbf{B}_{n+1} = \boldsymbol{\sigma}_b^j) \\ &\qquad \cdot P(\mathbf{A}_n = \boldsymbol{\sigma}_a^i) P(\mathbf{B}_{n+1} = \boldsymbol{\sigma}_b^j) \\ &= \sum_{i=1}^{|\Sigma_A^*|} \sum_{j=1}^{|\Sigma_B^*|} T_{i,j} \ \pi_{A,i} \ \pi_{B,j} \\ &= \pi_A \cdot T \cdot \pi_B^t, \end{aligned}$$

where we have used the independence of $\mathbf{A}_n$ and $\mathbf{B}_{n+1}$.

Example Continuing the example earlier for $\frac{g_2}{g_1} = \frac{1}{1+D+D^2}$, we find:

$$T = \begin{bmatrix} 0 & p & 0 & 0 & p \\ 0 & p & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & p & 0 & p & p \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

For the APP algorithm, the output will be an erasure only if the estimate from the channel U_n, and the output of the SISO algorithm U_n^O are both erasures. So,

$$p_e^{APP} = p \cdot p_e.$$

We found analytic expressions for the output erasure probability p_e for selected codes with $\nu = 1, 2, 3, 4$. The expression takes the form of a ratio of two polynomials in p. However, the degree of these polynomials grows very quickly for increasing ν. For various $\nu = 1, 2$ codes, the results are listed in Table I. For $\nu = 3$, we choose the code $\left[1 \ \frac{1+D+D^2+D^3}{1+D+D^3}\right]$ and tabulated the coefficients of p_e in Table II. For $\nu = 4$, we found an exact form of p_e for the code $\left[1 \ \frac{1+D+D^3+D^4}{1+D+D^4}\right]$, but the result is too large to include in the paper. However, in Fig. 3, p_e is plotted as a function of p for this code, and the $\nu = 3$ code as well, for APP decoding. Computer simulations confirmed these analytical results.

IV. A Fast SISO Decoding Algorithm

The above analytic technique suggests an algorithm for decoding convolutional codes over the erasure channel. The forward recursion (1) computes $\mathbf{A}_{n+1}$ as a function of $\mathbf{A}_n$, U_n and C_n. Since $\mathbf{A}_n$ can only take on $|\Sigma_A|$ values, and U_n and C_n can each only take on three values (0, $\frac{1}{2}$ or 1), it is possible to represent this function with a lookup table; likewise for the backward recursion (2) and the output function (3). Thus, the SISO algorithm can be implemented with three table lookups per information bit, replacing the extensive multiplications and additions normally required. Note that the number of operations of the proposed algorithm is independent of the number of states in the code being decoded. Furthermore, we want to emphasize that this algorithm is exact—it produces the same result as the SISO algorithm. A table lookup-based approach for decoding of convolutional codes on the binary symmetric channel was proposed in [10]; however this technique works with the syndrome trellis, whereas the algorithm proposed here works on the decoding trellis directly.

In this section, we remove the all-zeros codeword assumption and permit any codeword to be transmitted, as would be required for an implementation.

The size of the lookup table is a function of the memory length ν, and here we give a rough idea of the amount of memory required for storing the lookup tables in binary-addressable memory. Each table has indices corresponding to the function's arguments, and a value corresponding to the function's value. Since neither the number of possible coded

TABLE II

Coefficients for the exact probability of erasure for $\left[1 \ \frac{1+D+D^2+D^3}{1+D+D^3}\right]$, where $p_e = \frac{\sum_i n_i p^i}{\sum_i d_i p^i}$.

i	n_i	d_i	i	n_i	d_i
0		1	19	134239	-74054
1		-10	20	-129855	39712
2		45	21	79410	536
3		-114	22	9568	-35889
4		156	23	-108748	55316
5	4	-14	24	179954	-56418
6	-31	-497	25	-196341	45266
7	110	1534	26	157705	-28631
8	-200	-3179	27	-88596	10944
9	68	5018	28	22358	3299
10	708	-5454	29	17728	-10162
11	-2594	1958	30	-28370	9746
12	5954	7519	31	21466	-5944
13	-10365	-23778	32	-10978	2522
14	12758	46477	33	4018	-748
15	-6168	-72636	34	-1046	149
16	-17003	94825	35	185	-18
17	57335	-104206	36	-20	1
18	-103253	96729	37	1	

TABLE III

LOOKUP TABLE STORAGE REQUIREMENTS FOR SOME REPRESENTATIVE CODES.

Code	$\lvert\Sigma_A\rvert$, $\lvert\Sigma_B\rvert$	$\lvert\Sigma_A^*\rvert$, $\lvert\Sigma_B^*\rvert$	Table sizes, bits forward, backward	output	Total storage
$\begin{bmatrix}1 & \frac{1}{1+D}\end{bmatrix}$	3	2	$2^6 \times 2$	$2^6 \times 2$	0.046875 kbytes
$\begin{bmatrix}1 & \frac{1}{1+D^2}\end{bmatrix}$	9	4	$2^8 \times 4$	$2^{10} \times 2$	0.5 kbytes
$\begin{bmatrix}1 & \frac{1+D^2}{1+D+D^2}\end{bmatrix}$	11	5	$2^8 \times 4$	$2^{10} \times 2$	0.5 kbytes
$\begin{bmatrix}1 & \frac{1}{1+D+D^2+D^3}\end{bmatrix}$	45	14	$2^{10} \times 6$	$2^{14} \times 2$	5.5 kbytes
$\begin{bmatrix}1 & \frac{1+D+D^3}{1+D+D^2+D^3}\end{bmatrix}$	51	16	$2^{10} \times 6$	$2^{14} \times 2$	5.5 kbytes
$\begin{bmatrix}1 & \frac{1+D+D^3+D^4}{1+D+D^4}\end{bmatrix}$	307	67	$2^{13} \times 9$	$2^{20} \times 2$	274 kbytes

