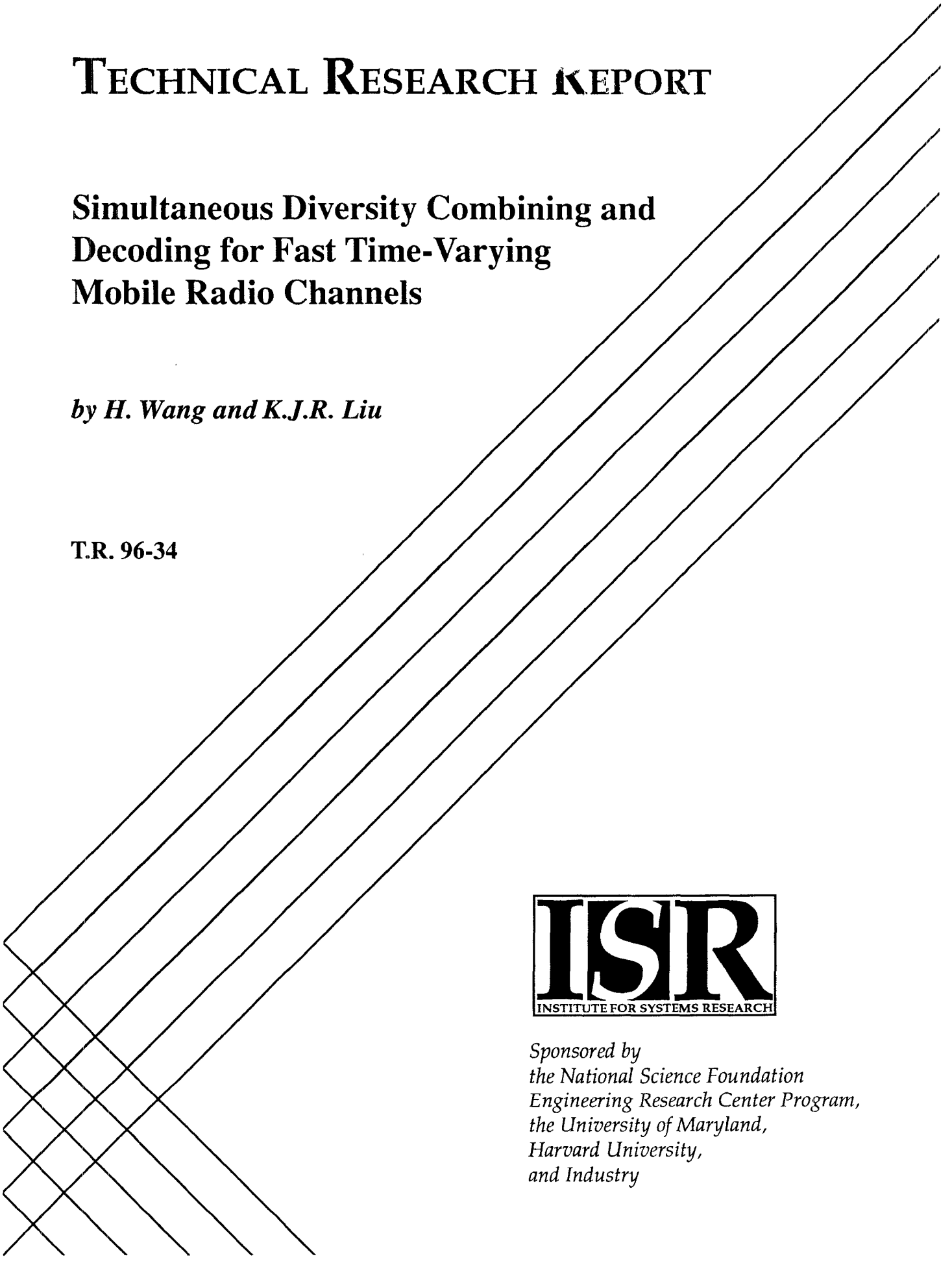
TECHNICAL RESEARCH REPORT

Simultaneous Diversity Combining and Decoding for Fast Time-Varying Mobile Radio Channels

by H. Wang and K.J.R. Liu

T.R. 96-34



Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Simultaneous Diversity Combining and Decoding for Fast Time-Varying Mobile Radio Channels

Hongyi Wang and *K. J. Ray. Liu*

Electrical Engineering Department and Institute for Systems Research
University of Maryland at College Park
College Park, Maryland 20742
Phone: 301 405-6619
Fax: 301 405-6707
hongyi@eng.umd.edu and kjrliu@eng.umd.edu

ABSTRACT

In slowly time-varying mobile radio channels, adaptive diversity combining can reduce multipath fading of desired signal and suppress interfering signals. However, for fast time-varying fading channels, there exist no effective techniques to achieve the same results. The continued use of decision directed adaptive array algorithms will cause error propagation. This paper presents a novel adaptive diversity combining technique with QRD-RLS based parallel weights tracking and a proposed M-D decoder. With moderate increase in complexity, this system significantly reduces error propagation in the decision directed array systems while maintaining the same tracking speed. Its effectiveness and much better performance then that of the conventional technique has been confirmed by computer simulation.

Keywords: diversity combining, fading, TDMA, convolutional code, Viterbi algorithm, RLS, Trellis coded modulation

---

This work was supported in part by the NSF grant MIP9457397 and MIP9309506

I. Introduction

Diversity combining has been widely used in wireless communication. It is a powerful technique for combating multipath flat fading. The most commonly used diversity combining techniques include maximal-radio combining [1], MMSE optimum coherent combining, equal-gain combining [3] [2] and selective combining [3]. Among them, the optimum coherent combining can reduce multipath fading of the desired signal as well as suppress interfering signals [4]. It has attracted a lot of attention recently [4]-[8].

The array weights in the optimum coherent combining are chosen to minimize the error between the reference signal and array output. Unfortunately, the reference signals are not always available. In IS-136 digital cellular standard, we have only 14 symbols at the begining of each time slot that can be used as reference signals. After that, We have to make symbol by symbol decision, and feedback the decided symbol every time to update the array weights. To track fast fading channel, various kinds of RLS can be used. The updating window size used has to be small. If we make a decision error, this error weights heavily in the estimation of the next weights. The next estimated weights are no longer optimal. The diversity combining based upon the erroneous weights will cause the next symbol decision error. This decision error will further propagate and cause subsequent decision errors. Therefore, the effectiveness of optimum coherent array combining on a fast time-varying channel depends on the tracking speed of the adaptive algorithms and on the control of a decision error propagation.

Adaptive optimum diversity combining weights tracking and adaptive channel equalization are two closely related problems. Most of the techniques used for equalization can also be applied to adaptive optimum diversity combining. However, the majority of previous studies on channel equalization or adaptive array combining have concentrated on slowly time-varying fading channels.

In a slowly time-varying fading channel, LMS algorithm is used for most channel equalization. A small change in the equalizer weights is capable of tracking the fading variation. Error propagation is not a severe problem in this situation. Bit Error Rate (BER) performance can be further improved with the combination of an equalizer and a Viterbi decode [9]- [11]. A

tentative decision with small delay or no delay from a Viterbi decoder has been used for channel tracking. In contrary, in a fast time-varying fading channel, a decision delay results in poor tracking performance and a premature tentative decision will cause error propagation.

Also, in a slowly time-varying fading channel, blind equalization techniques such as constant modulus algorithm (CMA) can be used to avoid error propagation in the decision-directed channel equalization [12]. Unfortunately, blind equalization algorithms converge slowly and is not capable of tracking fast time-varying fading channel. Although various improved schemes of CMA have been investigated [13], most of them involve significant increases in complexity or computational costs, and the convergence rates are still lower than that can be achieved by RLS algorithms.

Recently, respective-states channel estimation (RCE) [14][15] was proposed and better performance over conventional decision-directed channel equalization was reported on fast time-varying fading channels. Nevertheless, the continued use of the Viterbi algorithm (VA) in RCE tends to introduce error when there are strong cochannel interferences. Moreover, the complexity of this approach is high. This method has not yet been studied on adaptive array systems.

To effectively perform diversity combining on a fast time-varying fading channel, we developed an adaptive diversity combining system using a M-D decoder. The array weights are tracked by using QRD-RLS algorithm along each of M surviving paths selected by using an M-D decoding algorithm. M and D are to be selected according to RLS updating window length and interference to signal ratio. This system significantly reduces error propagation in the decision directed array system while maintaining the same tracking speed. Our technique is first developed for the convolutional encoded signals and then extended to trellis-coded modulation (TCM) signals to increase information bit rate.

This paper is organized as follows: In section II, the conventional adaptive diversity combining system is described and decision directed error propagation problem is stated. In section III, the idea of simultaneous diversity combining and decoding is introduced. To realize it, computationally efficient D-symbol delay algorithm and M-D algorithm are then developed. QRD based parallel weights tracking techniques are presented. The performance and the computational

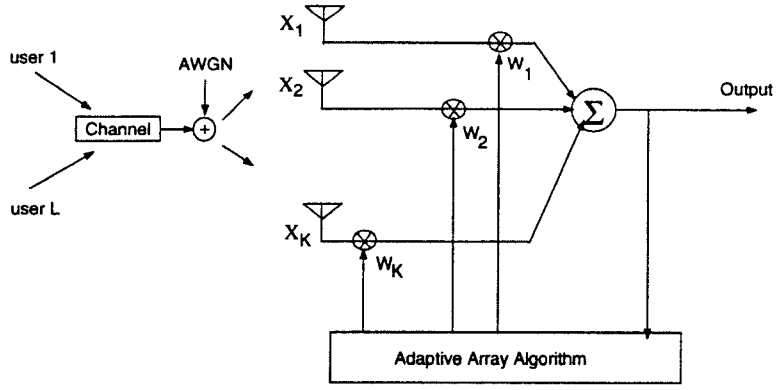


Fig. 1. Block diagram of adaptive array

complexity of the developed adaptive diversity combining system are analyzed. In section VI, the D and M-D algorithms are extended to the TCM signals. In section VI, computer simulation results are provided to demonstrate the significantly improved performance of our techniques. Section VII concludes our work.

