Equalization -based on sequence estimation

Soft-output Equalizers

There are two possible techniques for combating the main impairments of high-speed digital transmission, maximum likelihood sequence estimation (MLSE) and maximum *a posteriori* probability (MAP). The MLSE can be implemented using Viterbi algorithm (VA) which provides optimum performance in the sense of minimizing the sequence error probability. The MAP is based on the computation of *a posteriori* probability of a symbol error that they can provide soft-output in nature.

Here four types of equalizers are investigated, one generates hard-output (VE) and the others generate soft-output.

1. Viterbi Equalizer (VE) uses the conventional VA to generate hard-output.
2. Soft-Output Viterbi Equalizer (SOVE) used the modified version of VA (SOVA) to generate hard-output and reliability information for each symbol.
3. Sup-optimum Soft-output Equalizer (SSE).
4. Max-Log-MAP Equalizer.

Assuming that the sampled received signal r_k for the kth symbol can be represented by

$$r_k = \sum_{i=0}^{L} s_{k-i} g_i(k) + n_k \ . \tag{1}$$

Where s_k represents the TCM/MTCM-coded signal constellation for the kth symbols, $g_i(k)$ denote the combined sampled impulse response at the kth symbol, which can be characterized as a zero-mean complex guassian variable [**Proakis,** 1995], L is the number of symbols influenced by ISI. Finally, n_k denoted the additive noise sequence. Its probability density function can be represented by

$$r(n_k) = \frac{1}{pN_0} \exp\left(- \frac{\left| n_k^2 \right|}{N_0} \right). \tag{2}$$

Thus, the equivalent discrete-time channel model can be represented in Fig 1.

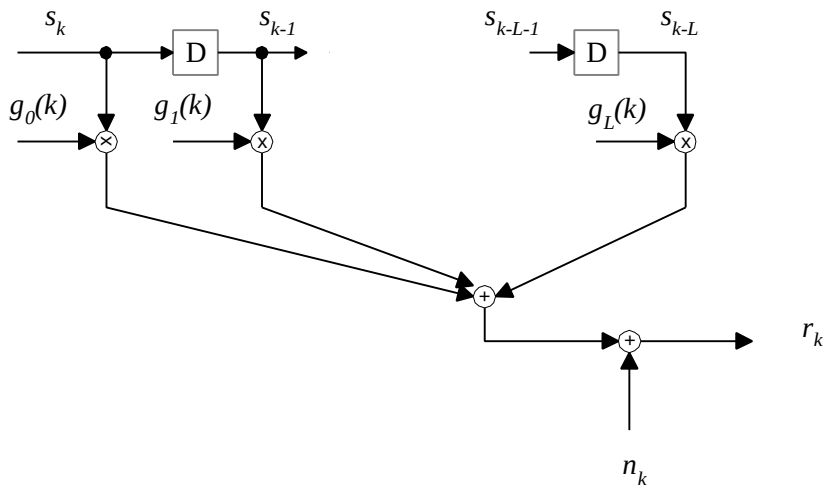


Figure 1 Equivalent discrete-time channel model.

Since all algorithms employed in equalizers are based on the trellis diagram of the frequency-selective fading channel, the following definitions are used to describe the channel trellis diagram [Forney, 1973].

1. *State*: a state s_k at time k is defined as,

$$s_k = (s_{k-L}, s_{k-L+1}, \ldots, s_{k-2}, s_{k-1}), \quad \text{where } s_i = 0 \text{ while } i<0 \tag{3}$$

2. *Transition*: a transition is determined by a symbol transmitted to the channel and defined as

$$x_k = (s_k, s_{k+1}) = (s_{k-L}, s_{k-L+1}, \ldots, s_{k-2}, s_{k-1}, s_k) = (s_k, s_k) \tag{4}$$

3. *Path*: a path p_k is defined as a sequence of states or equivalently a sequence of transmitted symbols up to time k.

$$p_k = (x_0, x_1, \ldots, x_k) = (s_0, s_1, \ldots, s_{k+1}) = (s_0, s_1, \ldots, s_k) \tag{5}$$

4. *Additive Branch Metric (ABM)*: an ABM is the squared Euclidean weight caused by the transition x_k,

$$l_{k+1}(x_k) = l_{k+1}(s_k, s_{k+1}) = \left| r_k - \sum_{i=0}^{L} s_{k-i} g_i(k) \right|^2 \quad (6)$$

5. *Additive Path Metric (APM)*: an APM is defined as the sum of the ABMs of all transitions that constitute the path p_k,

$$\Gamma_{k+1}(p_k) = \Gamma_{k+1}(s_0, s_1, \ldots, s_{k+1}) = \sum_{j=0}^{k} \left| r_j - \sum_{i=0}^{L} s_{j-i} g_i(j) \right|^2 \quad (7)$$

6. *Survivor metric* of state s_{k+1}: a survivor path at time instant k is defined as the path having minimum APM among all M paths stemming from state s_k and entering state s_{k+1}. The survivor metric is the APM of that survivor path.

$$c_{k+1}(s_{k+1}) = \min_{s_k}[\Gamma_{k+1}(p_k)] = \min_{s_k}[\Gamma_{k+1}(s_0, \ldots, s_{k+1})] \quad (8)$$

A. Viterbi Equalizer (VE)

The Viterbi Algorithm (VA) is a recursive optimal solution to the problem of estimating the state sequence of a *discrete-time finite-state Markov process* observed in memoryless noise [**Viterbi**, 1967]. Let the transmitted data sequence $\mathbf{s}=\{s_0,s_1,\ldots,s_{K-1}\}$ consist of K symbols where s_i at each time instant takes on one of M possible values U_i, $i=1,2,\ldots,M$. Equation (1) can be re-written as

$$r_k = s_k g_0(k) + \sum_{i=1}^{L} s_{k-i} g_i(k) + n_k \quad (9)$$

In (9), the second term in the right-hand side represents the intersymbol interference (ISI) introduced by the frequency-selective fading channel and the last term is AWGN. Given the received data sequence $\mathbf{r}=\{r_0,r_1,\ldots,r_{K-1}\}$, the Viterbi equalizer (VE) chooses the data sequence that minimize the Euclidean distance metric.

$$D(\boldsymbol{r}, \boldsymbol{s}^{(m)}) = \sum_{k=0}^{K-1} \left| r_k - \sum_{i=0}^{L} s_{k-i}^{(m)} g_i(k) \right|^2 \tag{10}$$

Where $\boldsymbol{s}^{(m)} = \{s_0^{(m)}, s_1^{(m)}, \ldots, s_{K-1}^{(m)}\}$, $m=1,\ldots,M^K$ denotes the possible transmitted symbol sequences.

The Euclidean distance metric in (10) can be re-written as,

$$D(r, s^{(m)}) = \Gamma_K(s_0, s_1, \ldots, s_K) \tag{11}$$

The trellis structure can be defined as, for each time instant $k=0,1,\ldots,K$, there is a set of M^L nodes where each node corresponds to a values of s_k . Each node has M branches stemming from it, each branch represents an allowable transition from the state s_k to the next state s_{k+1} and corresponds to the transmission of symbol s_k.

The implementation steps of VE can be described as follows:

Step 1: Initialization at time $k=0$. Assume that the trellis is started with known state s_0 .

- Set $c_0(s_0) = 0$; $c_k(s_k) = arbitrary$, $s_k \neq 0, k = 1, \ldots, K$;

$$\Gamma_0(s_0) = 0; \Gamma_k(s_k) = \infty, \qquad s_k \neq 0, k = 1, \ldots, K;$$

Step 2: At any time instant k, r_k is received and there are transitions from s_k to s_{k+1} .

- Use (6), (7) and (8) to compute

$$\Gamma_{k+1}(s_0, \ldots, s_{k+1}) = c_k(s_k) + l_{k+1}(s_k, s_{k+1}) \tag{12}$$

$$c_{k+1}(s_{k+1}) = \min_{s_k} [\Gamma_{k+1}(s_0, \ldots, s_{k+1})] \tag{13}$$

- Update the corresponding state sequence or equivalently symbol sequence of each surviving path of s_{k+1} .

$$p_k = (p_{k-1}, s_k) \tag{14}$$

- Set k to $k+1$ and repeat until $k=K$.

Step 3: The algorithm terminates at time $k=K$. Compute the minimum APM of all paths through the trellis, $\min_{s_K} c_K(s_k)$. The minimum-length path corresponds to the most likely sequence of state and hence the sequence of symbols.

Figure 2 shows trellis diagram for the case $L=1$ and $M=4$ (QPSK). The possible symbols transmitted in each branch are taken from signal set $\{U_1, U_2, U_3, U_4\}$.

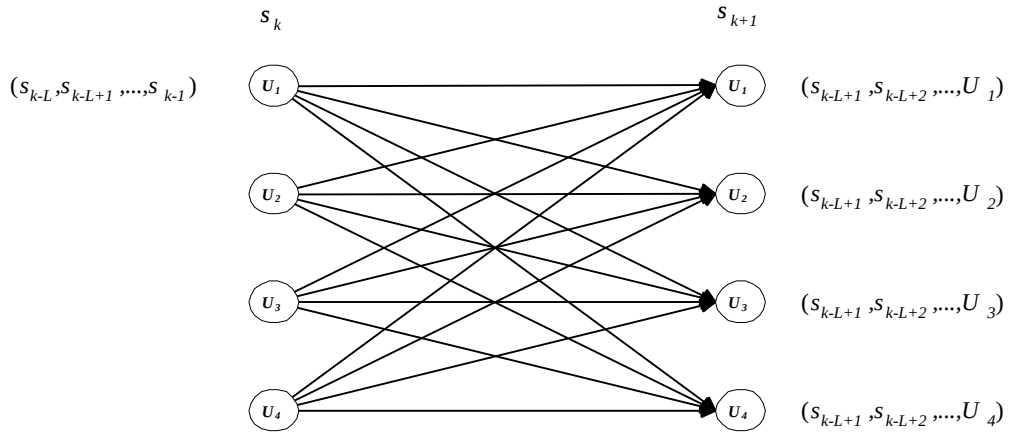


Figure 2 Transitions in the trellis diagram

We illustrate the algorithm for a simple four-state trellis covering 5 time units in Fig 3 and Fig 4. The Fig 3 shows the complete trellis, with each branch labelled with a length (In a real application, the lengths would be functions of received data e.g. squared Euclidean distance).

