Modified Maximum A Posteriori Algorithm
For Iterative Decoding of Turbo codes

Prof M. Srinivasa Rao\textsuperscript{#1}, Dr P. Rajesh Kumar\textsuperscript{*2}, K. Anitha\textsuperscript{#3}, Addanki Srinu\textsuperscript{*4}

\textsuperscript{#1}Professor, Department of ECE, Sir C R Reddy Engineering College, Eluru, AP, INDIA
\textsuperscript{#2}Professor, Department of ECE, PVP Siddhartha institute of technology, Vijayawada, AP, INDIA
\textsuperscript{#3}Associate Professor, Department of ECE, Sir C R Reddy Engineering College, Eluru, AP, INDIA
\textsuperscript{*4}Post graduate student, Department of ECE, Sir C R Reddy Engineering College, Eluru, AP, INDIA
\textsuperscript{1}msrao55@yahoo.com \textsuperscript{2}rkpanakala@yahoo.com \textsuperscript{3}komma_anitha22@yahoo.co.in \textsuperscript{4}addanki6@gmail.com

Abstract:
Turbo codes are one of the most powerful error correcting codes. What makes these codes so powerful is the use of the so-called iterative decoding or turbo decoding. An iterative decoding process is an iterative learning process for a complex system where the objective is to provide a good suboptimal estimate of a desired signal. Iterative decoding is used when the true optimal estimation is impossible due to prohibitive computational complexities. This paper extends the mathematical derivation of the original MAP algorithm and shows log likelihood values can be computed differently. The proposed algorithm results in savings in the required memory size and leads to a power efficient implementation of MAP algorithm in channel coding.

Key words: Turbo codes, Iterative decoding, Map algorithm, Memory savings

I. INTRODUCTION

Recently, the digital communications has been further strengthened by important developments in at least two specific areas.

One of the area is digital signal decoding and detection. When applied to turbo codes and low-density parity – check (LDPC) codes, iterative algorithms are capable of approaching Shannon’s channel capacity bound within a very small margin [1], [2].

This paper is concerned with dynamic analysis of iterative decoding for turbo codes. Our aim is to show how to analyze an iterative decoding process using a system theory based approach more specifically.

II. TURBO ENCODING

Ever since Shannon published his famous channel coding theorem [14] in 1948, the advances in the field of communications theory can in one way be viewed as a painstaking pursue for discovering practical coding and decoding algorithms which enable us to approach the Shannon capacity limit.

A turbo encoder is illustrated in Figure 1. It is a binary code, consisting of two Recursive systematic encoders $G_1(z)$ and $G_2(z)$, an interleaver. The two constituent encoders are typically the same and will be denoted by $G(z)$. The two constituent encoders share the same systematic sequence $x_s$. 
III. MAP DECODING

The turbo decoding algorithm is depicted in Figure 2. The input data to the turbo decoder are $y_s$, $y_{p1}$ and $y_{p2}$, which are the noisy version of $x_s$, $x_{p1}$ and $x_{p2}$ coming from demodulation.

Using this method the likelihoods of different bits are computed and passed to the second decoder. The second decoder computes the likelihood ratios and passed the first decoder. The process is repeated until the likelihoods suggest high probability of correct decoding for each bit. $L_{e12}$ and $L_{e21}$ represent the information passed between the two decoders.

IV. MODIFIED MAP ALGORITHM

The Map decoding algorithm is a recursive technique that computes the Log-Likelihood Ratio (LLR) of each bit based on the entire observed data block of length $N$.

$$y_k = (Ys.k, Y_{1p}.k); y_i^j = (y_1, y_2, ..., y_j); y = y_1^n$$

Where $Ys.k$ is the $k$-th element of $Ys$, and $Y_{1p}.k$ is similarly defined. A posteriori probability (APP) ratio of $u$, as defined below;

$$R(u_k) = \frac{P(u_k = +1 \mid y)}{P(u_k = -1 \mid y)}$$

(1) Where $u_k$ is the $k$-th symbol of $u$.

The log-likelihood ratio (LLR) is given as

$$L(u_k) = \log \left( \frac{P(u_k = +1 \mid y)}{P(u_k = -1 \mid y)} \right)$$

(2) $P(u_k = +1 \mid y) + P(u_k = -1 \mid y) = 1$
Denoting the set of all possible states by $S$ and the state at the $k$-th symbol by $S_k$. Now using the Bayes rule, we have

$$L(uk) = \log \left( \frac{\sum s + P(s_{k-1} = s^1, S_k = s, y) / P(y)}{\sum s R(s_{k-1} = s^1, S_k = s, y) / P(y)} \right)$$

(3)