II. Conventional Adaptive Diversity Combining System

Fig. 1 is a block diagram of a prototype multi-user communication system with a conventional adaptive diversity combining receiver. There are L users at the same time. These L users are assigned with different training sequences. The communication channel is a flat fading channel with additive white Gaussian noise. The antenna array has K elements. The received array signal vector $\mathbf{r}(n)$ consists of a desired signal vector $\mathbf{A}_s(n)s(n)$, $(L-1)$ interfering signal vectors $\mathbf{A}_{I_j}(n)I_j(n), j = 1, \cdots, (L-1)$ and an additive Gaussian noise vector $\mathbf{n}$, i.e.,

$$\mathbf{r}(n) = \mathbf{A}_s(n) \cdot s(n) + \sum_{j=1}^{L-1} \mathbf{A}_{I_j}(n)I_j(n) + \mathbf{n}(n). \tag{1}$$

where $\mathbf{A}_i, i = s, I_j, j = 1, \cdot L-1$ is an array vector measuring the amplitude and phase distortion of the ith signal caused by fast flat fading at each antenna, $s(n)$ is a desired signal and $I_j(n)$ is the jth interfering signal. The Minimum Mean Square Error (MMSE) optimum diversity combining chooses a $\mathbf{w}$ to minimize $E|s(n) - \mathbf{w}^H(n)\mathbf{r}(n)|^2$. The resolved $\mathbf{w}$ satisfies the following normal equation:

$$\mathbf{R}_{xx}\mathbf{w}_{opt} = \mathbf{r}_{xd} \tag{2}$$

where $\mathbf{R}_{xx} = E[\mathbf{r}(n)\mathbf{r}^H(n)]$ and $\mathbf{r}_{xd} = E[\mathbf{r}(n)s^*(n)]$. With this optimal combining, co-channel interference can be suppressed and $M-(L-1)$ diversity gain can be achieved for each user [7].

If the channel is also frequency selective, then instead of using diversity combining, we need to add a tap delay line in each antenna receiver to suppress intersymbol interference. In this case, $\mathbf{w}(n)$ should be replaced by a weighting matrix. In this paper, we consider only flat fading channels, but most of the results can be extended to the frequency selective fading channels.

In practice, it is impossible to calculate MSE exactly. The method of recursive least square is used instead. The array weighting vector which is used to minimize the cost function that consists of the sum of error squares,

$$\mathcal{E}(n) = \sum_{i=n_0}^{n} \lambda^{n-i} \mid e(i) \mid^2, \tag{3}$$

satisfies the following equation:

$$\hat{\mathbf{R}}_{xx}\hat{\mathbf{w}}(n) = \hat{\mathbf{r}}_{xd}. \tag{4}$$

If a sliding window is employed as the data weighting function, then $\lambda = 1$, $\hat{\mathbf{R}}_{xx} = \sum_{i=n_0}^{n} \mathbf{x}(i)\mathbf{x}^H(i)$ and $\hat{\mathbf{r}}_{xd} = \sum_{i=n_0}^{n} \mathbf{x}(i)d^*(i)$. If an exponential window is employed, then $0 < \lambda < 1$, $n_0 = 1$ $\hat{\mathbf{R}}_{xx} = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)$ and $\hat{\mathbf{r}}_{xd} = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)d^*(i)$. At the begining of each time slot, training sequences are available and serve as the desired signal d. After the training sequence, conventionally, the decided symbol at time n is fed back and serves as $\hat{d}(n)$ to update $\hat{\mathbf{r}}_{xd}(n-1)$. The data received at time n are used to update $\hat{\mathbf{R}}_{xx}$. An updated $\mathbf{w}(n)$ satisfies

$$\hat{\mathbf{R}}_{xx}(n)\hat{\mathbf{w}}(n) = \hat{\mathbf{r}}_{xd}(n). \tag{5}$$

This $\mathbf{w}(n)$ is used to get the next estimated symbol $\hat{d}(n+1)$ as shown in the following equation.

$$\hat{\mathbf{w}}(n)^H\mathbf{x}(n+1) \rightarrow \hat{\mathbf{d}}(n+1). \tag{6}$$

However, when there is a decision error, this error feeds back to the estimation of the next set of weights and will cause a wrong decision. The further propagation of errors eventually causes failure of the subsequent decisions.

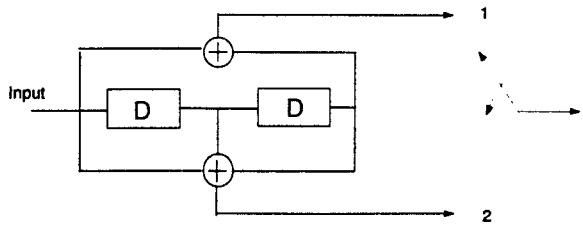


Fig. 2. (2,1,2) convolutional encoder

III. Simultaneous Weights Tracking and Decoding

We have shown in the previous sections that a premature symbol by symbol decision is not reliable and will cause error propagation. One solution is to adopt blind equalization technique, which avoids the use of reference signals, however, its convergence rate is too slow. The other solution is to simply combine array weights updating with convolutional decoding, and feed back more reliable delayed decision from a convolutional decoder. However, a decision delay may cause poor channel tracking.

To reduce error propagation, and to make a more reliable delayed decision but without lossing weights tracking speed, we propose the following simultaneous array weights tracking and decoding technique.

A. More reliable decision based on D symbols

We use a convolutional code shown in Fig. 2 to encode the transmitted information bit and use QPSK to modulate the transmitted signals. At each state, two possible QPSK symbols might be transmitted based upon a "0" or a "1" input. Instead of making an immediate bit decision based upon one symbol (the procedure of making symbol by symbol decision for the convolutional code is provided in Appendix A, we make a more reliable symbol decision based upon D symbols.

We elucidate our thinking using the example shown in Fig. 3. In Fig. 3(a), $D = 5$. At each symbol interval, all the possible sequences in the next D stages are saved. On each path, we perform the following operations:

- Update diversity weights using each path's own reference signals, i.e. its own path outputs.

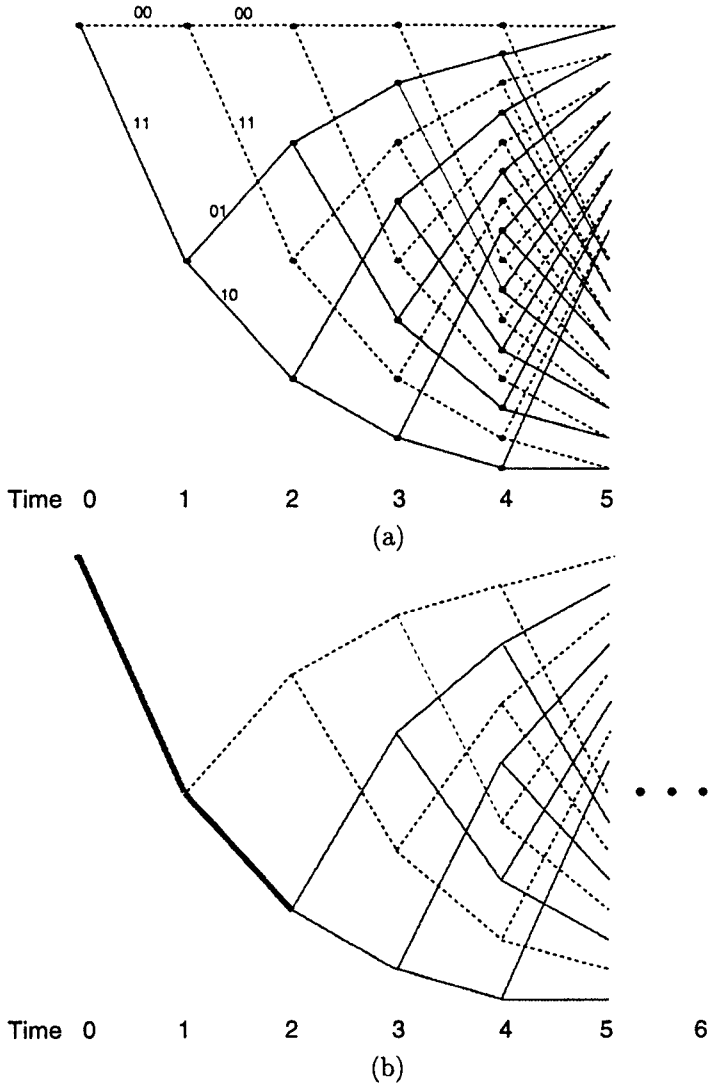


Fig. 3. An example of delayed bit decision based on 5 symbols

- Calculate the Euclidean distance between array output and path output for each branch as follows:

$$b_i(n) = [\mathbf{w}_i^H(n-1)\mathbf{x}(n) - s_i(n)]^2 \tag{7}$$

where $\mathbf{w}_i(n-1)$ is a previously updated weighting vector, $\mathbf{x}(n)$ is a currently received array data and $s_i(n)$ is the reference signal which is the path output on each branch.

- Calculate and save the accumulated Euclidean distance which is evaluated by $\sum_{n=1}^{D} b_i(n)$

along its own path.