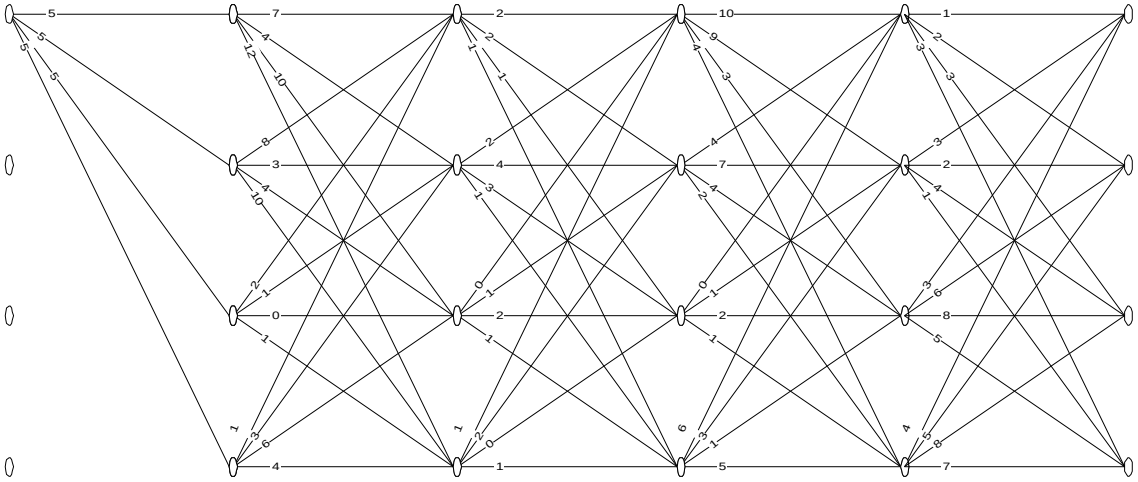
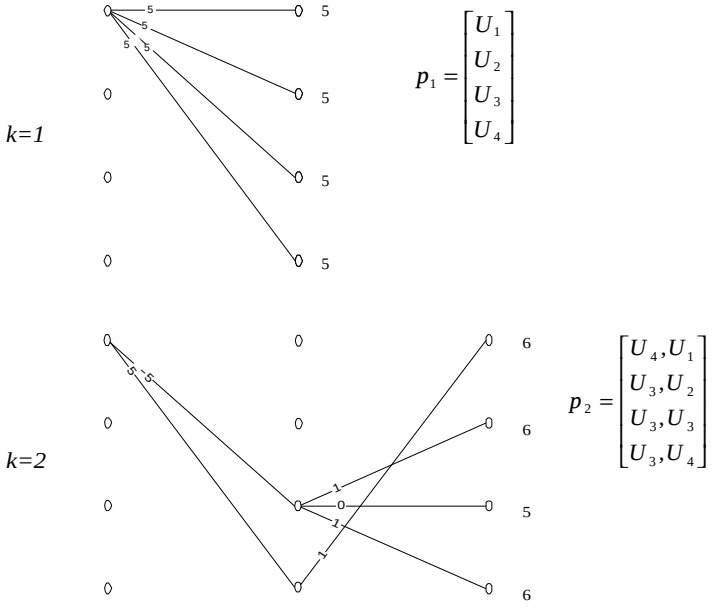


Figure 3 Trellis labeled with branch lengths; *L=1,M=4,K=5*.

Fig. 4 shows the five-recursive steps by which the algorithm determines the shortest path from the initial to the final node. At each stage only the 4 (or fewer) survivors are shown, along with their lengths. The shortest complete path is the heavy line at final stage and the output sequence from VE of this example is $\{U_3, U_4, U_3, U_1, U_1\}$.

Example 1



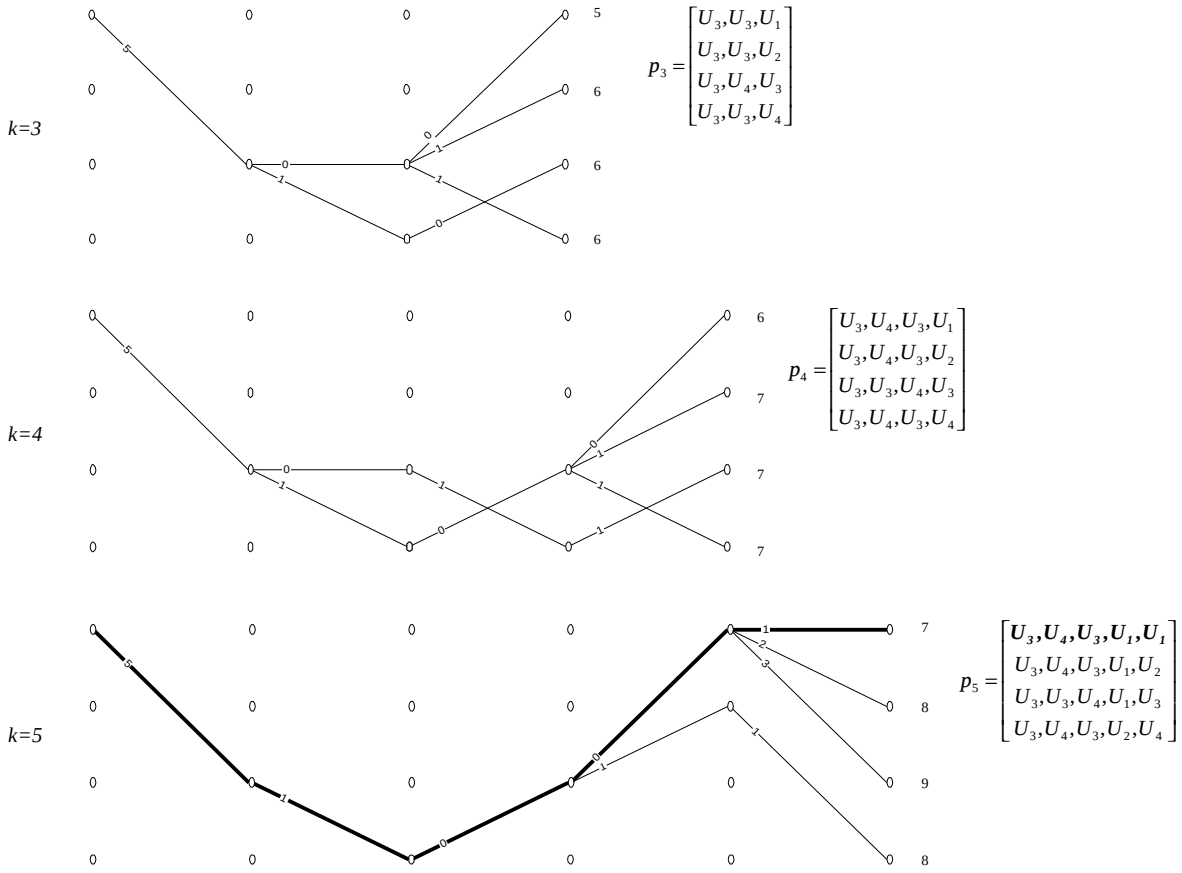


Fig. 4 Recursive determination of the shortest path via the VA.

The complexity of VA is exponential with respect to the length of channel impulse response *(L)* but grows only linearly in the length of data sequence *(K)*.

B. Soft-output Viterbi Equalizer (SOVE)

The SOVA is a modification of the conventional VA to deliver the maximum likelihood path together with reliability information (soft output) individually for each symbol. The soft output decision is a measure of probability of error in decoding a particular symbol. It can also be interpreted as a measure of reliability of the sequence estimator's hard decision [**Hagenauer and Hoeher,** 1989].

In [**Hoeher,** 1990] paper, he proposed the *M*-ary soft-output Viterbi algorithm. At any symbol instant *k*, the probability that a path entering state s_{k+1} has passed the state s_k is given by,

$$P^X(s_{k+1} \mid s_k) = \exp[-\Gamma^X_{k+1}(s_0, \ldots, s_{k+1})], \quad X=1,\ldots,M \qquad (15)$$

Where $\Gamma^X_{k+1}(s_0, \ldots, s_{k+1})$ is the APM of path X given by (7).

There are M possible paths connecting states s_k to a certain state s_{k+1} and hence M possible values of $P^X(s_{k+1}|s_k)$, $X=1,\ldots,M$. The probability P_{s_j} of the jth symbol $s_j \in \{U_1, U_2, \ldots, U_M\}$ of path $X \in \{1, \ldots, M\}$ terminating at state s_{k+1} is

$$P_{s_j} = \frac{\sum_{X \in \Omega_{X_j}} P^X(s_{k+1} \mid s_k)}{\sum_{X \in \{1,\ldots,M\}} P^X(s_{k+1} \mid s_k)} \quad . \qquad (16)$$

Where Ω_{X_j} is the set of all paths, which has the jth symbol s_j identical to the jth symbol of a specified path X.

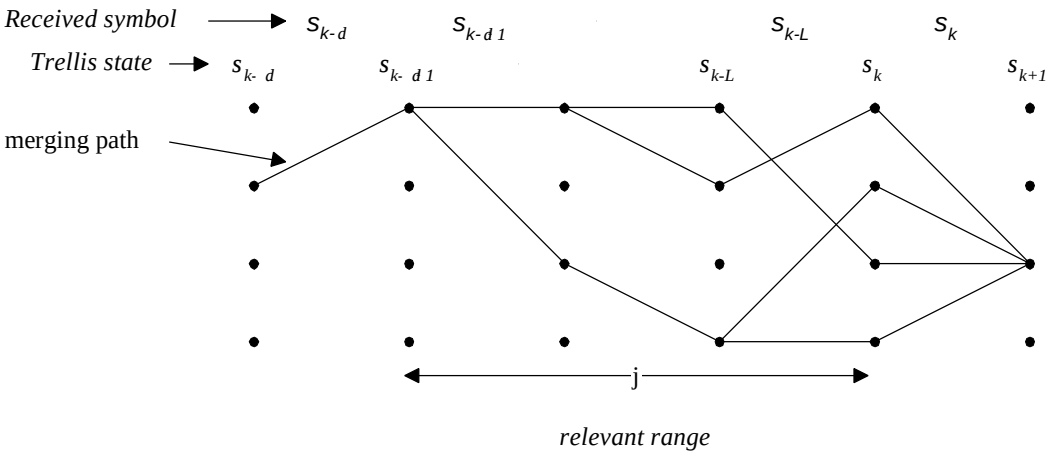


Figure 5 Four-state trellis diagram with memory $L=1$ and $M=4$ transitions per state.

Consider the example of the trellis diagram given in Fig 5. Assume that all the paths have merged at symbol instant $k-d$ (or at state s_{k-d+1}). The relevant range is defined as the range where the symbols may differ

and the relevant symbols are named for the different symbols at the instant. Let $P_{s_j hist}$ be the stored probability for the symbol s_j, (initialized to one). The recursive update of this stored probability is performed by

$$P_{s_j hist} = P_{s_j hist} \cdot P_{s_j} \tag{17}$$

This updating procedure has to be performed whenever there are at least two different symbols at the jth time instant for different paths $X \in \{1,\ldots,M\}$ entering the same state s_{k+1}.

A simplified version of (17) is to compute

$$LogP_{s_j hist} = \min\left\lfloor LogP_{s_j hist}, LogP_{s_j} \right\rfloor \tag{17*}$$

In summary, the SOVA can be described as follows.

For all states (M^L states at s_{k+1})
    determine the survivor path
    for all paths $X \in \{1,\ldots,M\}$
        for the "relevant range"
            for the "relevant symbol"
                update the stored reliability

At time k we output the final hard decision s_{k-d} with the M-ary reliability vector, $P_{s_j hist}$. Where the decisions delay $d$is chosen such that all paths have merged.

The computation complexity of the SOVE makes it difficult to implement in practice for high-rate digital transmission. The complexity reduction with a suboptimum version is possible. The operations of reliability updating process can be reduced significantly when this process is done for only a limited number of paths and/or a limited number of time slots [**Hoeher,** 1990].

- H-SOVE, one-dimensional horizontal update, is to update reliability information value only for the globally best survivor path at a given time k for the relevant length.

- V-SOVE, one-dimensional vertical update, is to update memorized reliability information value for all symbols, but only one '*slot*' in depth.

The implementation steps of the V-SOVE are given below.

Storage

k, time index

$\boldsymbol{u}(s_k)=[s_{k-d}(s_k), s_{k-d+1}(s_k),\ldots,s_k(s_k)]$, $1 \le s_k \le M^L$, Storage for hard tentative decisions

$\mathbf{P}=[\mathbf{p}_{s_{k-d}},\mathbf{p}_{s_{k-d+1}},\ldots,\mathbf{p}_{s_k}]$, Storage for reliabilities,

where $\mathbf{p}_{s_j}=[p_1, p_2,\ldots, p_M]^{\mathrm{T}}$, $k-d \le j \le k$ and p_i, $1 \le i \le M$, is the probability of the symbol being equal to i.