It is clear from above that $P(y)$ can be cancelled and we only need to find a way for computing $P(S_{k-1} = s^1, S_k = s, y)$, or $P(s_1, s, y)$ for short. By breaking $Y$ into $(y_{k-1}, y_k, y_{k+1})$ and applying the Bayes rule again. We can write

$$p(s^1, s, y) = \alpha_{k-1}(s^1) \cdot y_k \cdot \beta_k(s)$$

(4)

$$\alpha_{k-1}(s^1) = p(s_{k-1} = s^1, y_{k-1}^k)$$

(5)

$$\gamma_k(s^1, s) = p(s_k = s, y_k | s_{k-1} = s^1)$$

(6)

$$\beta_k(s) = p(y^n_{k+1} | s_k = s)$$

(7)

These terms can be computed recursively using the Bayes rule again. More precisely,

$$\alpha_k(s) = \sum_{s_i \in S} \alpha_{i-1} \cdot \gamma_i \cdot \beta_i(s)$$

(8)

With the initial conditions

$$\alpha_0(s = 0) = 1; \quad \alpha_0(s \neq 0) = 0;$$

Where $s=0$ is the known initial state for the code. Similarly,

$$\beta_k(l(s)) = \sum_{s_i \in S} \beta_{i-1} \cdot \gamma_i \cdot \beta_i(s)$$

(9)

With the terminating conditions

$$\beta_n(s = 0) = 1; \quad \beta_n(s \neq 0) = 0$$

If $s=0$ is the known terminating state for the code. If the code is not terminated, $\beta_n(s)$ is usually set equally.

It remains to compute $\gamma_k(s^1, s)$, for which we have

$$\gamma(s^1, s) = P(s | s^1)P(y^k_k | s^1, s)$$

$$= Pa(u_k)P(y_k | u_k, x_{pl,k})$$

(10)

Where the values of $u_k$ and $x_{pl,k}$ correspond to the transition from $s^1$ to $s$. The term $Pa(u_k)$ is the a priori probability of $u_k$ which is related to the extrinsic information $L_{21}$ as follows:

$$\overline{L}_{21,k} = \log \left( \frac{P_u(uk = +1)}{P_u(uk = -1)} \right)$$

(11)

i.e., the $k$-th element of $L_{21}$ is the log a priori probability ratio for $u_k$.

For example, if the channel is an additive Gaussian white noise (AGWN) channel with noise variance $\sigma^2$, then $y_{sk}$ and $y_{pl,k}$ are independent and we have

$$P(y_k / u_k, x_{pl,k}) = P(y_{sk,k} / u_k)P(y_{pl,k} / x_{pl,k})$$

$$= C \exp \left[ \frac{(y_{sk,k} - u_k)^2}{2\sigma^2} \right] \exp \left[ \frac{(y_{pl,k} - x_{pl,k})^2}{2\sigma^2} \right]$$

(12)

With a constant $C$ which does not affect $L(uk)$. If we want to produce a hard estimate of $u_k$, we simply take

$$U_k = \text{sign}(L(u_k))$$

(13)
If we want to compute the extrinsic information $L_{12}^e$ for further iterations, we simply subtract the input extrinsic information from $L(u_k)$, i.e.,

$$L_{12,k}^e = L(u_k) - L_{21,k}^e$$ (14)

V. SIMULATION RESULTS

For different cases the Simulation results are:

The figures (3, 4, 5 and 6) show BER curves for the amount of data of 500, signal to noise ratio as up to 4 db, number of iterations are 5 and the interleaver lengths are taken as 50, 100, 150 and 200 respectively. Here interleaver lengths are only changed.

![Figure 3](image3.png)

![Figure 4](image4.png)
The below figures (7, 8, 9 and 10) shows BER curves for taking amount of data as 500 bits, signal to noise ratio as up to 4 db, the interleaver length is 50 and the number of iterations are taken as 3, 4, 5 and 6 respectively. Here the number of iterations only changed.
Effect of number of iterations on the performance

Figure 8

Effect of number of iterations on the performance

Figure 9

Effect of number of iterations on the performance

Figure 10
VI. CONCLUSION

From the simulation results we can notice two things.

1. The bit error probability decreases as the iterations goes up. This means as the iterations increase the reliability of the decisions taken increases.

2. The interleaver length also affects the performance. As the interleaver length increases the bit error probability decreases.

As BCJR algorithm is very complex, we are trying to modify the algorithm to save memory and to reduce complexity. The basic idea is as follows.

This paper extends the mathematical derivation of the original MAP algorithm and shows that the log likelihood values can be computed using only partial state metric values. By processing N stages in a trellis concurrently, the proposed algorithm results in savings in the required memory size and leads to a power efficient implementation of the MAP algorithm in channel decoding.

REFERENCES