In Fig. 3 (a), all the solid line originate from a "1" input bit at stage "0" and all the dashed line originate from a "0" input bit. If at stage 0, the transmitted information bit is "1" which corresponds to path output "11", then a correct path must be a solid line path. After D symbol interval, for one half of the solid paths, weights are updated based upon one correct reference signals and $D-1$ wrong reference signals. One fourth of the solid line paths are updated based upon two correct reference signals and $D-2$ wrong reference signals and so on. On the other hand, on each dashed line path, weights are updated erroneously along D branches and will result in a large accumulated path metric. The smallest accumulated path metric of all the solid line paths should be smaller than that of all the dashed line paths. We then decide the input bit back by D stages. That input bit should be the bit that leads to the selected smallest accumulated path metric.

Once the input bit back by D stages is chosen, we save all the branches originating from it and discard the rest. In Fig. 3 (a), we discard all the dashed line paths. At the next stage, we repeat this process to choose the bit at stage 1, which is shown in Fig. 3 (b).

Along the correct path, weights are updated symbol by symbol. There is no loss of tracking. Moreover, there is no error propagation. We use the correct path's own path output as the reference signal. There is no need to feedback any decided symbol. This way, we make a more reliable symbol decision based on D symbols. Therefore, we reduce error propagation while keeping the same tracking speed.

B. D-symbol Delay Algorithm

We introduced our idea on reducing error propagation based upon convolutional code and accumulated path metrics along $D+1$ symbols.

In the following, we will present an efficient D-delay algorithm to perform the simultaneous weights tracking and decoding.

In Fig 3 (a), at stage 5, two paths merge into each state before we make decision. After the decision, we discard one path in every pair. We will see in the following that each state only need to remember one $\mathbf{w}$ and one accumulated path metric. It then contains all the information

on the past D symbols. Each time, we only need to update the information contained in the current states as shown in Fig. 4.

The decoding algorithm relies on the following key properties of a trellis diagram with 2^D states.

- **Property 1:** The states in the trellis are in the order from 0 to $2^D - 1$. The state of the encoder is set to 0 at the begining of each frame.

  An example is shown in Fig. 5 where $D = 4$.
- **Property 2:** The ith bit of each state corresponds to the input bit i stages back.

  For example, in Fig. 5, at stage $N - 1$, all the states with "0" as the last bit are from a "0" input 4 stages back at stage $N - 5$.

  From Fig. 5, we observe that a state $X_1X_2 \cdots X_D$ can only be reached from the previous state that is either $X_2 \cdots X_{D-1}0$ or $X_2 \cdots X_{D-1}1$. This is demonstrated more clearly in Fig. 6. From this point on, we will refer to the path from state $X_2 \cdots X_{D-1}0$ to $X_1X_2 \cdots X_D$ as an upper path and the path from the state $X_2 \cdots X_{D-1}1$ to $X_1X_2 \cdots X_{D-1}X_D$ as a lower path. State $X_2 \cdots X_{D-1}0$ is always above state $X_2 \cdots X_{D-1}1$. Therefore, the upper path that merges to $X_1X_2 \cdots X_D$ must be from $X_2 \cdots X_{D-1}0$ and the lower path must be from $X_2 \cdots X_{D-1}1$. According to Property 2, $X_2 \cdots X_{D-1}0$ is from a "0" input bit D stages back and $X_2 \cdots X_{D-1}1$ is from a "1" input bit D stages back. This leads to property 3.
- **Property 3:** Every upper path of a pair that merges to each state corresponds to a "0" input bit $D+1$ stages back. Every lower path of a pair that merges to each state corresponds to a "1" input bit $D + 1$ stages back.

Based on these properties, we developed the following D-symbol delay algorithm.

- At each stage, calculate the next accumulated path metrics of all the paths that are generated from all the current states. Compare their accumulated path metrics. If the path with the smallest accumulated path metric is an upper path, then every upper path at each state is kept and every lower path is discarded. The information bit D stages back is decided to be 0. Otherwise, all the lower paths are kept and all the upper path are discarded, the information bit D stages back is decided to be 1.

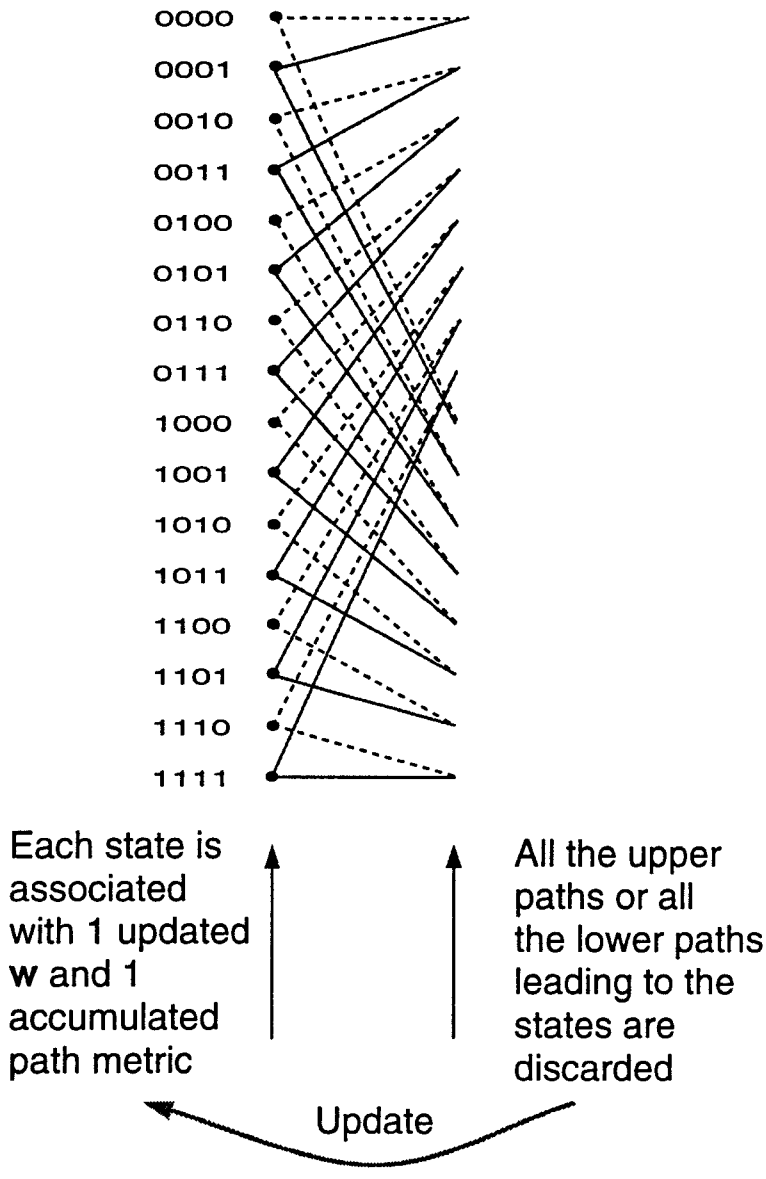


Fig. 4. D-delay algorithm

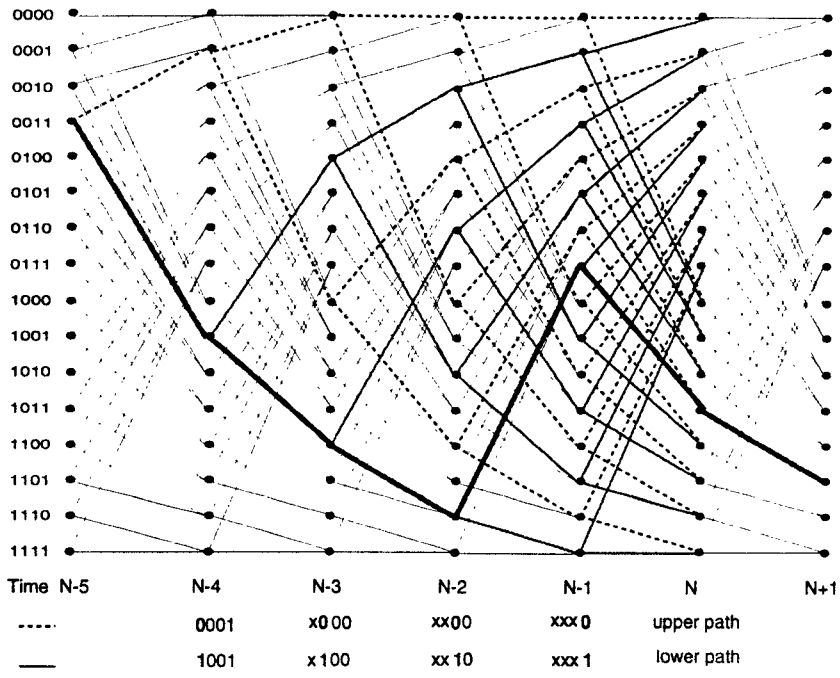


Fig. 5. 4-symbol delay decoding trellis diagram

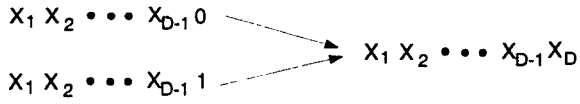


Fig. 6. States change relationship between consecutive stages

- At the current stage, a set of array weights $\mathbf{w}_s(n)$ at each state s, $s = 1, \cdots, 2^D$, are updated by using $\mathbf{w}_{pre-s}(n-1)$, which is a set of array weights obtained at the state prior to "s" along the surviving branch, and by using array data $\mathbf{x}(n)$ and the modulated output signal of the path from state "$pre - s$" to "s".