$c(s_k)$, $1 \le s_k \le M^L$, Survivor metric vector

Initialization

$k=0$,

$c(s_0)=0$, i.e. the initial state s_0 of the channel is 1.

$c(s_0)=\infty$, $2 \le s_0 \le M^L$

$\mathbf{P}=[1,1,\ldots,1]$

Iteration

Step 1:

- If $k > d$

  r = position{min[$c(.)$]}, where $c(.)$ is the stored survivor metric vector.

  Output the hard decision on the symbol s_{k-d};

  $\boldsymbol{sym}(k-d)=\boldsymbol{u}_{k-d}(r)$.

  Output M-ary reliability vector ; $\boldsymbol{Prob}(k-d)=\mathbf{p}_{s_{k-d}}$.

- Use (6) and (7) to compute

$$\Gamma_{k+1}(s_0,\ldots,s_{k+1}) = c_k(s_k) + l_{k+1}(s_k, s_{k+1}) \qquad (18)$$

For each $l_{k+1}(s_k, s_{k+1})$ such that the transition from state s_k to state s_{k+1} is allowable. Where $c_k(s_k) = c(s_k)$ and the value of s_k shows the position of the element in the survivor metric vector $c(.)$.

Step 2: For each state s_{k+1}

- Compute

$$c_{k+1}(s_{k+1}) = \min_{s_k}[\Gamma_{k+1}(s_k, s_{k+1})] \tag{19}$$

Where the minimum is taken over the values of s_k having transition with s_{k+1}. In other word, the left-hand side in (19) is the minimum APM among a set W of all M values of s_k having transition with s_{k+1}.

- Store the resulting value from (19) in the survivor metric vector $c(.)$

$$c(s_{k+1}) = c_{k+1}(s_{k+1}) \tag{20}$$

Where the value of s_{k+1} shows the position for which $c_{k+1}(s_{k+1})$ is stored in the survivor metric vector $c(.)$.

- Store the corresponding survivor sequence of symbols, which constitutes this minimum APM merging in s_{k+1} as follows

$$\boldsymbol{u}(s_{k+1}) = [\boldsymbol{u}(s_k), s_{k+1}] \tag{21}$$

Where s_k is defined such that the path having a minimum APM among all M paths merging in s_{k+1} gone through s_k. In other words, among M values of s_k having transition with s_{k+1} the value of s_k in (21) is the one that makes the path going through it to s_{k+1} as the minimum APM (survivor) of all M paths terminating in s_{k+1}.

We call this state s_k as the *survivor state* with respect to state s_{k+1}, other M-1 states are called *non-survivor state*. And s_{k+1} is the tentative detected symbol defined by the transition from the *survivor state* to s_{k+1}.

Step 3: Update and store soft-decision (for the symbols in the time slots from *k-(L+1)* to *k-L*).

- *For all* s_{k+1} *determine the survivor* r *= position {min[c(.)]}, where c(.) is the stored survivor metric vector.* Compute

$$D = \sum_{s_k \in \Omega} \exp\{-\Gamma_{k+1}(s_k, s_{k+1} = r)\} \tag{22}$$

Where W is a set of all M values of s_k having transition with this $s_{k+1} = r$.

- For $s_{k-L} = U_1$ to U_M , update stored soft-decision of the symbols at time instant *k-L*. Compute

$$N_{s_{k-L}} = \sum_{s_k \in \Omega_{s_{k-L}}} \exp\{-\Gamma_{k+1}(s_k, s_{k+1} = r)\}, \text{ where } s_k \in \Omega_{s_{k-L}} \tag{23}$$

Where $\Omega_{s_{k-L}}$ is a set of all values of s_k having transition with this $s_{k+1} = r$ which satisfy the condition that the $(k\text{-}L)^{\text{th}}$ symbol s_{k-L} of paths going through $s_k \in \Omega_{s_{k-L}}$ and terminating at $s_{k+1} = r$ is identical. Compute

$$\mathrm{P}_{s_{k-L}} = \frac{N_{s_{k-L}}}{D}. \tag{24}$$

- Update stored reliability

$$\mathrm{p}_{s_{k-L}} = \mathrm{p}_{s_{k-L}} \cdot \mathrm{P}_{s_{k-L}} \tag{25}$$

Step 4: Let *k=k+1*, if *k<K* go to *Step 1*.

- Output the final hard decision and *M*-ary reliability vector for the last *d*symbols.

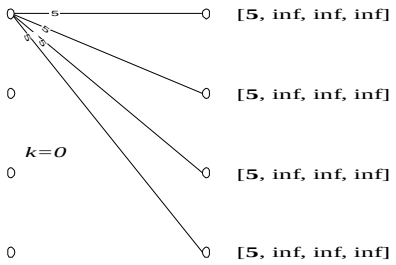
$$\boldsymbol{sym}(k\text{-}\,dk\text{-}1) = \boldsymbol{u}_{(k-\,dk-1)}(r)$$

$$\boldsymbol{Prob}(k\text{-}\,dk\text{-}1) = \mathbf{P}_{(k-\,dk-1)}$$

As an example, a simple four-state trellis, in *which each branch labeled with a length*, from Fig 3 was equalized by SOVE in Figure 6a to 6f. The updating process is performed for the symbols in time slot from *k-(L+1)* to *k-L* and decision delay *d=3-only the globally best path is selected- (V-H-SOVA).*

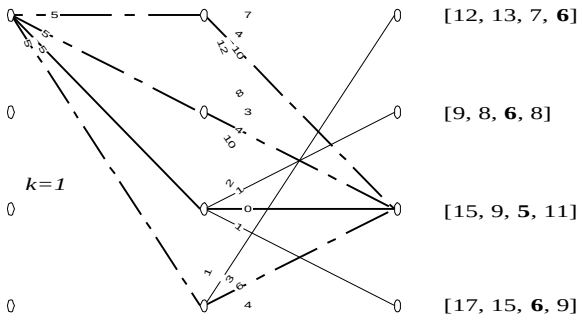
The values of APM are shown in the bracket at the end of trellis diagram.

Example 2



$$c = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}, \; \boldsymbol{u} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}, \; \mathbf{P} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Figure 6a Iterative determination of SOVA at time *k=0*.

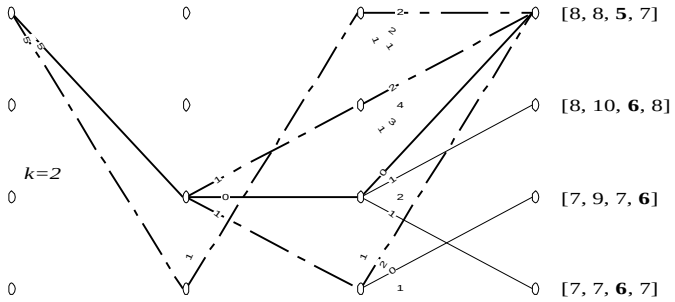


$$D = \sum \exp(-[15 \quad 9 \quad 5 \quad 11])$$

$$\mathrm{P}_{U1} = \frac{\exp(-15)}{D}, \quad \mathrm{P}_{U2} = \frac{\exp(-9)}{D}, \quad \mathrm{P}_{U3} = \frac{\exp(-5)}{D}, \quad \mathrm{P}_{U4} = \frac{\exp(-11)}{D}$$

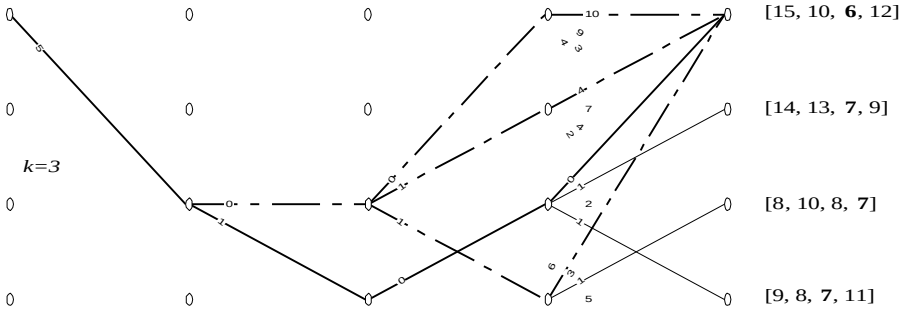
$$c = \begin{bmatrix} 6 \\ 6 \\ \mathbf{5} \\ 6 \end{bmatrix}, \; \boldsymbol{u} = \begin{bmatrix} U_4, U_1 \\ U_3, U_2 \\ U_3, U_3 \\ U_3, U_4 \end{bmatrix}, \; \mathbf{P} = \begin{bmatrix} 1 & 0.00004 \\ 0 & 0.01794 \\ 0 & 0.97959 \\ 0 & 0.00243 \end{bmatrix}$$

Figure 6b Iterative determination of SOVA at time *k=1*.



$$c = \begin{bmatrix} \mathbf{5} \\ 6 \\ 6 \\ 6 \end{bmatrix}, \; \boldsymbol{u} = \begin{bmatrix} U_3, U_3, U_1 \\ U_3, U_3, U_2 \\ U_3, U_4, U_3 \\ U_3, U_3, U_4 \end{bmatrix}, \; \mathbf{P} = \begin{bmatrix} 1 & 0.00004 & 0.04032 \\ 0 & 0.01794 & 0.04032 \\ 0 & 0.97959 & 0.80977 \\ 0 & 0.00243 & 0.10959 \end{bmatrix}$$

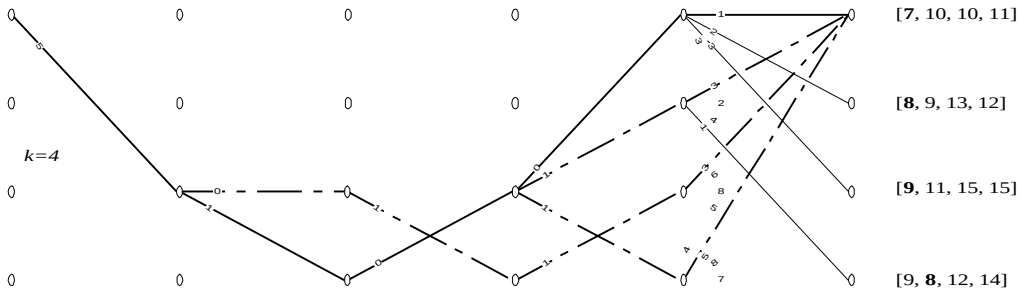
Figure 6c Iterative determination of SOVA at time *k=2*



$$c = \begin{bmatrix} \mathbf{6} \\ 7 \\ 7 \\ 7 \end{bmatrix}, \; \boldsymbol{u} = \begin{bmatrix} \boldsymbol{U_3}, U_4, U_3, U_1 \\ U_3, U_4, U_3, U_2 \\ U_3, U_3, U_4, U_3 \\ U_3, U_4, U_3, U_4 \end{bmatrix}, \; \mathbf{P} = \begin{bmatrix} 0.00004 & 0.04032 & 0.00012 \\ 0.01794 & 0.04032 & 0.01794 \\ 0.97959 & 0.80977 & 0.97951 \\ 0.00243 & 0.10959 & 0.00243 \end{bmatrix}$$

