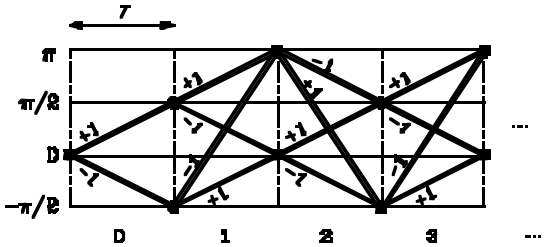


Fig.1. Phase evolution in a Trellis diagram

Where T is a symbol period time. As it can be deduced from the Trellis diagram, all the vectors of an interval (branches) are given by the original phase and its corresponded symbol. Therefore the branches can be characterized by a branch vector, $\underline{S}_{branch}$, that depends on the original phase θ_{branch} and the corresponded symbol α_{branch} :

$$\underline{S}_{br}\left(\theta_{br},\alpha_{br}\right)=e^{j\theta_{br}}\cdot e^{j\frac{\pi}{2}\cdot\alpha_{br}\cdot\underline{q}}=e^{j\theta_{br}}\cdot\begin{Bmatrix}\underline{S}^{(+1)}\\ \underline{S}^{(-1)}\end{Bmatrix} \quad (4)$$

By this way, the signal (in the k interval) $\underline{x}_k$ is reduced to a vector that characterizes a determinate branch (the branch whose associated vector is equal to the transmitted signal) plus the noise:

$$\underline{x}_k=\underline{S}_k\left(\theta_k,\alpha_k\right)+\underline{n}_k \quad (5)$$

Thus, the phase of the received signal will be determinied by a determinate path along the Trellis diagram (and definite by the burst of transmitted symbols $\underline{\alpha}$) affected by the noise. The received signal, in the analog time domain, can be expressed as:

$$x(t)=s(t,\underline{\alpha})+n(t) \quad (6)$$

Assuming that the noise is given by an additive, white Gaussian channel, the optimum detector must minimize the log-likelihood function [4]:

$$\log(\Lambda_N[x(t)])=\int_0^{NT}\left[x(t)-s(t,\hat{\underline{\alpha}})\right]^2 dt \quad (7)$$

where $\hat{\underline{\alpha}}$ is the estimated symbol's vector. The last expression is the Euclidean distance between the received signal and the "signal" definite by the estimated path coursed along the Trellis diagram.

The simplest algorithm that minimizes (7) is called the Maximum Likelihood Algorithm. However, this algorithm does not try minimize (7) operating with the complete estimated burst of symbols $\hat{\underline{\alpha}}$. The ML estimation is acomplished symbol by symbol, without any kind of memory of the past and evaluating all the Euclidean distances with each processed symbol. In order to obtain the Euclidean distance (weight), $\underline{x}_k$ it is subtracted to each of branches vectors of k intervals:

$$W_{branch}=\left\|\underline{x}_k-\underline{S}_{br}\right\|^2 \quad (8)$$

And the symbol associated to the branch whose weight is the minimum, results the estimated symbol.

However, to guarantee the minimum of Euclidean distance for each symbol interval does not imply that it is guarantied this minimum in all the symbols burst, as can be easy deduced. Therefore it is interesting to develop an algorithm that takes into account the past; that in some way take into account the weight along the symbols (branches). Because there are lots of possible paths, it is necessary to leave aside the paths with higher accumulated weight, deciding at the end the burst of symbols that are associated to the path with minimum accumulated weight. This is, briefly, the Viterbi algorithm.

The discrimination of paths with higher weights is carry through by the following method: since two branches arrive to each final phase of the interval (final state) with an accumulated weight (which is determinated by the accumulated weight in the original phase and the associated ones to the branch), the branch (and the path) with higher accumulated weight will be left aside. The other accumulated weight (with its associated path) will be associated to the final state, that will become an initial state in the following interval. An example of this process can be seen in the following figure:

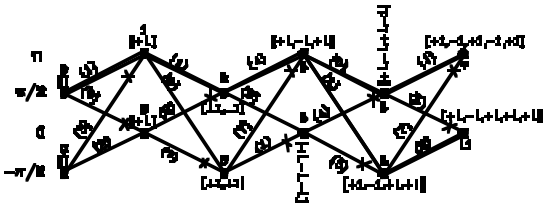


Fig. 2. Weights in the Viterbi Algorithm. Over the states are the accumulated weight and its associated path.