  In Fig. 5 at stage $N-1$, the path from state 0111 to state 1011 has the smallest accumulated path metric. All the lower paths, i.e. all the solid line paths, are saved as the surviving paths. The information bit 4 stages back is 1 which is the input bit that leads to the path from state 0011 at time $N-5$ to state 1001 at time $N-4$. We then update the array weights along the surviving paths. For example, $\mathbf{w}_{1011}(n)$ is updated based on $\mathbf{w}_{0111}(n-1)$, the QPSK

modulated signal $(\frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2})$ of path output 00 and received array data. Finally, we update the information at the current states with new updated array weights and accumulated path metrics of the surviving paths. The whole updating and symbol decision process is completed by only using the information at current time slot as shown in Fig. 4.

- At the end of each time slot, The last D input bits are decided based upon the final state of the selected path. The last ith bit is equal to the first ith bit of the selected state. In Fig. 5 the last state is 1101, the last four input bits are 1011.

C. Performance Analysis

In this section, we will prove that the reliability of bit decision increases with an increase of D in the D-symbol delay algorithm when there is no strong cochannel interference. But when there is strong cochannel interference, D has to be appropriately chosen to achieve an optimal BER.

In general, the branch metrics measured by Eq.(7) can be divided into three categories:

- Case 1: $\mathbf{w}$ is updated using n desired symbols, and $s_i(n)$ is a desired symbol. The expected value of $b_i(n)$ is the mean square error of the estimated array output $\mathbf{w}_i^H(n-1)\mathbf{x}$. We denote it as mse. $\frac{b_i(n)}{mse} \sim \chi_1^2$. The mse is determined by the adaptive algorithm used, such as RLS or LMS and also determined by the channel conditions such as SNR, fading rate and cochannel interference to signal ratio.
- Case 2: $\mathbf{w}$ is updated using n desired symbols, $s_i(n)$ is an undesired symbol, $\frac{b_i(n)}{mse}$ is from noncentral χ_1^2 with 1 degree of freedom and noncentral parameter d_1, where d_1 is the Euclidean distance between two output signals from two branches that stem from the same node.
- Case 3: $\mathbf{w}$ is updated using $n-m$ desired symbols and m undesired symbols. $s_i(n)$ is an undesired symbol, The expected value of $b_i(n)$ is large in this case and is denoted by MSE. If there is no strong interference in the received array signals, along a wrong path the path outputs are randomly connected with respect to the received array data, and the updated array weights are getting more and more divergent. As a result, we have

$$MSE_D \geq MSE_{D-1} \geq MSE_{D-2} \geq \cdots \geq MSE_1 > mse. \tag{8}$$

Accordingly, we evaluated the path metrics $P_i(n), i = 1, \cdots 2^{D+1}$ by $P_i(n) = \sum_{n=1}^{D+1} b_i(n), i = 1, \cdots 2^{D+1}$, there are three classes of path metrics:

- Class 1: P_c is a path metric of a correct path.
- Class 2: The first m branches belong to Case 1, the $(m+1)$th branch belongs to case 2, the last $D-m$ branches belong to Case 3, $m = 1, \cdots, D$
- Class 3: The first branch metric belongs to Case 2, the rest branches belong to Case 3.

In a D-delay decoder, we make a choice between two subsets of equal size based on the smallest P_i at each stage. Note that all the paths in the undesired subset belong to Class 3, and the paths in the desired subset belong to either Class 1 or Class 2. Obviously, the average difference between a P_u in the undesired subset and the P_d corresponding to the correct path will decrease with an increase of D. The average difference between a P_u and a P_i of a false path in the desired subset is

$$E(P_u - P_i) = \sum_{i=D-m+1}^{D} MSE_i - m \cdot mse \quad m = 1, 2, \cdots, D, \tag{9}$$

The overall average difference E_D between a path in Class 3 and a path in Class 2 for a D-symbol fixed-delay decoder is given by

$$\begin{aligned}
E_D &= \overline{E(P_u - P_i)} && (10)\\
&= \frac{1}{2^D - 1} \sum_{m=1}^{D} 2^{D-m} \Big(\sum_{i=D-m+1}^{D} MSE_i - m \cdot mse \Big) && (11)\\
&= \frac{1}{2^D - 1} \Big[\sum_{m=1}^{D-1} 2^{D-m} \Big(\sum_{i=D-m+1}^{D} MSE_i - m \cdot mse \Big) + \Big(\sum_{i=1}^{D} MSE_i - D \cdot mse \Big) \Big] && (12)\\
&= \frac{1}{2^D - 1} \Big[2 \cdot \sum_{m=1}^{D-1} 2^{D-1-m} \Big(\sum_{i=D-m+1}^{D} MSE_i - m \cdot mse \Big) + \Big(\sum_{i=1}^{D} MSE_i - D \cdot mse \Big) \Big] && (13)\\
&\geq \frac{1}{2^D - 1} \Big[2 \cdot \sum_{m=1}^{D-1} 2^{D-1-m} \Big(\sum_{i=D-1-m+1}^{D-1} MSE_i - m \cdot mse \Big) + \Big(\sum_{i=1}^{D} MSE_i - D \cdot mse \Big) \Big] && (14)\\
&= \frac{1}{2^D - 1} \Big[2 \cdot (2^{D-1} - 1) E_{D-1} + \Big(\sum_{i=1}^{D} MSE_i - D \cdot mse \Big) \Big] && (15)\\
&= E_{D-1} + \Big[\Big(\sum_{i=1}^{D} MSE_i - D \cdot mse \Big) - E_{D-1} \Big] && (16)\\
&> E_{D-1} && (17)
\end{aligned}$$

The bigger of the decision delay, the larger the average Euclidean distance between the correct path and a false path in the other subset and also the larger the average Euclidean distance between paths in the two different subsets. Therefore, the BER decrease with the increase of the decoding decision delay. However, if the output symbols of one of the 2^D surviving paths happen to be the same as the transmitted symbols from the interference, the further increase of D will cause the weights to converge along the interference path and will result in an increase of BER. The relationship in Eq.(8) no longer holds. MSE_i increases initially and then decreases. When D is large enough, MSE_i finally converges to a mean square error between (i) the combined array outputs using the weights that are trained by the interference signals and (ii) the interference signals. We denote this mean square error by mse_I. When the Ith interference is stronger than the desired signal, we have $mse_I < mse$. Thus P_I may become smaller than P_d when D is large enough. Our simulation results also showed that the increase of D will decrease the BER when the interference signals are not as strong as the desired signal. When an interference signal is stronger than the desired signal, BER decreased as we increase the decision delay from 1 to 4 symbols. The improvement becomes smaller and smaller. As we further increase the decision delay, we observe a slight increase of BER. Therefore, a proper delay length needs to be decided based on I/S ratio and RLS updating window size to minimize BER.

D. M-D algorithm

The use of the D-symbol delay algorithm reduces the error propagation in the conventional decision directed array weights tracking algorithm. However, the complexity would increase drastically if we uses D-symbol delay algorithm, because now we have to update 2^D weighting vector instead of 1 weighting vector.

To reduce complexity, we exploit the fact that the correct path should have much smaller accumulated path metric than most of the other selected paths. So, we should be able to discard a majority of the selected paths and keep those most likely ones without compromising the performance. This results in the following M-D algorithm.

Based on the three properties we discussed in the previous section, we find that if all the binary representation of the surviving states have a common last bit, they must be from a unique state

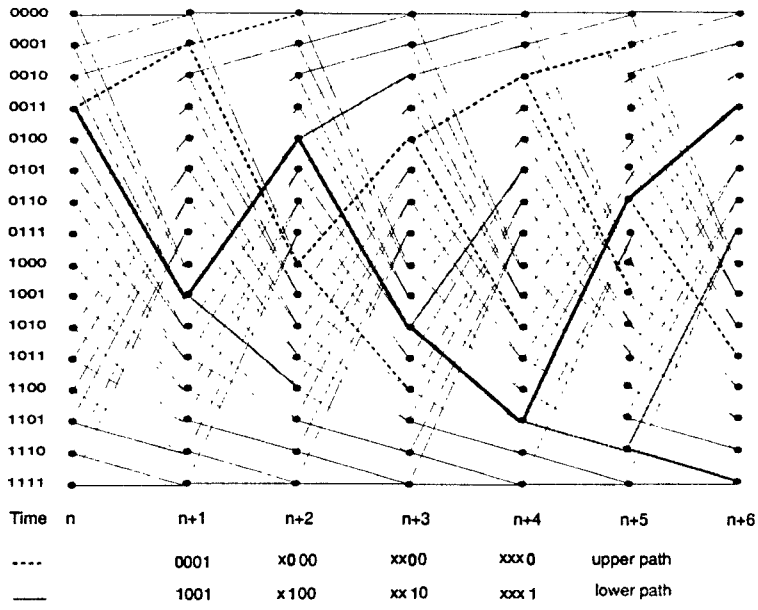


Fig. 7. An example of M-D decoding trellis diagram, $M = 2, D = 4$

D stages back, and the input bit to that state is equal to this last bit. Otherwise, they are from different states D stages back. From this finding and the D-delay algorithm presented in previous sections, we developed the following M-D algorithm.

- At each stage, calculate the next branch metrics and accumulated path metrics of all the paths that are generated from the surviving states.
- When all the binary form of the surviving states have a common last bit, either "1" or "0", the input bit D stages back is decided to be equal to this common last bit. Select M paths that have the smallest accumulated path metrics from all the surviving paths from the surviving states.
- When all the surviving states have a different last bit, compare all the path metrics from these states. If the smallest path metric is from an upper path, keep all the upper paths and discard all the lower paths. The input bit D stages back is decided to be "0". Vice versa. Select M paths that have the smallest accumulated path metrics among the surviving paths.
- A set of array weights $\mathbf{w}_s(n)$ at each surviving state is updated by using $\mathbf{w}_{pre-s}(n-1)$ obtained at the state prior to 's' along the surviving branch, and by using array data $x(n)$

and the modulated output signal of the path from state "$pre - s$" to "s".