Figure 6d Iterative determination of SOVA at time *k=3*

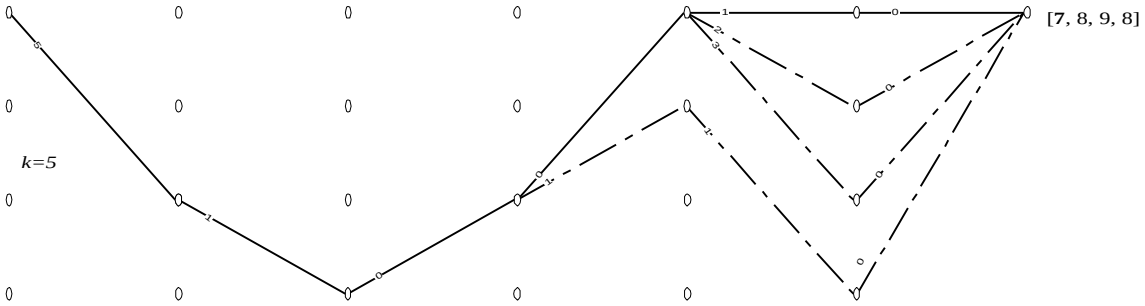
$$\boldsymbol{sym} = [U_3], \ \boldsymbol{Prob} = \begin{bmatrix} 0.00004 \\ 0.01794 \\ 0.97959 \\ 0.00243 \end{bmatrix}$$



$$c = \begin{bmatrix} \mathbf{6} \\ 7 \\ 7 \\ 7 \end{bmatrix}, \ \boldsymbol{u} = \begin{bmatrix} \boldsymbol{U_4}, U_3, U_1, U_1 \\ U_4, U_3, U_1, U_2 \\ U_4, U_3, U_1, U_3 \\ U_4, U_3, U_2, U_4 \end{bmatrix}, \ \mathbf{P} = \begin{bmatrix} 0.04032 & 0.00012 & 0.89454 \\ 0.04032 & 0.01794 & 0.04454 \\ 0.80977 & 0.97951 & 0.04454 \\ 0.10959 & 0.00243 & 0.01638 \end{bmatrix}$$

Figure 6e Iterative determination of SOVA at time *k=4*

$$\boldsymbol{sym} = [U_3, U_4], \ \boldsymbol{Prob} = \begin{bmatrix} 0.00004 & 0.04032 \\ 0.01794 & 0.04032 \\ 0.97959 & 0.80977 \\ 0.00243 & 0.10959 \end{bmatrix}$$



$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{U_3}, \boldsymbol{U_1}, \boldsymbol{U_1}, U_1 \\ U_3, U_1, U_1, U_2 \\ U_3, U_1, U_1, U_3 \\ U_3, U_1, U_1, U_4 \end{bmatrix}, \ \mathbf{P} = \begin{bmatrix} 0.00012 & 0.89454 & 0.53445 \\ 0.01794 & 0.04454 & 0.19661 \\ 0.97951 & 0.04454 & 0.07233 \\ 0.00243 & 0.01638 & 0.19661 \end{bmatrix}$$

Figure 6f Iterative determination of SOVA at time *k=5*

At time $k=K$, output the final hard decision and M-ary reliability vector for the last 3 symbols.

$$\boldsymbol{sym} = [U_3, U_4, U_3, U_1, U_1], \ \boldsymbol{Prob} = \begin{bmatrix} 0.00004 & 0.04032 & 0.00012 & 0.89454 & 0.53445 \\ 0.01794 & 0.04032 & 0.01794 & 0.04454 & 0.19661 \\ 0.97959 & 0.80977 & 0.97951 & 0.04454 & 0.07233 \\ 0.00243 & 0.10959 & 0.00243 & 0.01638 & 0.19661 \end{bmatrix}$$

C. Suboptimum Soft-output Equalizer (SSE)

The Optimum Soft-output Algorithm (OSA) and Suboptimum Soft-output Algorithm (SSA) are improved versions of the MAP algorithm. The OSA generates optimum soft-outputs with only a forward recursion and the number of quantities to be stored and recursively update increases linearly, rather than exponentially, with the decision delay. The SSA is a modified version of the OSA algorithm. The SSA was derived to overcome the other disadvantages of MAP algorithms, It has the following advantages [**Li *et al***, 1995]:

1. The noise variance is not required.
2. Computations are in the logarithmic domain.
3. As in the VA, the add-compare-and-select (ACS) is the main operation.

To derive the steps for SSA, the following definitions are made following [**Li et al,** 1995]. Let $d$is the decision delay, i.e., at time k the algorithm will make a decision on symbol s_{k-v}. For any value of $L £ v £ d$the paths at time k can be divided into M mutually exclusive subsets

$$Q_k(v,s) = \{p_k \mid s_{k-v} = U_i\} \qquad i=1,2,\ldots,M\,. \tag{26}$$

The SSA then selects from each subset $Q_k(v,s)$ the path with the minimum APM. The APM's of the M selected paths are taken as the soft-output decision for signal s_{k-v}. Therefore we have the information packet as a vector of M APM's:

$$\left[\min_{p_k \in Q_k(d,1)} (\Gamma_{k+1}(p_k)), \min_{p_k \in Q_k(d,2)} (\Gamma_{k+1}(p_k)), \ldots, \min_{p_k \in Q_k(d,M)} (\Gamma_{k+1}(p_k)) \right] \tag{27}$$

The *augmented survivor metric* of s_k and s_{k-v} ($L \le v \le d$) is the minimum APM for all paths leading to state s_k *provided that* the symbol transmitted at time *k-v* is s_{k-v}.

$$c_k(s_k, s_{k-v}) = \min[\Gamma_k(p_{k-1}) \mid s_k, s_{k-v}], \quad L \le v \le d \tag{28}$$

s_{k-v} can be any one of *M* possible symbols. The augmented survivor metric of s_{k+1} and s_{k-v} is then calculated as follows.

$$c_{k+1}(s_{k+1}, s_{k-v}) = \min_{s_k}[c_k(s_k, s_{k-v}) + l_{k+1}(s_k, s_{k+1})] \tag{29}$$

For $L \le v \le d$ and the minimization is taken over all *M* states s_k, which leads to state s_{k+1}. For $v=L$ we have

$$c_{k+1}(s_{k+1}, s_{k-L}) = c_k(s_k) + l_{k+1}(s_k, s_{k+1})] \tag{30}$$

In (30) s_k is uniquely determined by s_{k+1} and s_{k-L}. Finally the information packet of SSA is

$$\min_{p_k \in Q_k(d,j)} [\Gamma_{k+1}(p_k)] = \min_{s_{k+1}} [c_{k+1}(s_{k+1}, s_{k-d} = U_j)]. \tag{31}$$

Where $1 \le j \le M$ and the minimizations are over all *M* states s_{k+1}. For each state s_k we define the soft survivor $G(s_k)$ contains *M* augmented survivor metrics $c_k(s_k, s_{k-L-i})$. The extended soft survivor $G_E(s_{k+1})$ of state s_{k+1} is define as a $(d-L+1) \times M$ matrix where the *i*th row of which contains the M augmented survivor metrics $c_{k+1}(s_{k+1}, s_{k-L-i+1})$. Note that in [**Li et al,** 1995] the soft survivor and extended soft survivor are defined to have *(M-1)* columns. This is due to the fact that the augmented survivor metric of s_k (or s_{k+1} respectively) and the hard-decision estimate symbol is equal to the stored survivor $c_k(s_k)$ (or $c_{k+1}(s_{k+1})$ respectively) and hence does not need to be stored again.

Assuming that a soft survivor $G(s_k)$ and the survivor metric $c_k(s_k)$ have been calculated for each state s_k, final steps for SSA are summarized as follows.

Step 1: Calculate the ABM's $l_{k+1}(s_k, s_{k+1})$ for all M^{L+1} ($M.M^L$) branches as in (6).

Step 2: For each state s_{k+1}

1. Calculate the survivor metric $c_{k+1}(s_{k+1})$ as in (8).
2. Calculate the extended soft survivor $G_E(s_{k+1})$ as in (29) and (30).

Step 3: If $k \geq d$ calculate the information packet as in (31).

Step 4: When the information packet has been made, the last row of $G_E(s_{k+1})$ is no longer needed and the extended soft survivor of state s_{k+1} becomes the soft survivor. This soft survivor will be used in step 2 when next symbol is coming.

The following example shows the steps, which are calculated based on the recursive steps as mentioned. We used a simple four-state trellis in Fig 3 and the values of APM from *example 2*. The calculations of extended soft survivor G_E and information packet (Soft-output) are shown.

Example 3

Storage : L=1 here.

ABM_k : ($M \times M$) Additive Branch Metric
APM_k : ($M \times M$) Additive Path Metric
G_E : ($M \times d \times M$) Extend soft survivor metric

$$G_{E_k}(s=1:M, v=1:d, s=1:M) = \begin{bmatrix} g_{1,k-1,1} & \cdots & g_{1,k-1,M} \\ \vdots & & \vdots \\ g_{M,k-1,1} & \cdots & g_{M,k-1,M} \end{bmatrix} \cdots \begin{bmatrix} g_{1,k-d,1} & \cdots & g_{1,k-d,M} \\ \vdots & & \vdots \\ g_{M,k-d,1} & \cdots & g_{M,k-d,M} \end{bmatrix}$$

.

Initialization :

$G_E = [\mathbf{0}][\mathbf{0}][\mathbf{0}]$
ABM=[**0**]
APM=[**0**]

Recursive computation :

In this example the decision delay $d=3$, the extended soft survivor metric can be calculated as,

$$G_{E_k} = \begin{bmatrix} APM_k(1,1) & \cdots & APM_k(1,M) \\ \vdots & & \vdots \\ APM_k(M,1) & \cdots & APM_k(M,M) \end{bmatrix} \begin{bmatrix} g_{1,k-2,1} & \cdots & g_{1,k-2,M} \\ \vdots & & \vdots \\ g_{M,k-2,1} & \cdots & g_{M,k-2,M} \end{bmatrix} \begin{bmatrix} g_{1,k-3,1} & \cdots & g_{1,k-3,M} \\ \vdots & & \vdots \\ g_{M,k-3,1} & \cdots & g_{M,k-3,M} \end{bmatrix}$$

.

Where the elements of G_{E_k} are defined as follows.

$$g_{s_k,k-1,s_{k+1}} = APM_k(s_k, s_{k+1}) \quad (32a)$$

$$g_{s_k,k-2,s_{k+1}} = \min[ABM_k(:, s_{k+1}) + G_{E_{k-1}}(:, k-2, s_{k+1})] \quad (32b)$$

$$g_{s_k,k-3,s_{k+1}} = \min[ABM_k(:, s_{k+1}) + G_{E_{k-1}}(:, k-3, s_{k+1})] \quad (32c)$$

If $k \geq d$ we output information packet;

$$Soft-output(s_k) = \min[G_E(:, k-d, s_k)] \quad (33)$$

The following shows the calculation results at time instant *k=0* to *k=5*. The values of *APM* and *ABM* from previous example are used.