Since the associated path is incremented in each iteration, it is necessary a memory of two times the number of transmitted symbols. Furthermore, we could not decide between the two paths of minimum accumulated weight until the end of the burst of symbols, delaying the decoding until (at least) the end of transmission. In order to avoid this problems, it will be taken into account that the two paths must have the same beginning, differing only in the last symbols; thus, only a few symbols have to be saved, the (i.e.) L symbols immediately anterior and give as decoded symbol the symbol L +1 anterior, that it will be common to the two selected paths. The final result will be sensibly better than the ML criteria ones without increase the computational cost in a symbol interval.

III. ORTHOGONAL DECOMPOSITION

Until now, as well in the ML as in the Viterbi algorithms, the weights of the branches were given by the Euclidean distance between the branch and the received signal in the symbol interval (8), which implies to

operate with Ns x 1 dimensional vectors. It will be of great interest to try to reduce the vectors' dimension. Our question is how to obtain shorter vectors.

The transmitted signal ($\underline{S}^{(+1)}$ or $\underline{S}^{(-1)}$) can be decomposed (Orthogonal Decomposition, OD) into a subspace (that will be called "signal subspace"), this is, to present it in a lineal combination of orthonormal base vectors:

$$\begin{aligned}\underline{S}^{(+1)} &= a_{11}\cdot\underline{V}_1 + a_{12}\cdot\underline{V}_2 = \begin{pmatrix}\underline{V}_1 & \underline{V}_2\end{pmatrix}\cdot\underline{a}^{(+1)}\\ \underline{S}^{(-1)} &= a_{21}\cdot\underline{V}_1 + a_{22}\cdot\underline{V}_2 = \begin{pmatrix}\underline{V}_1 & \underline{V}_2\end{pmatrix}\cdot\underline{a}^{(-1)}\end{aligned} \tag{9}$$

with

$$\underline{a}^{(+1)} = \begin{pmatrix}a_{11}\\ a_{12}\end{pmatrix} \quad \text{and} \quad \underline{a}^{(-1)} = \begin{pmatrix}a_{21}\\ a_{22}\end{pmatrix} \tag{10}$$

where $\underline{V}_1$ and $\underline{V}_2$ are the orthonormal base vectors; and since they are, it can be obtained easily, in a symbol interval, the $\underline{a}_k$ components multiplying $\underline{V}_1$ and $\underline{V}_2$ by $\underline{S}_k$. For example, if +1 has been transmitted, it will be obtained $\underline{a}^{(+1)}$:

$$\begin{aligned}\underline{V}_1^H\cdot\underline{S}^{(+1)} &= \underline{V}_1^H\cdot\begin{pmatrix}\underline{V}_1 & \underline{V}_2\end{pmatrix}\cdot\underline{a}^{(+1)} = \begin{pmatrix}1 & 0\end{pmatrix}\cdot\underline{a}^{(+1)}\\ &= a_{11}\\ \underline{V}_2^H\cdot\underline{S}^{(+1)} &= \underline{V}_2^H\cdot\begin{pmatrix}\underline{V}_1 & \underline{V}_2\end{pmatrix}\cdot\underline{a}^{(+1)} = \begin{pmatrix}0 & 1\end{pmatrix}\cdot\underline{a}^{(+1)}\\ &= a_{12}\end{aligned} \tag{11}$$

Components of $\underline{a}^{(-1)}$ will be obtained analogously.

In order to obtain the components $\underline{a}^{(+1)}$ and $\underline{a}^{(-1)}$ it is necessary to know the vectors $\underline{V}_1$ and $\underline{V}_2$. This vectors are obtained applying a Singular Value Decomposition (SVD) to the covariance matrix C_s generated by $\underline{S}^{(+1)}$ and $\underline{S}^{(-1)}$:

$$C\_s = \frac{1}{2}\left(\underline{S}^{(+1)}\cdot\left(\underline{S}^{(+1)}\right)^H + \underline{S}^{(-1)}\cdot\left(\underline{S}^{(-1)}\right)^H\right) =$$

$$= \begin{pmatrix}\underline{V}_1 & \underline{V}_2 \cdots \underline{V}_{Ns}\end{pmatrix}\begin{pmatrix}\lambda_1 & & \underline{0}\\ & \lambda_2 & \\ & \ddots & \\ \underline{0} & & \lambda_{Ns}\end{pmatrix}\begin{pmatrix}\underline{V}_1 & \underline{V}_2 \cdots \underline{V}_{Ns}\end{pmatrix}^H \tag{12}$$

where $\underline{V}_i$ are real eigenvectors and λ_i are their associated eigenvalues (real numbers and major to minor ordered). Since C_s is generated by two linearly independent vectors, C_s is range-2 and $\lambda_3, \lambda_4, \ldots, \lambda_{Ns}$ will be nulls. Thus, the base vectors $\underline{V}_1$ and $\underline{V}_2$ will be the eigenvectors associated to the no nulls eigenvalues.