and uncoded symbol probabilities nor (in general) $|\Sigma_A|$ or $|\Sigma_B|$ is a power of two, we will round up to the smallest integer number of bits required. For example, U_n could be one of three values, and so indexing (or addressing) requires $\lceil \log_2 3 \rceil$ bits.

The forward recursion table's indices are composed of representations of $\mathbf{A}_n$, U_n and C_n and the table's value is a representation of $\mathbf{A}_{n+1}$. The index of an element of $|\Sigma_A|$ requires $\lceil \log_2 |\Sigma_A| \rceil$ bits, and each of U_n and C_n requires $\lceil \log_2 3 \rceil$ bits, so $\lceil \log_2 |\Sigma_A| \rceil + 2\lceil \log_2 3 \rceil$ bits are required for indexing the table. And, the table's value must describe an element of Σ_A, which requires $\lceil \log_2 |\Sigma_A| \rceil$ bits of storage per index. So the total storage required for the forward recursion lookup table is:

$$2^{\lceil \log_2 |\Sigma_A| \rceil + 2\lceil \log_2 3 \rceil} \cdot \lceil \log_2 |\Sigma_A| \rceil \text{ bits.}$$

Likewise, the backward recursion table's indices are composed of $\mathbf{B}_{n+1}$, U_n and C_n, and the table's value is $\mathbf{B}_n$. The storage requirement for the backward recursion lookup table is:

$$2^{\lceil \log_2 |\Sigma_B| \rceil + 2\lceil \log_2 3 \rceil} \cdot \lceil \log_2 |\Sigma_B| \rceil \text{ bits.}$$

To produce the uncoded output, the lookup table's index is composed of $\mathbf{A}_n$, $\mathbf{B}_{n+1}$ and C_n, and the table's value is U_n. So, the storage requirement is:

$$2^{\lceil \log_2 |\Sigma_A| \rceil + \lceil \log_2 |\Sigma_B| \rceil + \lceil \log_2 3 \rceil} \cdot \lceil \log_2 3 \rceil \text{ bits.}$$

In Table III we list the table sizes that would be required for such a lookup table-based decoder.

V. CONCLUSION

In this paper, we investigated convolutional codes for the binary erasure channel when they are decoded by the SISO and APP algorithms. We used Markov chain theory to describe the steady-state distribution of the forward recursion and backward recursion steps, and from this obtained an analytic expression for the exact probability of erasure, for codes with $\nu = 1, 2, 3, 4$. Because these expressions are exact, they are valid for all values of the channel erasure probability $0 \leq p \leq 1$.

This analysis suggested a fast forward-backward algorithm which replaces the numerous computations of the conventional algorithm at each recursion step with a single table lookup. Thus, the SISO output can be computed with three table lookups per bit, and the tables can be stored in a reasonable amount of binary-addressable memory.

REFERENCES

[1] K. E. Tepe and J. B. Anderson, "Turbo codes for binary symmetric and binary erasure channels," in *Proceedings of IEEE International Symposium on Information Theory*, (Cambridge, MA), p. 59, IEEE, August 1998.

[2] C. Méasson and R. Urbanke, "Further analytic properties of EXIT-like curves and applications," in *Proceedings of IEEE International Symposium on Information Theory*, (Yokohama, Japan), p. 266, IEEE, July 2003.

[3] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output APP module for iterative decoding of concatenated codes," *IEEE Communications Letters*, vol. 1, pp. 22–24, January 1997.

[4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.

[5] M. Lentmaier, D. Truhachev, and K. S. Zigangirov, "Analytic expression for the exact bit error probability of the (7,5) convolutional code," in *Proceedings of IEEE International Symposium on Information Theory*, (Lausanne, Switzerland), p. 7, IEEE, June 2002.

[6] M. R. Best, M. V. Burnashev, Y. Levy, A. Rabinovich, P. C. Fishburn, A. R. Calderbank, and J. Daniel J. Costello, "On a technique to calculate the exact performance of a convolutional code," *IEEE Transactions on Information Theory*, vol. 41, pp. 441–447, March 1995.

[7] T. N. Morrissey, "A Markovian analysis of Viterbi decoders for convolutional codes," in *Proceedings of National Electronics Conference*, pp. 303–307, Oct 1969.

[8] H. D. Pfister, *On the Capacity of Finite State Channels and the Analysis of Convolutional Accumulate-m Codes*. PhD thesis, University of California, San Diego, La Jolla, CA, USA, March 2003.

[9] G. W. Stewart, "Gaussian elimination, perturbation theory, and Markov chains," in *Linear Algebra, Markov Chains, and Queuing Models*, (New York, NY, USA), pp. 59–69, Springer-Verlag, 1992.

[10] J. P. M. Schalkwijk and A. J. Vinck, "Syndrome decoding of convolutional codes," *IEEE Transactions on Communications*, vol. 23, pp. 789–792, July 1975.