At time k=0 : from Fig 6a

$$ABM = \begin{bmatrix} 5 & 5 & 5 & 5 \\ \inf & \inf & \inf & \inf \\ \inf & \inf & \inf & \inf \\ \inf & \inf & \inf & \inf \end{bmatrix}, \quad APM = \begin{bmatrix} 5 & \inf & \inf & \inf \\ 5 & \inf & \inf & \inf \\ 5 & \inf & \inf & \inf \\ 5 & \inf & \inf & \inf \end{bmatrix},$$

$$G_E = \begin{bmatrix} 5 & \inf & \inf & \inf \\ 5 & \inf & \inf & \inf \\ 5 & \inf & \inf & \inf \\ 5 & \inf & \inf & \inf \end{bmatrix} \begin{bmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix} \begin{bmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix}$$

At time k=1 : from Fig 6b

$$ABM = \begin{bmatrix} 7 & 4 & 10 & 12 \\ 8 & 3 & 4 & 10 \\ 2 & 1 & 0 & 1 \\ 1 & 3 & 6 & 4 \end{bmatrix},\ APM = \begin{bmatrix} 12 & 13 & 7 & 6 \\ 9 & 8 & 6 & 8 \\ 15 & 9 & 5 & 11 \\ 17 & 15 & 6 & 9 \end{bmatrix},$$

$$G_E = \begin{bmatrix} 12 & 13 & 7 & 6 \\ 9 & 8 & 6 & 8 \\ 15 & 9 & 5 & 11 \\ 17 & 15 & 6 & 9 \end{bmatrix} \begin{bmatrix} 6 & \inf & \inf & \inf \\ 6 & \inf & \inf & \inf \\ 5 & \inf & \inf & \inf \\ 6 & \inf & \inf & \inf \end{bmatrix} \begin{bmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix}$$

At time k=2 : from Fig 6c

$$ABM = \begin{bmatrix} 2 & 2 & 1 & 1 \\ 2 & 4 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 0 & 1 \end{bmatrix},\ APM = \begin{bmatrix} 8 & 8 & 5 & 7 \\ 8 & 10 & 6 & 8 \\ 7 & 9 & 7 & 6 \\ 7 & 7 & 6 & 7 \end{bmatrix},$$

$$G_E = \begin{bmatrix} 8 & 8 & 5 & 7 \\ 8 & 10 & 6 & 8 \\ 7 & 9 & 7 & 6 \\ 7 & 7 & 6 & 7 \end{bmatrix} \begin{bmatrix} 11 & 9 & 5 & 8 \\ 13 & 10 & 6 & 8 \\ 12 & 11 & 6 & 7 \\ 10 & 9 & 6 & 7 \end{bmatrix} \begin{bmatrix} 6 & \inf & \inf & \inf \\ 6 & \inf & \inf & \inf \\ 5 & \inf & \inf & \inf \\ 6 & \inf & \inf & \inf \end{bmatrix}$$

At time k=3 : from Fig 6d

We get the Soft-output at this time, and we show some calculations of G_E and Soft-output for more understanding.

$$g_{1,k-2,1} = \min\left(\begin{bmatrix} 10 \\ 4 \\ 0 \\ 6 \end{bmatrix} + \begin{bmatrix} 8 \\ 8 \\ 7 \\ 7 \end{bmatrix}\right) = 7\,, \qquad g_{1,k-2,2} = \min\left(\begin{bmatrix} 10 \\ 4 \\ 0 \\ 6 \end{bmatrix} + \begin{bmatrix} 8 \\ 10 \\ 9 \\ 7 \end{bmatrix}\right) = 9$$

$$g_{2,k-2,1} = \min\left(\begin{bmatrix} 9 \\ 7 \\ 1 \\ 3 \end{bmatrix} + \begin{bmatrix} 8 \\ 8 \\ 7 \\ 7 \end{bmatrix}\right) = 8\,, \qquad g_{2,k-2,2} = \min\left(\begin{bmatrix} 9 \\ 7 \\ 1 \\ 3 \end{bmatrix} + \begin{bmatrix} 8 \\ 10 \\ 9 \\ 7 \end{bmatrix}\right) = 10$$

$$g_{1,k-3,1} = \min\left(\begin{bmatrix}10\\4\\0\\6\end{bmatrix} + \begin{bmatrix}11\\13\\12\\10\end{bmatrix}\right) = 12\ ,$$

$$g_{1,k-3,2} = \min\left(\begin{bmatrix}10\\4\\0\\6\end{bmatrix} + \begin{bmatrix}9\\10\\11\\9\end{bmatrix}\right) = 11$$

$$ABM = \begin{bmatrix}10 & 9 & 3 & 4\\4 & 7 & 4 & 2\\0 & 1 & 2 & 1\\6 & 3 & 1 & 5\end{bmatrix}, APM = \begin{bmatrix}15 & 10 & 6 & 12\\14 & 13 & 7 & 9\\8 & 10 & 8 & 7\\9 & 8 & 7 & 11\end{bmatrix},$$

$$G_E = \begin{bmatrix}15 & 10 & 6 & 12\\14 & 13 & 7 & 9\\8 & 10 & 8 & 7\\9 & 8 & 7 & 11\end{bmatrix}\begin{bmatrix}7 & 9 & 7 & 6\\8 & 10 & 8 & 7\\8 & 8 & 7 & 8\\8 & 10 & 8 & 7\end{bmatrix}\begin{bmatrix}12 & 11 & 6 & 7\\13 & 12 & 7 & 8\\11 & 10 & 7 & 8\\13 & 12 & 7 & 8\end{bmatrix}$$

$$Soft - output = \begin{bmatrix}11\\10\\6\\7\end{bmatrix}$$

At time k=4 : from Fig 6e

$$ABM = \begin{bmatrix}1 & 2 & 3 & 3\\3 & 2 & 4 & 1\\3 & 6 & 8 & 5\\4 & 5 & 8 & 7\end{bmatrix}, APM = \begin{bmatrix}7 & 10 & 10 & 11\\8 & 9 & 13 & 12\\9 & 11 & 15 & 15\\9 & 8 & 12 & 14\end{bmatrix},$$

$$G_E = \begin{bmatrix}7 & 10 & 10 & 11\\8 & 9 & 13 & 12\\9 & 11 & 15 & 15\\9 & 8 & 12 & 14\end{bmatrix}\begin{bmatrix}11 & 11 & 7 & 10\\14 & 12 & 8 & 11\\16 & 13 & 9 & 13\\13 & 13 & 8 & 10\end{bmatrix}\begin{bmatrix}8 & 10 & 8 & 7\\9 & 11 & 9 & 8\\10 & 12 & 10 & 9\\9 & 11 & 9 & 8\end{bmatrix}$$

$$Soft-output=\begin{bmatrix}11 & 8\\ 10 & 10\\ 6 & 8\\ 7 & 7\end{bmatrix}$$

At time k=5 : from Fig 6f

$$ABM=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix},\ APM=\begin{bmatrix}7 & 8 & 9 & 8\\ 7 & 8 & 9 & 8\\ 7 & 8 & 9 & 8\\ 7 & 8 & 9 & 8\end{bmatrix},$$

$$G_E=\begin{bmatrix}7 & 8 & 9 & 8\\ 7 & 8 & 9 & 8\\ 7 & 8 & 9 & 8\\ 7 & 8 & 9 & 8\end{bmatrix}\begin{bmatrix}7 & 8 & 10 & 11\\ 7 & 8 & 10 & 11\\ 7 & 8 & 10 & 11\\ 7 & 8 & 10 & 11\end{bmatrix}\begin{bmatrix}11 & 11 & 7 & 10\\ 11 & 11 & 7 & 10\\ 11 & 11 & 7 & 10\\ 11 & 11 & 7 & 10\end{bmatrix}$$

$$Soft-output=\begin{bmatrix}11 & 8 & 11 & 7 & 7\\ 10 & 10 & 11 & 8 & 8\\ 6 & 8 & 7 & 10 & 9\\ 7 & 7 & 10 & 11 & 8\end{bmatrix}$$

D. Max-Log-MAP Equalizer

In this section we consider Bahl's MAP algorithm [**Bahl et al.,** 1974] and propose a simplified version of such algorithm which does not require knowledge of the noise variance and can avoid the calculation of the nonlinear function which is involved in the probabilities. The simplified version works in logarithmic domain instead of probability domain and is referred to as Max-Log-MAP algorithm [**Chen et al.,** 1997][**Robertson et al.,** 1995].

The basic idea behind the Bahl's MAP algorithm is to make a decision about the transmitted symbol s_k by computing the probability distribution function:

$$p(s_k, r_1^K) = \sum_{s_k, s_{k+1}} p(s_k, r_1^{k-1}) \cdot p(s_{k+1}, r_k \mid s_k) \cdot p(r_{k+1}^K \mid s_{k+1}) \qquad (34)$$

Where the summation in (34) is taken over all pair of states (s_k, s_{k+1}) so that the transitions between them contain the symbol s_k. Note that (34) is obtained using the fact that the distribution of the outputs r_k^K from time *k* to time *K* does not depend on the outputs r_1^{k-1} from time 1 to time *k-1* for a given state s_k. The *multiplicative branch metric* (MBM) of branch x_k is calculated as,

$$m_{k+1}(x_k) = m_{k+1}(s_k, s_{k+1}) = p(s_{k+1}, r_k \mid s_k) \qquad (35)$$

The forward and backward recursion probability functions, $p(s_k, r_1^{k-1})$ and $p(r_k^K \mid s_k)$ respectively, can be defined as,

$$a_k(s_k) = p(s_k, r_1^{k-1}) = \sum_{s_{k-1}} a_{k-1}(s_{k-1}) \cdot m_k(s_{k-1}, s_k) \qquad (36)$$

$$b_k(s_k) = p(r_k^K \mid s_k) = \sum_{s_{k+1}} b_{k+1}(s_{k+1}) \cdot m_{k+1}(s_k, s_{k+1}) \qquad (37)$$

The output of the Bahl's MAP algorithm can be obtained from the MBM of each branch x_k and from those forward and backward recursions as,

$$p(s_k, r_1^K) = \sum_{s_k, s_{k+1}} a_k(s_k) \cdot m_{k+1}(s_k, s_{k+1}) \cdot b_{k+1}(s_{k+1}) \qquad (38)$$

If we assumed that the trellis states start and stop at zero state then we have the boundary conditions of the forward and backward recursions as follows.

$$a_1(0)=1 \text{ and } a_1(s_k)=0, \text{ for } s_k \neq 0$$
$$b_k(0)=1 \text{ and } b_k(s_k)=0, \text{ for } s_k \neq 0$$

The Bahl's MAP algorithm described above requires the knowledge of the noise variance. The probabilities in (34) to (38) are the summation of several terms. However, for a relatively high signal to noise ratio (E_b/N_o), only one term dominates. As a result, we only need to consider such term. Let $\bar{a}(\cdot) = Log(a(\cdot))$, $\bar{b}(\cdot) = Log(b(\cdot))$ and $\bar{m}(\cdot) = Log(m(\cdot))$ then we can use the approximation

$$Log(e^{d_1} + e^{d_2} + \ldots + e^{d_n}) \approx \max_{i \in \{1,\ldots,n\}}[d_i] \quad \text{to obtain the following}$$

equations.

$$\bar{m}_{k+1}(s_k, s_{k+1}) = -\frac{l_{k+1}(x_k)}{N_0} + Log(P(s_{k+1} \mid s_k)) + Log(\frac{1}{pN_0}) \qquad (39)$$

$$\bar{a}_k(s_k) = \max_{s_{k-1}} \{\bar{a}_{k-1}(s_{k-1}) + \bar{m}_k(s_{k-1}, s_k)\} \qquad (40)$$

$$\bar{b}_k(s_k) = \max_{s_{k+1}} \{\bar{b}_{k+1}(s_{k+1}) + \bar{m}_{k+1}(s_k, s_{k+1})\} \qquad (41)$$

$$Log(s_k, r_1^K) = \max_{s_k, s_{k+1}} \{\bar{a}_k(s_k) + \bar{m}_{k+1}(s_k, s_{k+1}) + \bar{b}_{k+1}(s_{k+1})\} \qquad (42)$$

The last two components of (39) are constant for a given channel profile, they have no effect on the maximization operation and can be neglected. Then (39) can be simplified to:

$$\bar{m}_{k+1}(s_k, s_{k+1}) = -l_{k+1}(x_k) \qquad (43)$$

Now the implementation steps for Bahl's Max-Log-MAP algorithm can be summarized as follows.