Once deduced the base vectors $\underline{V}_1$ and $\underline{V}_2$, we can obtain the components of the received signal. Therefore, for each interval:

$$\begin{aligned}\underline{V}_1^H\cdot\underline{x}_k &= \underline{V}_1^H\cdot\left(e^{j\theta_k}\cdot\begin{Bmatrix}\underline{S}^{(+1)}\\ \underline{S}^{(-1)}\end{Bmatrix} + \underline{n}_k\right) =\\ &= e^{j\theta_k}\begin{Bmatrix}a_{11}\\ a_{21}\end{Bmatrix} + \underline{V}_1^H\cdot\underline{n}_k\\ \underline{V}_2^H\cdot\underline{x}_k &= \underline{V}_2^H\cdot\left(e^{j\theta_k}\cdot\begin{Bmatrix}\underline{S}^{(+1)}\\ \underline{S}^{(-1)}\end{Bmatrix} + \underline{n}_k\right) =\\ &= e^{j\theta_k}\begin{Bmatrix}a_{12}\\ a_{22}\end{Bmatrix} + \underline{V}_2^H\cdot\underline{n}_k\end{aligned} \tag{13}$$

Thus, if +1 has been transmitted, we have:

$$\begin{aligned}l_{11} &= \underline{V}_1^H\cdot\underline{x}_k^{(+1)} = a_{11}\cdot e^{j\theta_k} + \underline{V}_1^H\cdot\underline{n}_k\\ l_{12} &= \underline{V}_2^H\cdot\underline{x}_k^{(+1)} = a_{12}\cdot e^{j\theta_k} + \underline{V}_2^H\cdot\underline{n}_k\end{aligned} \tag{14}$$

and it is defined the received vector $\underline{l}^{(+1)} = \begin{pmatrix}l_{11} & l_{12}\end{pmatrix}^T$ in the signal subspace. Analogously, if -1 is transmitted we obtain $\underline{l}^{(-1)} = \begin{pmatrix}l_{21} & l_{22}\end{pmatrix}^T$.

Therefore, if the received signal is multiplied in a symbol interval by the base vectors $\underline{V}_1$ and $\underline{V}_2$, the $\underline{l}_k$ vector can be defined as the vector that contains the components of this received signal in the signal subspace:

$$\underline{l}_k = \begin{pmatrix}\underline{V}_1^H\\ \underline{V}_2^H\end{pmatrix}\cdot\underline{x}_k \tag{15}$$

This vector $\underline{l}_k$ is 2-dimensional.

IV. ORTHOGONAL DECOMPOSITION, ML AND VITERBI

Since the received signal can be characterized by its projection in the signal subspace, the ML criteria can be applied to this projection instead of the direct signal (DS). Then, the minimum distance between the signal transmitted projection and the received signal projection will be looked for.

The possible transmitted signals are given in the interval branches, and in the anterior section were represented by (4). Now, the branches will be associated not to the entire signal; instead of that, the branches will be associated to its projection in the signal subspace:

$$\underline{S}'\,\text{branch}\left(\theta_{br}, \alpha_{br}\right) = e^{j\theta_{br}}\begin{Bmatrix}\underline{a}^{(+1)}\\ \underline{a}^{(-1)}\end{Bmatrix} \tag{16}$$

which will have 2x1 dimension. Therefore, the branch weights will be determinate by the Euclidean distance between the projection of received signal $\underline{l}_k$ and all the branch vectors:

$$W'\,\text{branch} = \left\| \underline{l}_k - \underline{S}'_{br} \right\|^2 \tag{17}$$

Now, the weight is given between 2x1 vectors which are the complete projection of $\underline{x}_k$ and $\underline{S}_{br}$ into the signal eigenvectors (Full OD).