- At the end of each time slot, the last D input bits are decided based on the final state of the selected path. The last ith bit is equal to the first ith bit of the final selected state.

In Fig. 7, the two surviving states 0010 and 1101 at stage $n+4$ have different last bits. They are from two different input bits back by D stages. Using the weights obtained at these two surviving states, and the received array signal, we calculate the branch metrics of the branches generated from these two states using its own path output. We update accumulated path metrics of the four corresponding paths. The path that enters state 0110 at stage $n+5$ has the smallest path metric. It is a lower path. The input bit 4 stages back is decided to be "1" which is the bit that generates the path from state 0011 at stage n to the state 1001 at stage $n+1$. We then discard all the upper paths at stages $n+5$ and select 2 paths from all the lower paths. In this case, both lower paths are selected and kept. We then update the weights of these two surviving paths using its own path output and save the updated weights separately at these two surviving states. At the next stage $n+5$, we compare the last bit of the two surviving states 0110 and 1110. They have a common last bit "0". The input bit 4 stages back is decided to be "0" which is the bit that generates the path from 1001 at stage $n+1$ to the state 0100 at stage $n+2$. We then continue to update the accumulated path metrics for the paths generated from these two surviving states, find the one with the smallest path metric, choose the next two surviving paths, and update the next two weights.

The use of M-D algorithm reduces the number of surviving paths from 2^D to M. Our simulation results showed that an M as small as two results in only a slightly degraded performance while $D = 5$. Although, compared to the decision directed weights tracking algorithm, M-D algorithm still increase the number of updating weighting sets from 1 to M, the complexity does not increase by M times. We will demonstrate this relation in the following sections.

E. Diversity Weights Tracking

To achieve MMSE optimum combining on time varying fading channels, recursive algorithms such as LMS or RLS can be applied. RLS has been known to have fast convergence rate and good tracking capability compared to the low complexity LMS algorithm [16] or blind adaptive

algorithms such as CMA. Among different RLS algorithms, QRD-RLS has much better numerical stability. Moreover, it is also computationally more efficient. We can obtain the branch metrics required in M-D algorithm directly using modified QRD-RLS algorithm without calculating the $\mathbf{w}$ explicitly. QRD-RLS algorithm is thus adopted in our system.

The weights in QRD-RLS algorithm satisfy the following equation

$$\mathbf{R}(n)\mathbf{w}(n) = \mathbf{p}(n) \tag{18}$$

which is another form of solution to minimizing (3). $\mathbf{R}$ is an upper triangular matrix obtained in the following equation, in which the weighted data matrix $\mathbf{\Lambda}^{1/2}(n)\mathbf{A}(n)$ is trianglerized through a unitary matrix $\mathbf{Q}(n)$.

$$\mathbf{Q}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{A}(n) = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix} \tag{19}$$

where $\mathbf{0}$ is a null matrix. $\mathbf{p}(n)$ is a vector defined by

$$\mathbf{p}(n) = \mathbf{F}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{d}(n) \tag{20}$$

and $\mathbf{F}(n)$ consists of the first M rows of $\mathbf{Q}(n)$.

To use the QRD-RLS algorithm for parallel residual error (square root of branch metric) calculation, M sets of array weights $\mathbf{w}_i(n), i = 1, \cdots M$, which are associated with the M surviving paths, must satisfy the following equation:

$$\mathbf{R}(n)[\mathbf{w}_1(n), \mathbf{w}_2(n), \cdots, \mathbf{w}_M(n)] = [\mathbf{p}_1(n), \mathbf{p}_2(n), \cdots, \mathbf{p}_M(n)] \tag{21}$$

To solve the least square problem recursively, we use a sequence of Givens rotations to annihilate all M elements in the new incoming data $\mathbf{x}(n)$ one by one. This procedure is shown as follows:

$$\begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix} = \mathbf{T}(n) \begin{bmatrix} \lambda^{1/2}\mathbf{R}(n-1) \\ \mathbf{0} \\ \mathbf{x}^H(n) \end{bmatrix} \tag{22}$$

where $\mathbf{T}(n)$ is the unitary matrix denoting the combined effect of a sequence of Givens rotations:

$$\mathbf{T}(n) = \mathbf{J}_M(n), \cdots, \mathbf{J}_2(n)\mathbf{J}_1(n) \tag{23}$$

$\mathbf{p}_k(n)$ is updated based upon the output $d_k^H(n)$ of the kth surviving branch at time n and the vector $\mathbf{p}_k(n-1)$ generated at the departing state of the branch.

$$\begin{bmatrix} \mathbf{p}_k(n) \\ \mathbf{v}_k(n) \end{bmatrix} = \mathbf{T}(n) \begin{bmatrix} \lambda^{1/2}\mathbf{p}_k(n-1) \\ \lambda^{1/2}\mathbf{v}_k(n-1) \\ \mathbf{d}_k^H(n) \end{bmatrix}. \tag{24}$$

Without calculating $\mathbf{w}$ explicitly, the residual error (or the branch metric) of the kth path can be obtained through $T(n)$ and the last element of $\mathbf{v}_k(n)$ [17].

Before updating, QRD-RLS algorithm requires the initial QR decomposition of the initial data matrix consisting of the first K received-array signal vectors and also the corresponding initial $\mathbf{p}$ vector obtained based upon the first K training symbols. Note that the array baseband signal $\mathbf{x}(n)$ is a complex signal and so are the reference signals. Generally we need to perform a complex QRD-RLS initialization and updating. Also note that if all the diagonal elements of R are real after initialization, they will remain real during the subsequent updating. In Appendix B, we developed a new exact initialization algorithm for complex QRD-RLS algorithm which guarantees the realization of all the diagonal elements of $\mathbf{R}$. This results in better stability and reduced computational complexity in subsequent updating.

F. Computational Complexity

In the QRD-RLS updating algorithm, the complexity of updating R is $O(K^2)$, The complexity of updating p is $O(K)$. The total complexity of calculating $2 \times M$ branch metrics is $O(K^2 + 2MK)$. While in the decision directed algorithm, in addition to updating R and p, we need to perform back substitution to calculate $\mathbf{w}$ in order to get estimated array output. The complexity of back substitution is $O(K^2)$. Overall, compared to decision directed algorithm, the complexity of updating M paths is only moderately increased when M is less than K.

If we keep M surviving paths, we need to calculate $2M$ branch metrics and compare $2M$ accumulated path metrics.

The cost of getting M surviving paths in this M-D decoder is evaluated in the following. One can find the best path out of $2M$ paths with $2M-1$ comparisons [19]. Let $\bar{C}_t(n)$ denote the

average number of comparisons required to find the tth largest of n elements. Then an algorithm exists [19] for which

$$\bar{C}_t(n) = n + t + f(n) \quad \text{where } \lim_{n\to\infty} f(n)/n = 0. \tag{25}$$

To find the Mth best path out of $2M$ paths, we need $2M + M + f(2M)$ comparisons. Once the metric of the Mth path is known, we can choose the rest $M-1$ best paths with at most $M-1$ comparisons [20]. In total, the cost of finding the M best paths out of $2M$ paths is

$$4M - 1 + f(2M) \quad \text{comparisons/branch released} \tag{26}$$

The complexity is $O(M)$. The memory required is also $O(M)$. Thus, this part adds a small amount of complexity when M is less than K.

G. Proposed Adaptive Diversity Combining System

A block diagram shown in Fig. 8 summarizes our proposed adaptive diversity combining system. Each source is encoded with a binary convolutional code. This system combines our proposed M-D decoder and QRD-RLS algorithm for surviving paths selection and symbol decision. At each symbol interval, M surviving paths are selected. Among them, only one is left after D symbol intervals. Each transmitted symbol is automatically decided after a D symbol delay. This system reduces error propagation, with moderate increase in computational complexity.

IV. M-D DECODING OF A TRELLIS CODED 8-PSK CODE

In some situations such as wireless video transmission, we need to transmit high speed data through some limited frequency bandwidth. A use of 8-PSK signal constellation in conjunction with trellis codes can double the transmitted information bit rate compared to a binary 1/2 convolutional coded 4-PSK signal used in the previous examples. Therefore, we modified the D and M-D decoding algorithm of convolutional code and applied it to TCM [18].

In our example of TCM, we partition the eight-phase signal constellation shown in Fig. 9 into subsets of increasing minimum Euclidean distance. We use the rate of 1/2 convolutional code to encode one information bit while the second information bit is left uncoded. The encoder

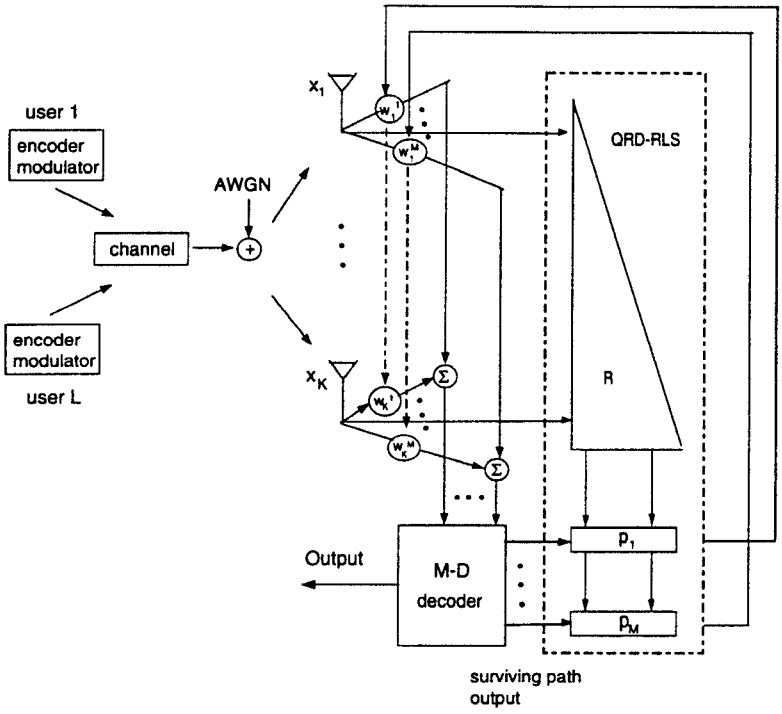


Fig. 8. The proposed adaptive diversity combining system

is shown in Fig 9. The coded bits are used to select one of the four subsets that contain two signal points each, while the uncoded bit is used to select one of the two signal points within each subset. The Euclidean distance between parallel paths is $2\sqrt{\epsilon}$, where ϵ is the energy of the signal.