Step 1: Initialization,

$$\bar{a}_1(0) = 0 \text{ and } \bar{a}_1(s_k) = -\inf \text{ , for } s_k \neq 0$$
$$\bar{b}_K(0) = 0 \text{ and } \bar{b}_K(s_k) = -\inf \text{ , for } s_k \neq 0$$

Step 2: For each observation r_k, $\bar{m}_{k+1}(s_k, s_{k+1})$ and $\bar{a}_k(s_k)$ are computed using (43) and (40) respectively.

Step 3: When sequence r_1^K has been completely received, $\bar{b}_k(s_k)$ are computed using (41).

Step 4: Calculate $Soft - output = -Log(p(s_k, r_1^K))$ for each transmitted symbol s_k as in (42).

Again, the simple four-state trellis in Fig. 3 was equalized by Max-Log-MAP algorithm in following example.

Example 4

$$\bar{m} = \begin{bmatrix} -5 & -5 & -5 & -5 \\ -\inf & -\inf & -\inf & -\inf \\ -\inf & -\inf & -\inf & -\inf \\ -\inf & -\inf & -\inf & -\inf \end{bmatrix} \begin{bmatrix} -1 & -4 & -10 & -12 \\ -8 & -3 & -4 & -10 \\ -2 & -1 & 0 & -1 \\ -1 & -3 & -6 & -4 \end{bmatrix} \begin{bmatrix} -2 & -2 & -1 & -1 \\ -2 & -4 & -3 & -1 \\ 0 & -1 & -2 & -1 \\ -1 & -2 & 0 & -1 \end{bmatrix} \begin{bmatrix} -10 & -9 & -3 & -4 \\ -4 & -7 & -4 & -2 \\ 0 & -1 & -2 & -1 \\ -6 & -3 & -1 & -5 \end{bmatrix} \begin{bmatrix} -1 & -2 & -3 & -3 \\ -3 & -2 & -4 & -1 \\ -3 & -6 & -8 & -5 \\ -4 & -5 & -8 & -7 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\bar{a} = \begin{bmatrix} 0 \\ -\inf \\ -\inf \\ -\inf \end{bmatrix} \begin{bmatrix} -5 \\ -5 \\ -5 \\ -5 \end{bmatrix} \begin{bmatrix} -6 \\ -6 \\ -5 \\ -6 \end{bmatrix} \begin{bmatrix} -5 \\ -6 \\ -5 \\ -6 \end{bmatrix} \begin{bmatrix} -6 \\ -7 \\ -7 \\ -7 \end{bmatrix} \begin{bmatrix} -7 \\ -8 \\ -9 \\ -8 \end{bmatrix},$$

$$\bar{b} = \begin{bmatrix} -3 \\ -7 \\ -2 \\ -3 \end{bmatrix} \begin{bmatrix} -2 \\ -4 \\ -3 \\ -1 \end{bmatrix} \begin{bmatrix} -6 \\ -5 \\ -1 \\ -4 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -3 \\ -4 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ -\inf \\ -\inf \\ -\inf \end{bmatrix}$$

$$Soft-output = \begin{bmatrix} 8 \\ 12 \\ 7 \\ 8 \end{bmatrix} \begin{bmatrix} 8 \\ 10 \\ 8 \\ 7 \end{bmatrix} \begin{bmatrix} 11 \\ 11 \\ 7 \\ 10 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \\ 10 \\ 11 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \\ 9 \\ 8 \end{bmatrix}$$

E. ***Complexity and Memory Requirement of the Equalizers***

We assume that the equalizer is employed with *M*-ary signals, the channel memory is *L*, the decision delay is *D*. It is also assumed that the ABM of all transitions are already known. The complexity and the memory requirement of different kinds of equalizers are considered as follows:

With the above assumption, the number of states in the ISI trellis is M^L.

- VE

There are *M* branches merging to each state s_{k+1}. Therefore, it requires *M* additions and (*M*-1) comparisons to find the APM for each s_{k+1}. The complexity of VE is then

- M^{L+1} additions per one trellis interval
- $M^L \cdot (M-1)$ comparisons per one trellis interval

The memory required for storing the survival paths and the survival metrics is

$$MEM_{VE} = D \cdot M^L + M^L = (D+1)M^L \quad \text{units.} \tag{44}$$

- SOVE

Additional operations from VE are performed to calculate the soft-output information. The following steps are required.

1) Determining the state with global minimum APM takes M^L-1 comparisons.

2) Requiring M takings exponential and M-1 additions.
3) Determining the soft-output information for each symbol takes M divisions.

Therefore, the complexity of SOVE is

- $M^{L+1} + M - 1$ additions per one trellis interval
- $M^{L+1} - 1$ comparisons per one trellis interval
- M takings exponential per one trellis interval
- M divisions per one trellis interval.

Additional memory is needed from VE for storing the soft-output information. Therefore, the memory requirement of SOVE is

$$MEM_{SOVE} = (D+1)M^{L} + D \cdot M \text{ units} \quad (45)$$

- SSE

The following steps are required to calculate the soft-output information.

1) For the symbol s_{k-L}, the minimum APM at each s_{k+1} will be the soft-survivor metric. It requires one addition per state per symbol s_{k-L}. The total number of additions is $M^{L} \cdot M = M^{L+1}$.
2) To update the extended soft-survivor metric, for each s_{k-d}, where $L < d < D$, and each s_{k+1}, M additions and M-1 comparisons are required. Therefore, the total number of additions is $M^{L+2} \cdot (D-L)$ and the total number of comparisons is $M^{L+1} \cdot (D-L) \cdot (M-1)$.
3) To select the minimum soft-output information for each symbol requires another
   $M^{L} - 1$ comparisons. The number of comparisons in this case is $M \cdot (M^{L} - 1)$.

Hence, the complexity of SSE is

- $M^{L+1} + M^{L+2} \cdot (D-L)$ additions per one trellis interval
- $M^{L+1} \cdot (D-L) \cdot (M-1) + M \cdot (M^{L}-1)$ comparisons per one trellis interval.

The memory required for storing soft-survivor metric for SSE is

$$MEM_{SSE} = M^{L+1} \cdot (D-L) \text{ units.} \tag{46}$$

- Max-Log-MAP equalizer

The following steps are required to calculate the soft-output information.

1) Calculating $\bar{a}_k(s_k)$ requires M additions and M-1 comparisons for each s_k. Therefore, for all states s_k, it requires M^{L+1} additions and $M^{L} \cdot (M-1)$ comparisons.
2) Calculating $\bar{b}_k(s_k)$ requires the same number of computation as for $\bar{a}_k(s_k)$.
3) The number of branches representing a symbol in one time slot is $M \cdot M^{L-1}$. Each branch requires two additions. There are M symbols, then the total number of additions is then $2M^{L+1}$. The number of comparisons is $M \cdot (M^{L}-1)$.

Therefore, the complexity of Max-Log-MAP equalizer is

- $4M^{L+1}$ additions per one trellis interval
- $3M^{L+1} - 2M^{L} - M$ comparisons per one trellis interval.

The memory required for storing the ABM, $\bar{a}_k(s_k)$ and $\bar{b}_k(s_k)$ in the Max-Log-MAP case is

$$MEM_{Max-Log-MAP} = N \cdot (M^{2L} + 2 \cdot M^{L}) \text{ units,} \tag{47}$$

where N is the number of transmitted symbols.

Summary:

- *Maximum likelihood sequence estimation (MLSE)*

  MLSE chooses the sequence x_0^{N-1} for which the probability $p(y_0^{N-1} \mid x_0^{N-1})$ is maximum. This corresponds to choosing the sequence that minimizes the Euclidean distance between the received sequence and the possible sequence.

  - *Viterbi equalizer (VE)*

    Viterbi algorithm (VA) is a recursive estimation algorithm that is usually applied with MLSE [1]. Therefore, MLSE is also referred to VE. It provides only the hard-output sequence.

  - *Soft-output Viterbi equalizer (SOVE)*

    VA is modified to produce soft information to be used in the decoding of outer codes [2]. Soft-output Viterbi Algorithm (SOVA) accepts and delivers soft sample values in addition to the hard-output sequence.

- *Maximum a posteriori probability (MAP)*

  In contrast to MLSE, MAP tries to minimize the symbol error probability by choosing the symbols which maximize $p(x_k \mid y_0^{N-1})$. The output of MAP is soft-output in terms of a posteriori probability.

  - *Sub-optimum soft-output equalizer (SSE)*

    SSE delivers soft-output for each symbol at time *k-D* as the minimum additive path metrics at time *k* corresponding to that symbol. The calculation is in the logarithmic domain, the noise variance is not required and the algorithm requires only forward recursion [3].

- *Max-Log-MAP equalizer*

  It is derived from the optimum Bahl's MAP algorithm. The calculation is in the logarithmic domain, the noise variance is not required and the algorithm requires both forward and backward recursion. Therefore, the disadvantage is that the equalizer has to wait for the whole block to be received before the decision can be made [4,5].

Decoding followed by Equalization:

The branch metrics of the Viterbi decoder depends on what type of equalizers is employed. For VE, the branch metrics will be the Euclidean distance between symbol on the branch trellis and the received decision symbol. For SOVE, the branch metrics will be the Euclidean distance like that in VE and subtracted by the log-likelihood of the soft-output information. For SSE and Max-Log-MAP, the soft-output from the equalizer can be directly used as branch metrics.

***References*:**

1. G. D. Forney, *Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference*, IEEE Trans. Inform. Theory, vol. IT-18, no. 3, pp. 363-378, May 1972.

2. J. Hagenauer and P. Hoeher, *A Viterbi Algorithm with Soft-Decision Outputs and its Applications*, in Globecom' 89 Conf. Rec., Dallas, Texas, vol. 3, pp. 47.1.1-47.1.7, Nov 1989.

3. Y. Li, B. Vucetic and Sato, Y., *Optimum Soft-Output Detection for Channels with Intersymbol Interference*, IEEE Trans. Commun., vol. 44, no.1, pp. 50-58, Feb. 1995.

4. N. H. Ha, R.M.A.P. Rajatheva, "Turbo Trellis Coded Modulation for Frequency Selective Fading Channels", ICC'99 Victoria, Canada, June.

5. D. N. Dung, R.M.A.P. Rajatheva, "Turbo equalization for Non-binary Coded Modulation Schemes over frequency Selective fading Channels", VTC2000, Tokyo, Japan.

ITERATIVE RECEIVER FOR CODED MODULATION SCHEMES

This section investigates the performance of coded modulation schemes over frequency selective Rayleigh fading channels. The combined receiver doing joint equalization and decoding of coded signals which significantly cancels out the detrimental effect of intersymbol interference is presented. The proposed receiver is applied to trellis coded modulation and turbo trellis coded modulation schemes. The performance of the new receiver is compared with the conventional approach (i.e. separate equalizer and decoder) through simulation. It is shown that the new receiver considerably improves the error performance after only a small number of iterations.

Introduction

The introduction of turbo codes (**Berrou** et al., 1993) has resulted in various applications in wireless communications, deep space communications, etc. With the remarkable performance of turbo codes, it is natural to combine turbo codes with multilevel modulation schemes in order to obtain large coding gains and high bandwidth efficiency over both AWGN and fading channels. Turbo trellis coded modulation (TTCM) proposed in (**Robertson** and **Woerz**, 1995) is the extension of the turbo codes where the component codes are replaced by Ungerboeck TCM codes in the recursive systematic forms to retain the advantages of both classical turbo codes and TCM codes.