However, the computational cost can be reduced if only the most important component is taken into account. That is, the 2x1 vectors are reduced to one dimension taking into account only the projection into the eigenvector associated to the principal eigenvalue, $\underline{V}_1$ (Partial OD):

$$w'_{branch} = \left\| l_{k,1} - a_{br,1} \cdot e^{j\theta_{br}} \right\|^2 \qquad (18)$$

Since each branch is associated to a definite weight, the ML criteria can be applied as was done in last section, deciding as transmitted symbol the ones associated to minimum weight branch given by (18).

The same way the weight of expression (18) is applied in ML criteria instead of expression (8), this kind of weight can be applied in Viterbi algorithm.

In both cases, the error probability will be higher than the error probability (BER) given by (18) (or the error probability given by (8). In computation simulating presents the same BER). The computational cost of the weights in a symbol interval will be modified as described at the following table:

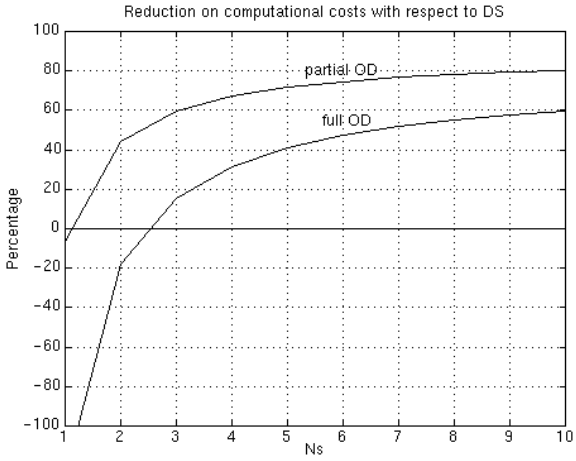
	DS	Full OD	Partial OD
Projection	0	2 x Ns p' 2x(Ns-1)a	Ns p' Ns-1 a
Subtract.	4 x Ns a	4 x 2 a	4 a
Module	4 x Ns p 4xNs-4 a*	4 x 2 p 4 x 1 a*	4 p
Total	16Ns p* 20Ns-4 a*	4Ns+32 p* 4Ns+32 a*	2Ns+16 p* 4Ns+14 a*

Table 1. Computational cost of weights in a symbol interval

Where "p" means product and "a" means adds between complex numbers. The superscript * refers to an operation between real numbers and ´ refers to operations between real and complex numbers. The total cost has been evaluated considering the cost of a complex multiplication (4 products and 2 adds).

V. RESULTS AND CONCLUSIONS

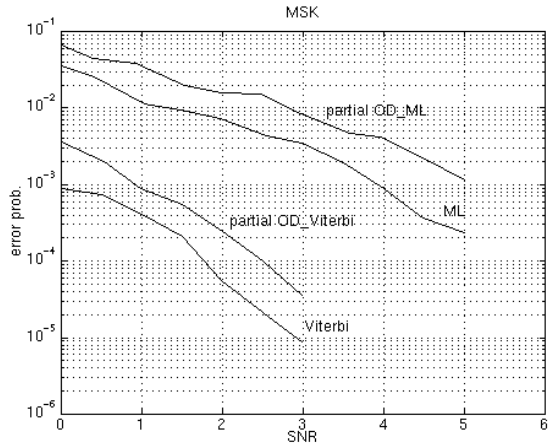
As it is deduced of table 1, the computational cost can be significantly reduced if a large number of samples in a symbol interval are performed. The following graphic shows the relative computational cost (assuming the same cost for adds and products) as function of Ns.



Graph. 1. Reduction on weight computation cost.

As can be seen, the number of operations can be reduced until 80% in partial OD schemes. If full OD schemes are used, the reduction becomes until 60%.

Since the partial OD does not take into account all the components into signal subspace, it can be expected a worst error probability. This is the price of reducing the computational cost. However, as can be seen in the following graphic, only approximately one dB is lost.



Graph. 2. Partial_OD vs. Full_OD.

The goal of this method consists in achieve a compromise between the computational cost and the error probability.

REFERENCES:

[1] Louis L. Scharf: "The SVD and Reduced Rank Signal Processing", *Signal processing* 25, Pg 113-133, 1991.

[2] R. Steele: "Mobile Radio Communications", Pentech Press, Chapter 6, 1992.

[3] G. David Forney: "The Viterbi Algorithm", *Proceedings of the IEEE*, vol. 61, Nº 3, March 1973.

[4] T. Aulin and C-E W. Sundberg: "Continuous Phase Modulation-Part I: Full Response Signaling", *IEEE Trans. On Commun.*, Vol 29, Nº 3, march 1981.