The decoding algorithm is given as follows:

- Select the branch having the smaller path metric among the parallel branches. If an upper path is selected, the uncoded bit at current time is decided to be "0", otherwise, it is decided to be "1".
- The coded bit is decoded using the D or M-D algorithm presented in the previous sections.

Fig. 10 shows a 2-symbol delay 8-PSK TCM decoder. Each time, one branch is selected between each pair of parallel branches in favor of the one having a smaller branch metric. Then we choose surviving paths among these selected paths following the same procedure that is used in the convolutional decoder having a 2-symbol delay. At time 3, one path is selected from each

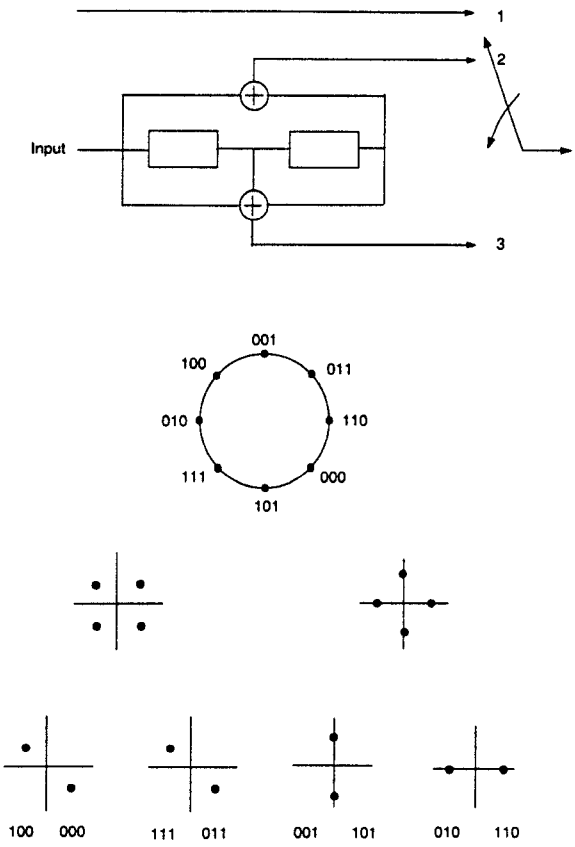


Fig. 9. Set partitioning of an 8-PSK signal set

pair of branches that are generated from surviving states 01 or 11 at stage 2. Within these four selected paths, the path enters state 10 at stage 3 has the smallest accumulated path metric. All the lower paths in the surviving paths are saved, i.e., the paths enter state 00 and state 10 at stage 3.

V. Simulation Results

In our simulations, four antennas are used. The channel is time-division-multiplexed. There are 162 symbols in each time slot. The first 14 symbols are from the training sequence. The carrier frequency is 900 MHz. The modulated data rate is 24.3ksym/s, which is the same as in IS-136 standard. The SNR is 15dB.

Computer simulation results provided in Fig. 12 give a quantitative examination of the BER

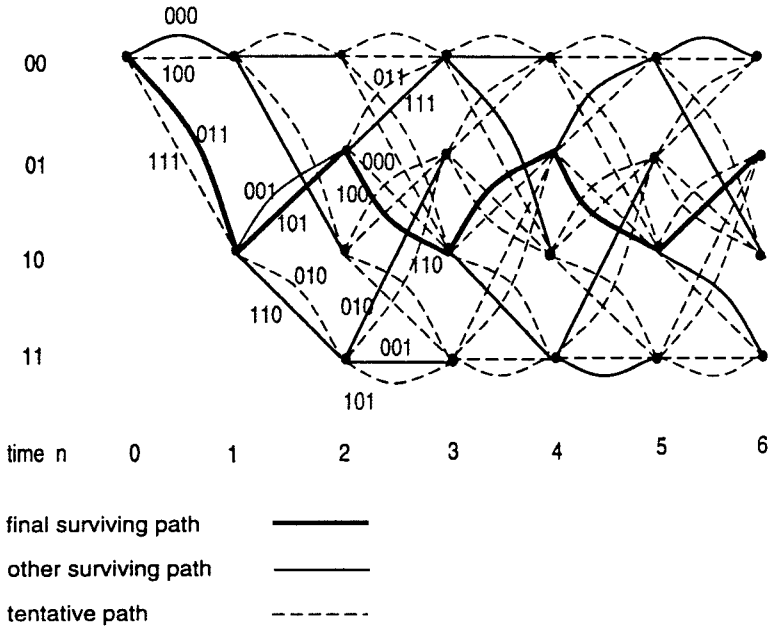


Fig. 10. 2-symbol delay 8-PSK TCM decoding trellis diagram

improvement from using D-symbol delay algorithm and M-D algorithm, respectively. Three sources are all encoded with the convolutional encoder shown in Fig 2. The desired vehicle is moving at 60 miles/hr. The other two are moving at 30miles/hr and have the same interference to signal ratio.

In Fig. 12, we observe improved BER performance as we increase D from 0 to 5: about 4dB improvement in one interference suppression and a total of 8dB improvement in both cases (a) and (b). With M-D algorithm, the improvement is slightly less compared to D algorithm, but the complexity of M-D algorithm is greatly reduced.

In Fig. 13, the M-D algorithm is used and M is set to 2. In Fig. 13(a), at low ISR, BER performance is gradually improved as D is increased. A total of 10dB improvement with $D = 5$ over $D = 0$. At high ISR, BER performance is improved as D is increased from 1 to 4 and is slightly decreased as D is further increased. This is because when the interference is stronger than a desired signal, weights may converge along a wrong surviving path. So D should be appropriately chosen at high ISR to achieve the optimal BER.

An extension of D-delay and M-D algorithms to the TCM code is demonstrated in the following example. A TCM encoder shown in Fig. 9 is used. In Fig. 14, the dotted line represents the BER performance under decision directed QRD-RLS for weights tracking. Coherent demodulation is made on an uncoded QPSK signal sequence. The solid line represent the BER under 1-symbol and 2-symbol delay algorithms. Using 2-symbol delay algorithm, 3 to 5 dB improvement in the interference suppression over the conventional decision directed algorithm is observed.

The BER performance achieved by using M-D algorithm with $M = 2, D = 4$ are compared to that achieved by using Viterbi algorithm [15] with 4 states. Infinite memory length is used in the Viterbi algorithm. Fig. 15 shows that more improvement in BER is achieved by using M-D algorithm at high ISR. The complexity of M-D($M = 2$) algorithm is also lower. This is because the length of two parallel surviving paths in the Viterbi decoding algorithm is not fixed and can not be controled. Therefore, using Viterbi decoding algorithm, we can not avoid the converging of weights along a wrong path at high ISR.

VI. Conclusions

This paper presents a simultaneous diversity weights updating and decoding technique. M-D decoding algorithm is developed for the binary convolutional codes and TCM codes. It provides instantaneously a set of candidate reference signals for weights tracking and makes a final symbol decision with a D symbol delay based on more reliable accumulated path metrics. It thus significantly reduces error propagation in the decision directed array systems while maintaining the same tracking speed.

The memory required by the M-D algorithm is only O(M). The computational complexity required is $O(K^2 + M \cdot K)$ for weights updating and $O(M)$ for decoding, which is not much increased from $O(K^2)$ in the conventional decision directed adaptive array system when M is 2.

Simulation results showed that about 8 to 10dB improvement in total interference suppression at low ISR and about 3 to 5dB improvement in interference suppression at high ISR can be achieved with a moderate increase in complexity.

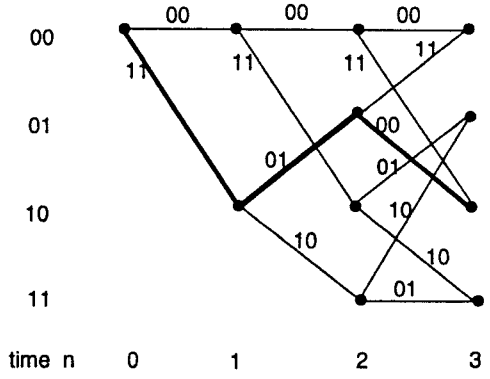


Fig. 11. No delay decoding trellis diagram

VII. Appendix

A. *No delay decoding*

In the no-delay symbol by symbol decision, decision is made based on a four states trellis diagram. In Fig. 11, at stage "0", the next possible transmited baseband signal is either $(-\frac{\sqrt{2}}{2}, -i\frac{\sqrt{2}}{2})$ or $(\frac{\sqrt{2}}{2}, i\frac{\sqrt{2}}{2})$. We compare the path metrics of the two corresponding paths. The path with output "11" has smaller metric, "11" are decided to be the transmitted bits, and $(\frac{\sqrt{2}}{2}, i\frac{\sqrt{2}}{2})$ is fed back to update the array weights. At stage "1", the next possible modulated signal is $(-\frac{\sqrt{2}}{2}, i\frac{\sqrt{2}}{2})$ or $(\frac{\sqrt{2}}{2}, -i\frac{\sqrt{2}}{2})$. The path ending at state "01" at stage "2" has the smaller accumulated path metric, and $(-\frac{\sqrt{2}}{2}, i\frac{\sqrt{2}}{2})$ is selected and fed back for weights updating. Then the next two possible paths are compared, one of them is selected. This process is repeated for all stages in the trellis. The effectiveness of this approach is equivalent to a conventional decision directed weights tracking for a BPSK signals. The error propagation cannot be avoided.