Turbo equalization introduced by authors in (**Douillard** et al., 1995) was the first effort to apply iterative principle to equalizing convolutionally coded signals transmitted over intersymbol interference channels. Recently, there have been some researches that were focusing on turbo equalization concept for convolutionally coded systems (**Bauch** et al., 1997), turbo coded systems (**Raphaeli** and **Zarai**, 1997), and serially concatenated systems (**Li** and **Mow**, 1999). In all of those, binary modulation was considered. The equalization and decoding blocks were basically based on soft-in soft-out (SISO) algorithms. In the work of **Glavieux** et al. (1997), multilevel modulation was considered but the equalizer used there was a kind of an adaptive one. It is well known that performance of the multilevel coded modulation schemes over frequency selective channels can be improved by employing SISO modules for both the equalizer and the decoder. In the conventional approach, the reliability information about the coded symbols from the equalizer is

transferred to the decoder, but the equalizer does not utilize the corresponding soft values out of the decoder.

In what follows, we use iterative decoding process to do the equalization and decoding of multilevel coded modulation signals over frequency selective Rayleigh fading channels. Since we are working at the symbol level, the decoder is modified to be able to output the soft information of both uncoded and coded symbols. In the multilevel coded modulation schemes, since the systematic bits are transmitted together with the parity bits in the same symbol, noise perturbs simultaneously both parity component and systematic component. The soft outputs of the decoder must contain soft information of the coded symbols to be used at the equalizer as the a priori information. The decoder is modified from the conventional SISO decoder. It is desirable to keep the new receiver similar to the conventional one as much as possible. In the decoder, a functional block is added to give out the coded symbols' extrinsic soft information. For the SISO blocks, the low-complexity, sub-optimum Max-Log-MAP algorithm is used.

The remainder of this section is organized as follows. Section 2 describes the system model. The explicit expansion of the SISO-MAP algorithm for the equalizer and decoder is discussed in section 3.

2 System Model

Figure 4.1 shows the block diagram of the communication system we are going to investigate.

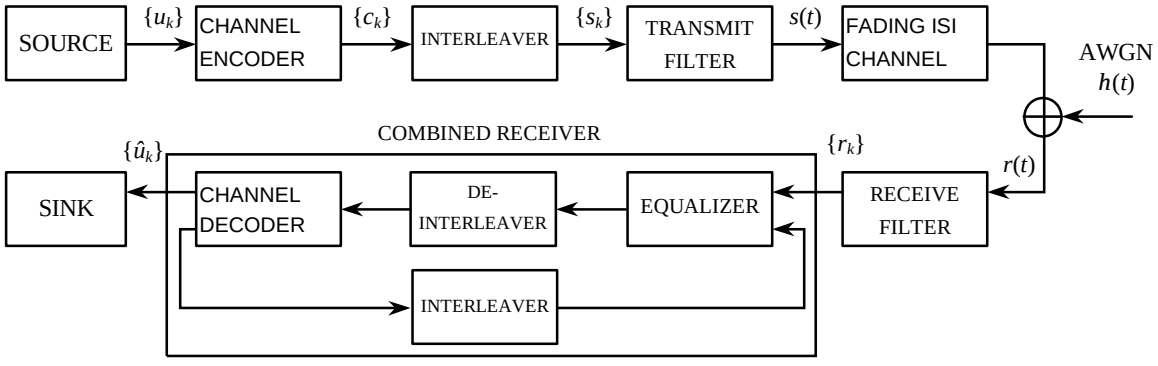


Figure 1 System model for communication system with combined channel

The information sequence $\{u_k\}$, with u_k takes on values 1 or 0, which is supposed to be equally likely distributed, is coming our of the source. The information sequence is encoded by a coded modulation encoder, which can be TCM or TTCM. The output symbol sequence $\{c_k\}$, where c_k is taken from one of M possible values (M-ary modulation), is block interleaved to break up burst errors caused by the fading channel before being transmitted into the channel. We assumed that the interleaving depth is infinite, so that the fading is uncorrelated. In reality, even this assumption cannot be met, by proper use of interleaving depth, which is greater than the maximum fade duration, the correlation is negligible. The transmit and receive filters are used for spectrum shaping. They are chosen to be Nyquist filters and are matched to each other. Let $p(t)$ be the impulse response of the filters, the transmitted signal can be expressed as

$$s(t) = \sum_k s_k \, p(t - kT) \tag{1}$$

where T is the symbol interval, s_k is the interleaved version of the TCM coded signal constellation point c_k. The overall channel model consists of a transmit filter, the fading channel, and a receive filter. It can often be modeled as a finite impulse response (FIR) filter having (L+1) taps $g(k)$ = ($g_0(k)$, $g_1(k)$, . . . , $g_L(k)$) at symbol instant k (**Proakis**, 1995).

The channel output at time instant kT is corrupted by the AWGN h_k, and the received signal r_k is then given by

$$r_k = \sum_{i=0}^{L} s_{k-i} \, g_i(k) + h_k \, . \tag{2}$$

The additive Gaussian noise, after being filtered by the receive filter, becomes correlated in general. We still assume that the correlation is negligible so that $\{h_k\}$ is the sequence of statistically independent and identically distributed (i.i.d.) complex Gaussian noise samples. The probability density function (pdf) of each sample is given by

$$r(h_k) = \frac{1}{2ps_N^2} \exp\left(- \frac{|h_k|^2}{2s_N^2} \right) \tag{3}$$

where s_N^2 is the noise variance in each signal space coordinate and $|h_k|$ is the magnitude of the complex noise sample h_k.

3 Iterative Equalization and Decoding

Since we are going to deal with trellis diagrams, the following definitions are used to describe the channel as well as the encoder trellis diagrams. These definitions mostly follow (**Forney**, 1973).

State: A state s_k at time *k* is defined as

$$s_k = (s_{k-L}, s_{k-L+1}, ..., s_{k-2}, s_{k-1}), \text{ where } s_i = 0 \text{ while } i < 0. \quad (4)$$

Transition: A transition x_k is defined as

$$\begin{aligned} x_k &= (s_k, s_{k+1}) = (s_{k-L}, s_{k-L+1}, ..., s_{k-2}, s_{k-1}, s_k), \\ &\text{where } s_i = 0 \text{ while } i < 0 \\ &= (s_k, s_k). \end{aligned} \quad (5)$$

Additive Branch Metric (ABM): An ABM is the *squared Euclidean distance* caused by the transition x_k

$$l_{k+1}(x_k) = l_{k+1}(s_k, s_{k+1}) = \left| r_k - \sum_{i=0}^{L} s_{k-i} g_i(k) \right|^2. \quad (6)$$

Multiplicative Branch Metric (MBM): An MBM of branch x_k is defined as

$$\begin{aligned} m_{k+1}(x_k) &= m_{k+1}(s_k, s_{k+1}) = p(s_{k+1}, r_k \mid s_k) \\ &= p(r_k \mid s_k, s_{k+1}) P(s_{k+1} \mid s_k) \\ &= r(l_{k+1}(x_k)) P(s_{k+1} \mid s_k) \end{aligned} \quad (7)$$

where $r(l_{k+1}(x_k)) = p(r_k \mid s_k, s_{k+1})$.

SISO-MAP Module

Consider a trellis code with *m*-tuple input and *n*-tuple output. Let a group of *m* information bits at time *k* be represented by a symbol u_k, and a group of *n* output bits be represented by a symbol c_k. We simply call u_k and c_k the uncoded and coded symbols respectively. If the code is systematic, c_k is composed of u_k and the parity bits. Associating with each transition x_k is the uncoded symbol u_k and the coded symbol c_k.

Basically, an SISO module is a four port device with two inputs and two outputs. The inputs to the module are the (logarithm of) a priori probability sequences of the uncoded and coded symbols denoted by $Li(u_k)$ and $Li(c_k)$ respectively. The outputs of it are the (logarithm of) extrinsic probability sequences of the uncoded and coded symbols, which are correspondingly denoted by $Lo(u_k)$ and $Lo(c_k)$.

Equalizer SISO-MAP Module

Clearly, we can treat the equivalent discrete-time channel as a rate one feed forward time variant convolutional encoder. The equalizer SISO module does the estimation of the transmitted channel symbols s_k basing on the complete received signal sequence $\{r_k;\ k=1, 2 .., K\}$. It calculates the following a posteriori probability (**Bahl** et al., 1974)

$$P\left(s_k \mid r_1^K\right) = H_1 \cdot \sum_{s_k, s_{k+1}:s_k} p\left(s_k, r_1^{k-1}\right) \cdot p\left(s_{k+1}, r_k \mid s_k\right) \cdot p\left(r_{k+1}^K \mid s_{k+1}\right) \quad (8)$$

where the summation is taken over all pairs of states (s_k, s_{k+1}) so that the transitions between them contain the symbol s_k and H_1 is a normalization constant.

Let

$$a_k(s_k) = p\left(s_k, r_1^{k-1}\right);\ m_{k+1}(x_k) = p\left(s_{k+1}, r_k \mid s_k\right);\ b_k(s_k) = p\left(r_k^K \mid s_k\right).$$

We have the following relations:

- **Transition probabilities**

$$m_{k+1}(x_k) = m_{k+1}(s_k, s_{k+1}) = r(l_{k+1}(x_k))P(s_{k+1} \mid s_k)$$
$$= \begin{cases} r(l_{k+1}(x_k))P(s_k), & \text{while } x_k \text{ contains } s_k \\ 0, & \text{elsewhere.} \end{cases} \quad (9)$$

- **Forward recursion**

$$a_k(s_k) = \sum_{s_{k-1}} p\left(s_{k-1}, r_1^{k-2}\right) \cdot m_k(x_{k-1})$$
$$= \sum_{s_{k-1}} a_{k-1}(s_{k-1}) \cdot m_k(s_{k-1}, s_k). \quad (10)$$

- **Backward recursion**

$$b_k(s_k) = \sum_{s_{k+1}} p\left(r_{k+1}^K \mid s_{k+1}\right) \cdot p(s_{k+1}, r_k \mid s_k)$$
$$= \sum_{s_{k+1}} b_{k+1}(s_{k+1}) \cdot m_{k+1}(s_k, s_{k+1}). \quad (11)$$

The output of the equalizer SISO-MAP can be obtained from the MBM of each branch x_k and from the forward and backward recursions as

$$p\left(s_k \mid r_1^K\right) = H_1 \cdot \sum_{s_k, s_{k+1}:s_k} a_k(s_k) \cdot m_{k+1}(s_k, s_{k+1}) \cdot b_{k+1}(s_{k+1}). \quad (12)$$

If we assume that the trellis states stay at zero state at the beginning as well as at the end of the received signal sequence, the initializations of $a_k(s_k)$ and $b_k(s_k)$ are as follows:

$$a_1(0) = 1 \text{ and } a_1(s_k) = 0, \quad \forall s_k \neq 0. \quad (13)$$

$$b_K(0) = 1 \text{ and } b_K(s_k) = 0, \quad \forall s_k \neq 0. \quad (14)$$

From Eqs. (9)-(12), we observe that the a priori probability $P(s_k)$ can be extracted from the summation in Eq. (12). We define the new quantity

$$P_e\left(s_k \mid r_1^K\right) = H_2 \frac{P\left(s_k \mid r_1^K\right)}{P(s_k)} \tag{15}$$

where H_2 is a normalization constant.

It is clear that $P_e\left(s_k \mid r_1^K\right)$ does not depend on the a priori probability $P(s_k)$. In the nature of turbo decoding, $P_e\left(s_k \mid r_1^K\right)$ is called extrinsic probability.