B. *Exact Initialization of the complex QRD-RLS Algorithm*

We begin the time recursions at $n = 1$. We multiply a complex number which takes the complex conjugate of the first element of data vector $\mathbf{x}(1)$ and normalize it to 1. This number can be expressed as $e^{-j\theta_1}$ where $\theta_1 = arctan[Im(x_1(1))/Re(x_1(1))]$. At $n = 2$, we append $\mathbf{x}(2)$ to form a matrix with two rows. We apply a 2×2 Givens rotation, denoted by $\mathbf{J}_1(2)$, to eliminate

the first element of the row vector.

$$\mathbf{J}_1(2)\cdot\begin{bmatrix} \sqrt{\lambda}e^{-j\theta_1}x_1(1) & \sqrt{\lambda}e^{-j\theta_1}x_2(1) & \cdots & \sqrt{\lambda}e^{-j\theta_1}x_K(1) \\ x_1(2) & x_2(2) & \cdots & x_K(2) \end{bmatrix} = \begin{bmatrix} r_{11}^{(2)}(1) & r_{12}^{(2)}(1) & \cdots & r_{1K}^{(2)}(1) \\ 0 & x_2^{(1)}(2) & \cdots & x_K^{(1)}(2) \end{bmatrix} \tag{27}$$

$$\mathbf{J}_1(2) = \begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \tag{28}$$

where

$$c = \frac{\sqrt{\lambda}e^{-j\theta_1}x_1(1)}{\sqrt{(\sqrt{\lambda}e^{-j\theta_1}x_1(1))^2 + |x_1(2)|^2}} \tag{29}$$

and

$$s = \frac{x_1(2)}{\sqrt{(\sqrt{\lambda}e^{-j\theta_1}x_1(1))^2 + |x_1(2)|^2}} \tag{30}$$

We then multiply a unitary matrix $\mathbf{U}(1)$ to make the $x_2^{(1)}(2)$ real.

$$\begin{bmatrix} 1 & \\ & e^{-j\theta_2} \end{bmatrix}\begin{bmatrix} r_{11}^{(2)} & r_{12}^{(2)} & \cdots & r_{1K}^{(2)} \\ 0 & x_2^{(1)}(2) & \cdots & x_K^{(1)}(2) \end{bmatrix} = \begin{bmatrix} r_{11}^{(2)} & r_{12}^{(2)} & \cdots & r_{1K}^{(2)} \\ 0 & r_{22}^{(2)} & \cdots & r_{2K}^{(2)} \end{bmatrix} \tag{31}$$

where $\theta_2 = arctan[Im(x_2^{(1)}(2))/Re(x_2^{(1)}(2))]$. At $n = 3$, we append vector $\mathbf{x}(3)$ to the matrix at the right side of Eq.(31) to form a three-row matrix. We eliminate the first two elements of $\mathbf{x}(3)$ using two Givens rotations $J_2(3)$ and $J_1(3)$.

$$J_2(3)J_1(3)\begin{bmatrix} r_{11}^{(2)} & r_{12}^{(2)} & \cdots & r_{1K}^{(2)} \\ 0 & r_{22}^{(2)} & \cdots & r_{2K}^{(2)} \\ x_1(3) & x_2(3) & \cdots & x_K(3) \end{bmatrix} = \begin{bmatrix} r_{11}^{(3)} & r_{12}^{(3)} & r_{13}^{(3)} & \cdots & r_{1K}^{(3)} \\ 0 & r_{22}^{(3)} & r_{23}^{(3)} & \cdots & r_{2K}^{(3)} \\ 0 & 0 & x_3^{(2)}(3) & \cdots & x_K^{(2)}(3) \end{bmatrix} \tag{32}$$

We then multiply a unitary matrix $\mathbf{U}(3)$ to make the $x_3^{(2)}(3)$ real.

$$\begin{bmatrix} 1 & & \\ & 1 & \\ & & e^{-j\theta_3} \end{bmatrix}\begin{bmatrix} r_{11}^{(3)} & r_{12}^{(3)} & r_{13}^{(3)} & \cdots & r_{1K}^{(3)} \\ 0 & r_{22}^{(3)} & r_{23}^{(3)} & \cdots & r_{2K}^{(3)} \\ 0 & 0 & x_3^{(2)}(3) & \cdots & x_K^{(2)}(3) \end{bmatrix} = \begin{bmatrix} r_{11}^{(3)} & r_{12}^{(3)} & r_{13}^{(3)} & \cdots & r_{1K}^{(3)} \\ 0 & r_{22}^{(3)} & r_{23}^{(3)} & \cdots & r_{2K}^{(3)} \\ 0 & 0 & r_{33}^{(3)} & \cdots & r_{3K}^{(2)} \end{bmatrix} \tag{33}$$

By continuing this process, we can construct a triangular matrix at $n = M$ with all diagonal elements real.

The initialization of the QRD-RLS algorithm also involves the initialization of $\mathbf{p}_1, \cdots, \mathbf{p}_M$. At time $n = 1$, we multiply $e^{-j\theta_1}$ to $d_1^*(1), \cdots, d_M^*(1)$ to get initial $p_1(1) \cdots p_M(1)$. At time $n = 2$, we use $\mathbf{J}_1(2)$, $\mathbf{U}(2)$ and reference signals at the second branch to update $p_1(1) \cdots p_M(1)$ and get M 2×1 vectors $\mathbf{p}_1(1) \cdots \mathbf{p}_M(1)$. This process is shown as follows:

$$\mathbf{U}(2) \cdot \mathbf{J}_1(2) \cdot \begin{bmatrix} p_1(1) & p_2(1) & \cdots & p_M(1) \\ d_1^*(2) & d_2^*(2) & \cdots & d_M^*(2) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1(2) & \mathbf{p}_2(2) & \cdots & \mathbf{p}_M(2) \end{bmatrix} \tag{34}$$

Similarly, at time $n = 3$, we perform

$$\mathbf{U}(3) \cdot \mathbf{J}_2(3) \cdot \mathbf{J}_1(3) \begin{bmatrix} \mathbf{p}_1(2) & \mathbf{p}_2(2) & \cdots & \mathbf{p}_M(2) \\ d_1^*(3) & d_2^*(3) & \cdots & d_M^*(3) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1(3) & \mathbf{p}_2(3) & \cdots & \mathbf{p}_M(3) \end{bmatrix} \tag{35}$$

By continuing this process, we construct the initial M $K \times 1$ vectors $\mathbf{p}_1(K), \cdots, \mathbf{p}_M(K)$.

ACKNOWLEDGMENT

The authors would like to thank J. H. Winters, C. Martin and other researchers in the AT&T Bell Labs for many stimulating discussions. The author would also like to thank B. Sampath for helpful discussions on the optimal ML decoding.

REFERENCES

[1] J. Proakis, "Digital Communications," Second Edition, McGraw-Hill, Series, 1989.

[2] A. J. Viterbi, "CDMA Principles of Spread Spectrum Communication" Addison-Wesley Publishing Company 1995

[3] M. D. Yacoub "Foundations of Mobile Radio Engineering" CRC Press, Inc. 1993.

[4] J. H. Winters, J. Salz and R. D. Gitlin, "The Impact of Antenna Diversity on the Capacity of Wireless Communication Systems," *IEEE Trans. on Communications*, vol. 42, No. 2/3/4, pp. 1740-1751, February/March/April 1994.

[5] J. Kennedy, M. C. Sullivan, "Direction Finding and "Smart Antennas" Using Software Radio Architectures *IEEE Communications Magazine*, pp. 62-68, May 1995.

[6] M. V. Clark, L. J. Greenstein, W. K. Kennedy and M. Shafi, "MMSE Diversity Combining for Wide-Band Digital Cellular Radio," *IEEE Trans. on Communications*, vol. 40, No.6, pp. 1128-1135, June 1992.

[7] J. H. Winters, "Signal Acquisition and Tracking with Adaptive Arrays in the Digital Mobile Radio System IS-54 with Flat Fading," *IEEE Trans. on Vehicular Technology*, vol. 42, No. 4, pp. 377-384, November 1993.

[8] P. Jung, "Performance Evaluation of a Novel M-Detector for Coherent Receiver Antenna Diversity in a GSM-Type Mobile Radio System," *IEEE Journal on Selected Areas in Communications*, vol. 13, No.1, pp. 80-88, January 1995.

[9] P. R. Chevillat and E. Eleftheriou, "Decoding of Trellis-Encoded Signals in the Presence of Intersymbol Interference and Noise," *IEEE Trans. on Communications*, vol. 37, No.7, pp. 669-676, July 1989.

[10] K. Zhou, J. G. Proakis and F. Ling, "Decision-Feedback Equalization of Time-Dispersive Channels with Coded Modulation," *IEEE Trans. on Communications*, vol. 38, No.1, pp. 18-24, January 1990.

[11] W. Sheen and G. L. Stuber, "MLSE Equalization and Decoding for Multipath-Fading Channels," *IEEE Trans. on Communications*, vol. 39, No.10, pp. 1455-1464, October 1991.

[12] T. Ohgane, T. Shimura, N. Matsuzawa and H. Sasaoka, "An Implementation of a CMA Adaptive Array for High Speed GMSK Transmission in Mobile Communication," *IEEE Trans. on Vehicular Technology*, vol. 42, No. 3, pp. 282-288, August 1993

[13] T. Schirtzinger, X. Li, and W. K. Jenkins, "A Comparison Of Three Algorithms For Blind Equalization Based On The Constant Modulus Error Criterion," *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Detroit, Michigan, May 9-12, 1995, pp. 1049-1052.

[14] H. Kubo, K. Murakami and T. Fujino, "An Adaptive Maximum-Likelihood Sequence Estimation for Fast Time-Varying Intersymbol Interference Channels" *IEEE Journal on Selected Areas in Communications*, vol. 42, No. 2/3/4, pp. 1872-1880, Feb./March/April 1994.

[15] H. Kubo, K. Murakami and T. Fujino, "Adaptive Maximum-Likelihood Sequence Estimation by Means of Combined Equalization and Decoding in Fading Environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, No. 1, pp. 102-109, January 1995.

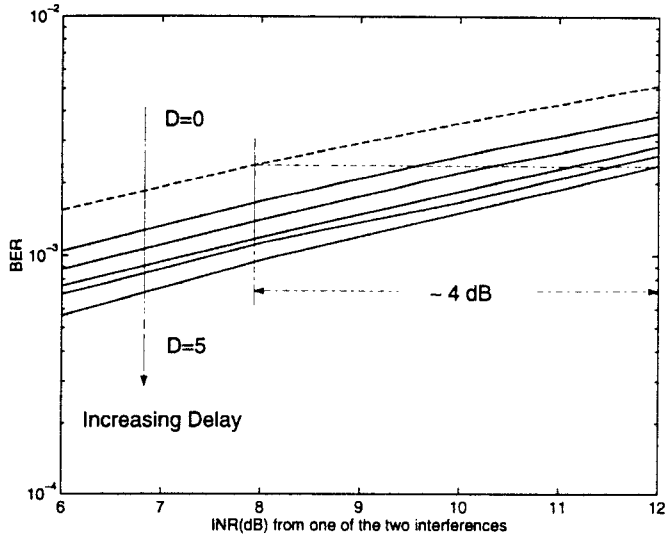
[16] J. Proakis, "Adaptive Equalization for TDMA Digital Mobile Radio," *IEEE Trans. on Vehicular Technology*, vol. 40, No.2, pp. 333-341, May 1991.

[17] S. Haykin, "Adaptive Filter Theory," Second Edition, Prentice Hall Information and System Sciences Series, 1991.

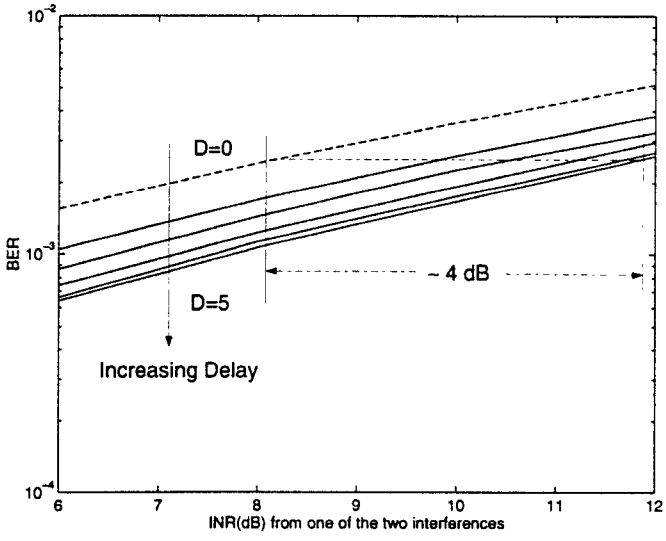
[18] G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction," *IEEE Communication Magazine*, vol. 25, No.2, pp. 5-21, May 1987.

[19] D. E. Knuth, "The Art of Computer Programming," Vol. III: Sorting and Searching. Reading, MA: Addison-Wesley. 1973

[20] J. B. Anderson and S. Mohan, "Sequential Coding Algorithms: A survey and Cost Analysis," *IEEE Trans. on Communications*, vol. 32, No.2, pp. 169-176, Feb. 1984.

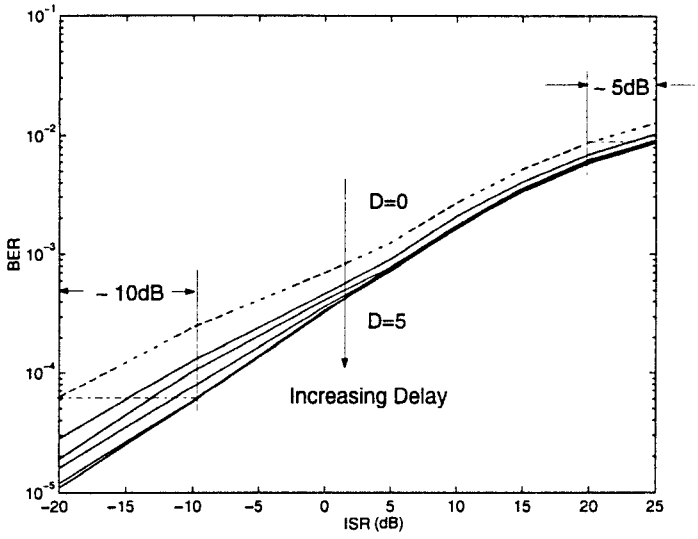


(a) D-delay

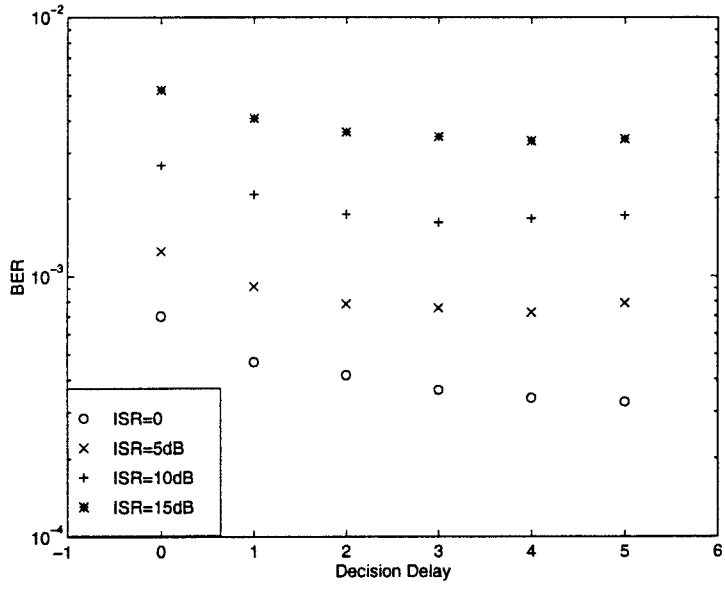


(b) M-D

Fig. 12. Influence of D on the performance of D-delay and M-D decoding algorithms



(a) Influence of D at low ISR and high ISR



(b) Influence of D at high ISR

Fig. 13. More significant improvement at low ISR (M-D algorithm, M=2)

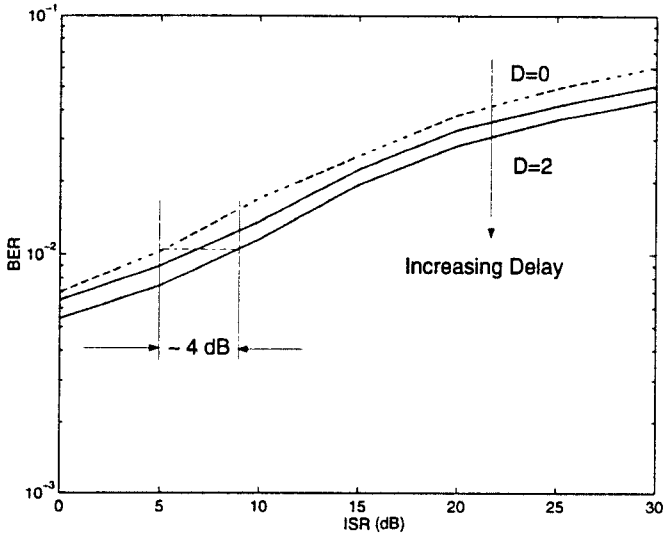


Fig. 14. BER performance of D-symbol delay decoding of a TCM code

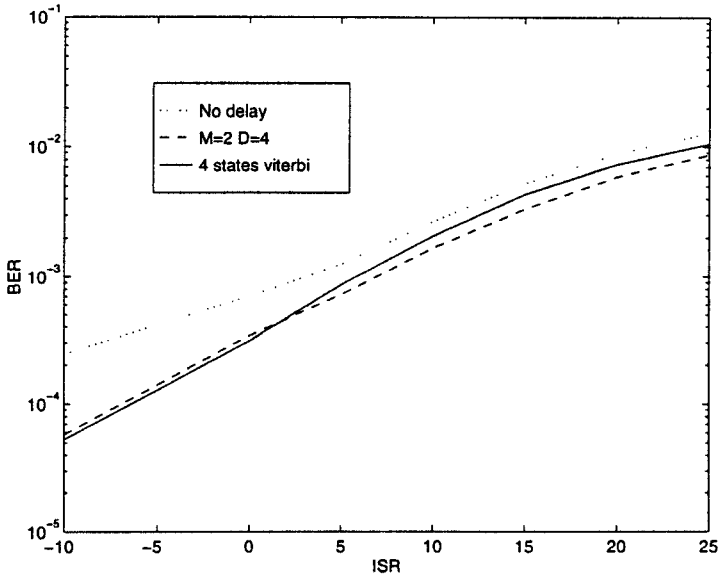


Fig. 15. Comparison between M-D algorithm and Viterbi algorithm