Now, we define the scaled transition probability $m_{k+1}^*(x_k)$ as

$$\begin{aligned} m_{k+1}^*(x_k) &= \frac{m_{k+1}(x_k)}{P(s_k)} = \frac{r(l_{k+1}(x_k))P(s_{k+1} \mid s_k)}{P(s_k)} \\ &= \begin{cases} r(l_{k+1}(x_k)), & \text{while } x_k \text{ contains } s_k \\ 0, & \text{elsewhere.} \end{cases} \end{aligned} \tag{16}$$

The extrinsic probability $P_e\left(s_k \mid r_1^K\right)$ is calculated through the following equation

$$\begin{aligned} &p_e\left(s_k \mid r_1^K\right) = \\ &H_1 \cdot H_2 \cdot \sum_{s_k, s_{k+1}:s_k} a_k(s_k) \cdot m_{k+1}^*(s_k, s_{k+1}) \cdot b_{k+1}(s_{k+1}). \end{aligned} \tag{17}$$

Because of the monotonicity of the logarithm function, we can simplify all the previous probability calculations by taking the natural logarithm of them. Define the following

$$Li(s_k) = \mathrm{Log}P(s_k). \tag{18}$$

$$Lo(s_k) = LogP_e\left(s_k \mid r_1^K\right). \tag{19}$$

$$\bar{a}_k(s_k) = Log\,a_k(s_k). \tag{20}$$

$$\bar{b}_k(s_k) = Log\,b_k(s_k). \tag{21}$$

$$\bar{m}_{k+1}(s_k, s_{k+1}) = Log\,m_{k+1}(s_k, s_{k+1}). \tag{22}$$

$$\bar{m}^*_{k+1}(s_k, s_{k+1}) = Log\,m^*_{k+1}(s_k, s_{k+1}). \tag{23}$$

$$L^c(s_k) = -\frac{l_{k+1}(x_k)}{2\sigma_N^2} - Log\left(2\pi\sigma_N^2\right) \quad \text{(channel information).} \tag{24}$$

The relation between $\bar{m}^*_{k+1}(x_k)$ and the channel information $L^c(s_k)$ is as follows:

$$\bar{m}^*_{k+1}(x_k) = \begin{cases} L^c(s_k), & \text{while } x_k \text{ contains } s_k \\ -\infty, & \text{elsewhere.} \end{cases} \tag{25}$$

When working at medium high SNR regions, we can use the approximation

$$Log(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_n}) \approx \max_{i \in \{1,\ldots,n\}} \delta_i \tag{26}$$

to arrive to Max-Log-MAP algorithm (**Robertson** et al., 1995), (**Ha** and **Rajatheva,** 1999)

$$\begin{aligned} \bar{m}_{k+1}(s_k, s_{k+1}) &= -\frac{l_{k+1}(x_k)}{2\sigma_N^2} - Log\left(2\pi\sigma_N^2\right) + LogP(s_{k+1} \mid s_k) \\ &= \begin{cases} Li(s_k) + L^c(s_k), & \text{while } x_k \text{ contains } s_k \\ -\infty, & \text{elsewhere.} \end{cases} \end{aligned} \tag{27}$$

$$\bar{a}_k(s_k) = \max_{s_{k-1}} \{\bar{a}_{k-1}(s_{k-1}) + \bar{m}_k(s_{k-1}, s_k)\}. \tag{28}$$

$$\bar{b}_k(s_k) = \max_{s_{k+1}} \{\bar{b}_{k+1}(s_{k+1}) + \bar{m}_{k+1}(s_k, s_{k+1})\}. \tag{29}$$

$$Lo(s_k) = \max_{s_k, s_{k+1}: s_k} \left\{\bar{a}_k(s_k) + \bar{m}^*_{k+1}(s_k, s_{k+1}) + \bar{b}_{k+1}(s_{k+1})\right\} + h_1 \tag{30}$$

where h_1 is a constant used to avoid numerical overflow.

We summarize the Max-Log-MAP SISO algorithm for the equalizer as follows:

(1) Initialization:

$$\bar{a}_1(0) = 0 \text{ and } \bar{a}_1(s_k) = -\infty, \quad \forall s_k \neq 0. \tag{31}$$

$$\bar{b}_K(0) = 0 \text{ and } \bar{b}_K(s_k) = -\infty, \quad \forall s_k \neq 0. \tag{32}$$

(2) Forward recursion at each time *k*: Compute $\bar{a}_k(s_k)$ through Eq. (28).

(3) Backward recursion, when r_i^K has been completely received: Compute $\bar{b}_k(s_k)$ through Eq. (29).

(4) Log extrinsic probability: Compute $Lo(s_k)$ through Eq. (30).

Decoder SISO-MAP Module

The SISO-MAP module for the trellis decoder calculates the logarithm of extrinsic probabilities of the uncoded and coded symbols in the similar way as that of the equalizer, i.e., as Eq. (30).

$$Lo(u_k) = \max_{s_k, s_{k+1}: u_k} \left\{\bar{a}_k(s_k) + \bar{m}^{*,c}_{k+1}(s_k, s_{k+1}) + \bar{b}_{k+1}(s_{k+1})\right\} + h_2. \tag{33}$$

$$Lo(c_k) = \max_{s_k, s_{k+1}: c_k} \left\{\bar{a}_k(s_k) + \bar{m}^{*,u}_{k+1}(s_k, s_{k+1}) + \bar{b}_{k+1}(s_{k+1})\right\} + h_3. \tag{34}$$

The main difference compared with the equalizer SISO-MAP is in the calculation of the new extrinsic probability for the coded symbols,

and the transition probabilities due to the fact that the decoder is not directly connected to the channel. The decoder SISO-MAP uses the available probabilistic information from the equalizer to calculate its transition probabilities as follows

$$\bar{m}_{k+1}(s_k, s_{k+1}) = \begin{cases} Li(c_k) + Li(u_k), & \text{while } x_k \text{ contains } u_k \text{ and } c_k \\ -\infty, & \text{elsewhere} \end{cases} \quad (35)$$

where $Li(c_k)$ is set equal to the extrinsic value coming out of the equalizer, i.e. equal to $Lo(s_k)$ for $c_k \equiv s_k$.

The scaled transition probabilities $\bar{m}^{*,c}_{k+1}(s_k, s_{k+1})$ and $\bar{m}^{*,u}_{k+1}(s_k, s_{k+1})$ in the summations from Eqs. (33) and (34) are calculated as

$$\bar{m}^{*,c}_{k+1}(x_k) = \begin{cases} Li(c_k), & \text{while } x_k \text{ contains } c_k \\ -\infty, & \text{elsewhere.} \end{cases} \quad (36)$$

$$\bar{m}^{*,u}_{k+1}(x_k) = \begin{cases} Li(u_k), & \text{while } x_k \text{ contains } u_k \\ -\infty, & \text{elsewhere.} \end{cases} \quad (37)$$

The forward and backward recursions are still calculated the same as Eqs. (28) and (29) respectively.

The Combined Receiver for TCM

Figure 3 shows the combined receiver applied to TCM. For the TCM scheme, the equalizer observes the distorted version of transmitted signal all the time. The equalizer SISO module does not need to include the constant $-Log(2ps^2_N)$ in the calculation of transition probabilities as far as the maximization is concerned. At the first iteration, we set the a priori probability to the equalizer equal zero. After that, the a priori information is extracted from the TCM decoder SISO-MAP.

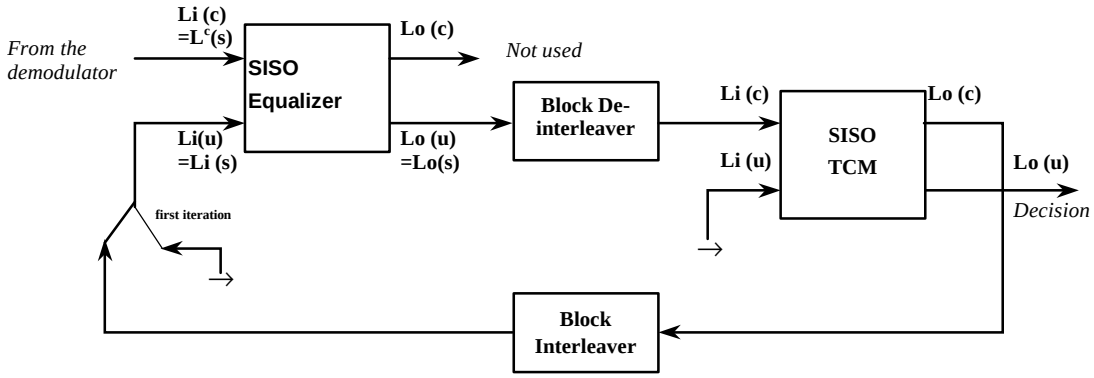


Figure 3 Combined receiver for TCM

The Combined Receiver for TTCM

Figure 4 shows the combined receiver applied to TTCM. The equalizer alternatively observes the distorted version of the transmitted signals form the upper and lower component TCM encoders. We set the a priori probability to the equalizer SISO the same as TCM scheme. For the first decoding step of the upper decoder (half iteration), we set $Li(u) = -mLog2$; where $m/(m+1)$ is the code rate of component TCM's.

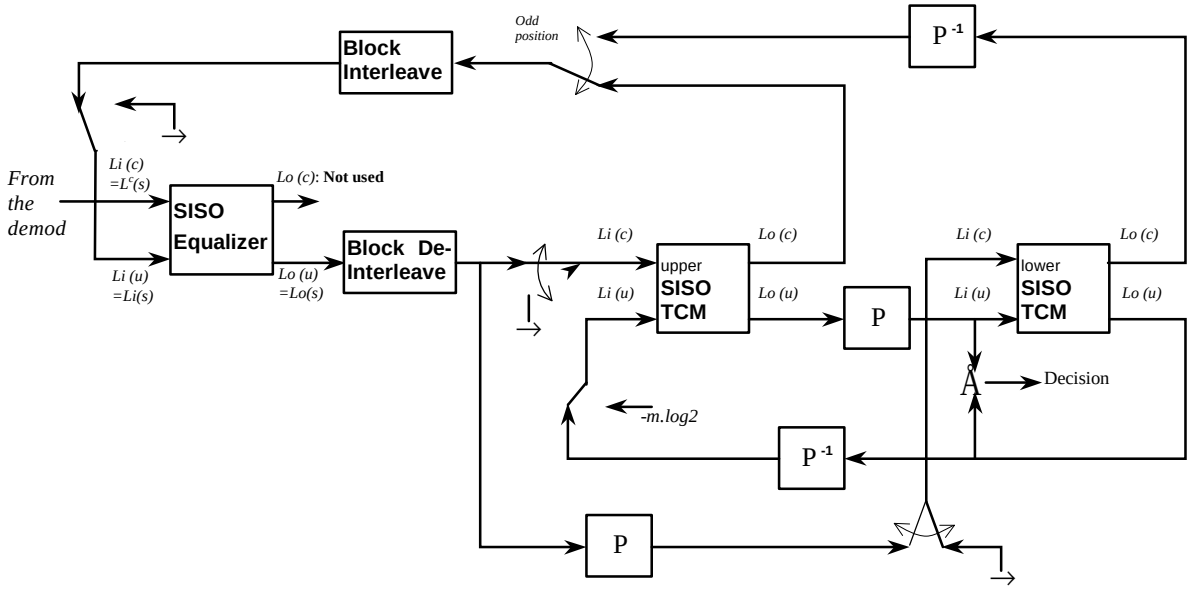


Figure 4 Combined receiver for TTCM